Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 319

Sichere Archivierung unter der Berücksichtigung der besonderen Anforderungen der Fertigung

Manuel Palenga

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr.-Ing. habil. Bernhard Mitschang
Betreuer/in:	Dr. rer. nat. Oliver Kopp, Dipl.-Inf. Andreas Bader
Beginn am:	7. April 2016
Beendet am:	6. Oktober 2016
CR-Nummer:	C.2.4, D.4.3, H.2.4

Kurzfassung

Anforderungen haben Einfluss auf das Design und die Funktionalitäten eines Softwaresystems. Eine steigende Zahl an gesetzlichen Anforderungen stellen Firmen vor Schwierigkeiten, da die Firmen diese Anforderungen erfüllen müssen und dabei die Kundenanforderungen nicht vernachlässigen wollen. Zusammen ergeben die Anforderungen ein genaues Bild wie ein Softwaresystem aussehen muss, um diesen Anforderungen zu entsprechen.

Traditionelle Archivierungssysteme erfüllen die gesetzlichen Anforderungen, aber nicht immer die neuen Kundenanforderungen. Neue Archivierungssysteme ermöglichen den Einsatz von neuer Technologie und die Erfüllung neuer Anforderungen. Das Problem ist, eine Technologie zu finden, welche die neuen Anforderungen erfüllt.

In dieser Arbeit wird ein Prototyp eines Archivierungssystem vorgestellt, welcher die gesetzlichen - und die Kundenanforderungen eines großen deutschen Unternehmens erfüllt. Weitere Anforderungen werden aus dem Mengengerüst von einem bestehenden Archivierungssystems gewonnen. Für das Design des Archivierungssystems werden zwei Technologien zur Datenhaltung benötigt, eine Meta-Datenbank und eine Archivierungsdatenbank. Als Technologien werden Datenbank- und Dateisysteme mit zwölf Kriterien evaluiert mit anschließender Diskussion. Die zwei benötigten Technologien zur Datenhaltung sind die Technologien, welche alle benötigten Kriterien erfüllt haben und die beste Performance aufweisen. Mit diesen Technologien wird ein Prototyp eines Archivierungssystems erstellt, wobei der Aufbau und die enthaltenen Daten beschrieben werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1. Einleitung

Die heutige Welt ist geprägt durch ihr steigendes Datenaufkommen, wobei ein Ende nicht in Sicht ist. Jedes zweite Jahr verdoppelt sich der Datenbestand eines Unternehmens [Ba14] und damit so schnell wie die Transistoren auf einem Chip [Moo06]. Überall werden Daten erfasst, ob in Wetterstationen, Verkehrsleitzentralen, im Social Media oder in der Industrie. Die Möglichkeiten Daten zu erfassen nimmt in der Zukunft zu, beispielsweise durch Industrie 4.0, wodurch noch mehr Daten verwaltet werden müssen [Ini14; WPG15].

Aber wieso müssen so viele Daten gespeichert werden? Firmen wollen einerseits Mehrwert aus den Daten ziehen und damit unter anderem mehr Umsatz machen. Jedoch werden nicht alle Daten gespeichert, gerade wenn es zu viele Daten sind, wie beispielsweise beim Teilchenbeschleuniger in Cern, wo nur ein einziges von 3 Millionen Ereignissen abgespeichert wird, wobei ein Ereignis etwa $5\,\mathrm{GB} - 10\,\mathrm{GB}$ hat. Das liegt daran, dass 600 Millionen Ereignisse pro Sekunde entstehen [CER12]. In den meisten anderen Fällen werden die Daten gespeichert und analysiert. Wobei das Speichern der Daten nicht immer auf freiwilliger Basis der Unternehmen geschieht, da hohe Kosten für die Speicherung anfallen können, sondern durch das Gesetz vorgegeben wird. Das liegt daran, dass für bestimmte Unterlagen Aufbewahrungsfristen existieren, welche eingehalten werden müssen um keine Geldbuße zahlen zu müssen [Kro10]. Dabei ist zu beachten, dass es auch Unterlagen gibt, welche archiviert werden müssen um sich gegen Produkthaftungsansprüche zu schützen, z.B. Fertigungsunterlagen [ED09].

Das ist der Fall in der Industrie, wobei viele Daten von hergestellten Produkten gesammelt werden, um die Qualität der Produkte nach der Fertigung nachweisen zu können. Daher müssen die gesammelten Qualitätsdaten archiviert werden, wobei die Archivierung selbst gesetzliche Anforderungen erfüllen muss [ED09].

Ein breites Spektrum an Archivierungssystemen existiert bereits, wobei die Anforderungen der Archivierung erfüllt werden. Dabei besitzen die Archivierungssysteme ein konkretes Anwendungsgebiet, wodurch kein universales Archivierungssystem existieren kann. Des Weiteren sind einige Archivierungssysteme nur Teil eines großen Produktes und können somit nicht einzeln genutzt werden.

In Rahmen dieser Arbeit wurde ein Prototyp eines Archivierungssystems entwickelt. Dieser kann Qualitätsdaten archivieren. Es werden jedoch nicht alle gesetzlichen - und Kundenanforderungen erfüllt, da es sich hierbei um einen Prototyp handelt. Ein aktuelles Mengengerüst [LL13] wurde aus dem bestehenden Archivierungssystem abgeleitet, wobei die Eigenschaften des Mengengerüsts beim zu entwickelnden Archivierungssystem berücksichtigt wurden. Des

Weiteren lässt sich durch die Archivierungslösung ein Mehrwert, der archivierten Daten, gewinnen.

Die Arbeit ist in folgender Weise gegliedert: In Kapitel 2 wird der Stand der Technik anhand von wichtigen Begriffen betreffend dieser Arbeit erklärt. In Kapitel 3 werden alle Anforderungen und Kriterien genannt, welche für das Archivierungssystem relevant sind. Im darauffolgenden Kapitel 4 werden Datenbanksysteme, Dateisysteme und existierende Archivierungssysteme ausgewählt und gegen die zuvor definierten Anforderungen und Kriterien evaluiert. Im Anschluss werden die Evaluationsobjekte verglichen und die beste Kombination der Evaluationsobjekte, welche sich für das Archivierungssystem am besten eignet, ausgewählt. In Kapitel 5 wird die zuvor ausgewählte Kombination prototypisch implementiert, wobei genauer auf den Aufbau und die Metadaten eingegangen wird. Zuletzt folgt das Kapitel 6, worin die Arbeit zusammengefasst wird und ein Ausblick gegeben wird.

2. Stand der Technik

Dieses Kapitel beschreibt für diese Arbeit grundlegende Begriffe, sowie den aktuellen Stand der Forschung in diesem Bereich. Daher wird als Erstes die Bedeutung von *Sichere Archivierung*, wie ein Teil des Titel dieser Arbeit lautet, in Abschnitt 2.1 definiert. Anschließend wird in Abschnitt 2.2 auf die Daten eingegangen, welche in der Fertigung entstehen und welche in dieser Arbeit durch ein Archivierungssystem archiviert werden sollen. Ein Computer Aided Quality-System (CAQ-System) ist ein System, welches bei Qualitätsmanagementaufgaben unterstützt. Für dieses CAQ-System werden Qualitätsdaten benötigt. Das CAQ-System wird in Abschnitt 2.3 erklärt. Da die Datenarchivierung ein Sonderfall der Datenspeicherung ist und zur Datenspeicherung oft Datenbanken genutzt werden, wird das *CAP-Theorem* in Abschnitt 2.4 vorgestellt, welches oft bei den Eigenschaften der Datenbank mit aufgelistet wird. Das CAP berücksichtigt nicht den partitionierungsfreien Bereich und die Latenz, wodurch die Erweiterung PACELC entstanden ist, welche in Abschnitt 2.5 erklärt wird. Anschließend werden in Abschnitt 2.6 verwandte Arbeiten zum CAQ-System aufgelistet.

Der Begriff Datenbank wird in dieser Arbeit als Synonym für das Datenbanksystem (DBS) genutzt, welches sich normalerweise aus der Datenbank (DB), der eigentliche Datenbestand, und dem Datenbankmanagementsystem (DBMS), eine Software zur Verwaltung von Datenbanken, zusammensetzt [HR01; Mit13]. Daraus wird klar, dass der umgangssprachliche Begriff „Datenbank“ und der Fachliteraturbegriff „DBMS“, beispielsweise Relationale Datenbankmanagementsysteme (RDBMS) und Time Series Databases (TSDB) [Bad16], normalerweise „DBS“ meint [Bad16].

2.1. Sichere Archivierung

Der Begriff „Sichere Archivierung“ wird unterschiedlich definiert, wobei der Kern des Begriffs, welcher im Folgenden erklärt wird, erhalten bleibt. Höllwart [Höl14] definiert im Kontext von Cloud Migration, dass sichere Archivierung vor den Risiken des Datenverlusts, Datenmanipulation und vor böswilligen Insidern schützt. Dr. Kampffmeyer [Kam03][Kam12] benutzt den Begriff als Abkürzung von *Revisionssichere (elektronische) Archivierung*, wobei dies erfüllt ist, wenn die VOI Richtlinien [VOI09], aufgelistet in Abschnitt 3.2, eingehalten werden. Der Begriff definiert wie bei Höllwart [Höl14] den Schutz vor Datenmanipulation und die technische Sicherheit. Dazu kommt der ganze Lebenszyklus von der Entstehung der Daten bis zur Löschung.

Der Begriff *Revisionssichere Archivierung* hat sich bereits in der Archivierungsbranche etabliert, da gesetzeskonforme Archivierungssysteme benötigt werden [Gru04; Kam03; SE05].

Diese Arbeit betrachtet nicht den kompletten Lebenszyklus der Daten, welcher in Abbildung 2.1 zu sehen ist, sondern ausschließlich den Teil der Archivierung. Aus diesem Grund wird in dieser Arbeit *Sichere Archivierung* in Anlehnung von Höllwart [Höl14] als *Archivierter Datenbestand vor Veränderungen und Unbefugten schützen.* definiert. Dadurch entsteht eine Abgrenzung zum Begriff *Revisionssichere Archivierung*.

Diese zwei Bereiche aus der Definition von *Sichere Archivierung* werden als externer und interner Schutz des Datenbestandes definiert. Dabei soll eine Datenmanipulation durch einen Benutzer oder durch das System verhindert werden können. Um daher ein sicheres Gesamtsystem gewährleisten zu können, müssen diese Bereiche separat betrachtet werden.

Externer Schutz des Datenbestandes

Ein Zugriffsschutz ermöglicht im ersten Schritt, dass nur authentifizierte Benutzer der Zugriff auf den Datenbestand gewährt wird. Im zweiten Schritt wird zwischen den Berechtigungen der Benutzer unterschieden, sodass nur bestimmte Benutzer beispielsweise lesen, schreiben und löschen dürfen. Durch diese zwei Schritte wird sichergestellt, dass keine unbefugten Benutzer den Datenbestand modifizieren oder im schlimmsten Fall löschen können.

Interner Schutz des Datenbestandes

Der interne Schutz des Datenbestandes definiert wie sich das System selbst vor Modifizierungen schützen kann. Dabei kann das System beispielsweise den Datenbestand durch verteilte Replikationen vor Auswirkungen von Netzwerkstörungen und Naturkatastrophen schützen.

2.2. Qualitätsdaten

Qualitätsdaten sind „Daten über die Qualität von Einheiten, über die bei der Ermittlung der Qualität angewendeten Qualitätsprüfungen und über die dabei herrschenden Randbedingungen sowie ggf. über die jeweiligen Einzelforderungen an die Qualitätsmerkmale“ [Gei07].

Maschinen erzeugen automatisch Qualitätsdaten bei der Fertigung von einem Produkt [BB15], wodurch sie zu den Betriebsdaten gezählt werden [Ker96; Kie10]. Sie dienen zur Qualitätskontrolle und zum Nachweis der Qualität eines Produktes [BB15; NSG+06]. Werden Qualitätsdaten mit Soll-Anforderungen ergänzt, werden diese Qualitätsinformationen genannt [BB15], wobei diese zum Nachweis von Produkt- und Prozesskonformität dienen [Ger08].

Sie enthalten Informationen von der Herstellung bis zur Logistik. Dabei werden alle durchlaufenen Stationen einer Produktionslinie mitsamt aller verbauten Komponenten protokolliert

[BB15]. Auch werden die Bestandteile von verbauten Komponenten protokolliert, welche wiederum Komponenten sein können. So kann beispielsweise rekursiv ermittelt werden, aus welchen einzelnen Komponenten ein Produkt besteht [BB15]. Ein wichtiger Bestandteil der Qualitätsdaten sind die Prüfergebnisse, welche Stationen oder Messanlagen zurückliefern können. Diese Prüfergebnisse enthalten Eigenschaften vom Produkt, beispielsweise wie stark eine Schraube angezogen wurde, und teilweise Soll- und Ist-Werte mit entsprechender Toleranz. Nachdem ein Produkt die Produktionslinie verlassen hat, werden Informationen der Logistik aufgezeichnet. [BB15]. Der Ablauf der Betriebsdatenerfassung (BDE) und die anschließende Speicherung der Qualitätsdaten mit der Anreicherung der Qualitätsdaten zu Qualitätsinformationen, welche die aktuelle Qualitätslage der Produktion bestimmt, bis zum Nutzen der Qualitätsdaten ist in Abbildung 2.2 zusehen.

Ein Beispiel ist ein unvollständiges Produkt, welches auf einem Förderband bzw. einer Produktionslinie liegt. Dieses Produkt gelangt durch das Förderband zu verschiedenen Stationen, wodurch neue Komponenten hinzugefügt werden. Beim Hinzufügen von Komponenten entstehen Qualitätsdaten, welche unter anderem die Komponente und das aktuelle Datum mit der aktuellen Uhrzeit beinhaltet. Eine weitere Station kann Datenwerte des Produkts mit den eingebauten Komponenten messen, so kann die Position und Ausrichtung eines eingebauten Steckers ermittelt werden. Diese Daten sind Prüfergebnisse, welche unter anderem den Toleranzbereich, beispielsweise die Positionsabweichung zum Optimum, enthalten können. Aber auch aktuelle Eigenschaften und der aktuelle Status einer Maschine sind möglich. Ist das Produkt vollständig, wird es verpackt und verladen, wodurch wiederum Qualitätsdaten entstehen, wie beispielsweise die Nummer der Palette, welche zum Transport des Produktes genutzt wird.

Der Qualitätsdaten-Lebenszyklus ist in Abbildung 2.1 dargestellt. In diesem Diagramm sind alle gesetzlichen Anforderungen bis auf eine einzige gesetzliche Anforderung berücksichtigt. Nicht berücksichtigt wird die Anforderung *Jedes Dokument muss in angemessener Zeit wiedergefunden und reproduziert werden können.* Darin ist zusehen, dass Qualitätsdaten von einer Maschine bzw. Station erstellt werden, wobei die erstellten Qualitätsdaten an ein System gesendet werden, welches sich um die Verwaltung von Qualitätsdaten kümmert. Dieses System archiviert die Qualitätsdaten und speichert diese parallel in eine separate Datenbank, sodass die Qualitätsdaten direkt genutzt werden können für beispielsweise die Prüfung der Qualität eines Produkts. Vorteil der Parallelität von Speicherung und Archivierung gegenüber einer Sequenz von Speicherung und anschließend Archivierung, ist, dass die Datenbank die Daten nicht sicher archivieren muss (siehe Abschnitt 2.1), da das Archivierungssystem die gleichen Daten sicher archiviert. Das liegt daran, da die Qualitätsdaten nicht nur im Archiv unverändert bleiben müssen, sondern auch von der Entstehung bis zur Archivierung, ausgenommen ist die Verdichtung, wodurch Soll-Anforderungen hinzukommen. Werden die Daten für die aktuelle Nutzung nicht mehr benötigt, so werden die nicht benötigten Qualitätsdaten aus der Datenbank gelöscht. Replikate existieren weiterhin im Archiv. Ist die Archivierungsdauer (siehe Abschnitt 3.2.1) abgelaufen, so werden die betreffenden Qualitätsdaten aus dem Archiv gelöscht. Nach der Löschung aus dem Archiv und der Datenbank sind die Qualitätsdaten nicht mehr verfügbar.

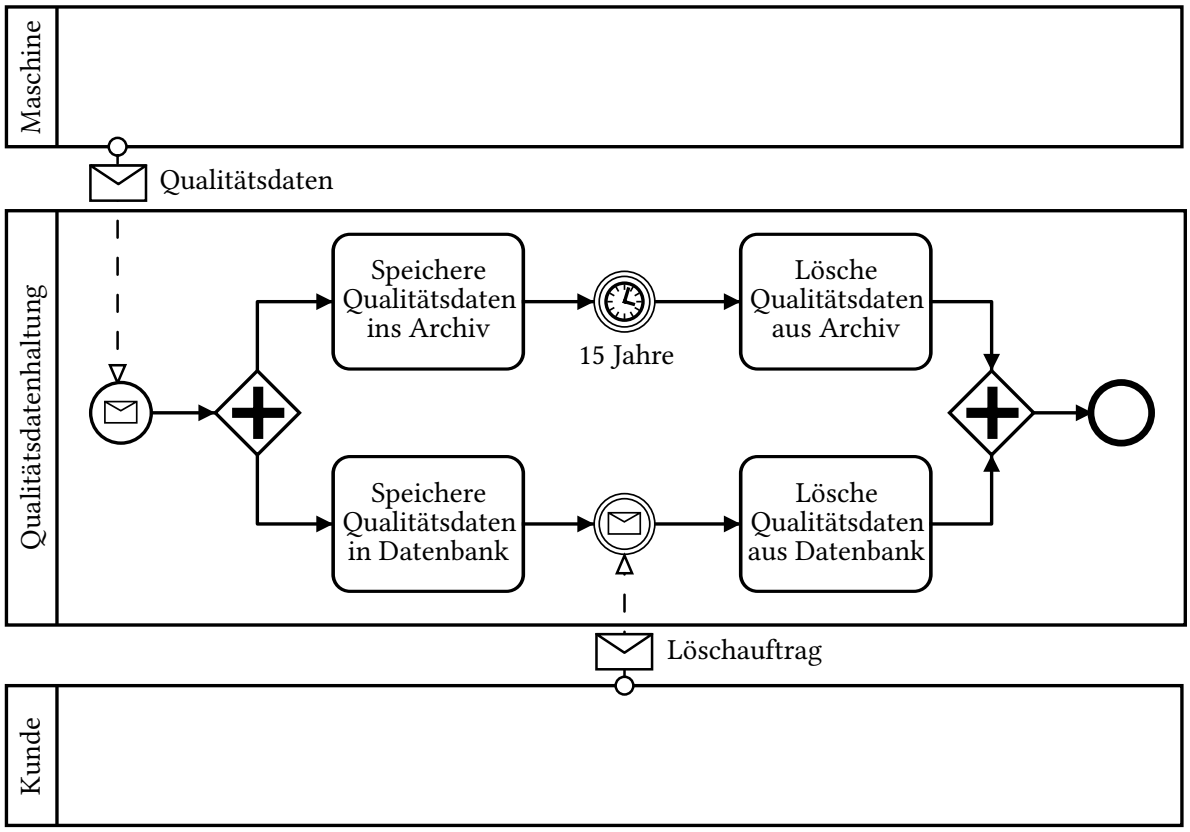

Abbildung 2.1.: Qualitätsdaten-Lebenszyklus – Von der Erstellung bis zur Löschung nach gesetzlichen Anforderungen

Qualitätsdaten dienen dazu einen Erzeugnis Lebenslauf eines Produkts nachverfolgen zu können [BB15]. Dadurch entstehen die beiden Anwendungsfälle[1]:

- Es können alle Produkte ermittelt werden, welche defekte Komponenten beinhalten, wodurch gezielte Rückrufaktionen durchgeführt werden können [BB15].
- Bei Reklamationen von defekten Produkten können alle verbauten Komponenten ermittelt werden und dadurch kann die defekte Komponente, welche den Schaden verursacht hat, ermittelt werden [BB15].

Somit können Qualitätsdaten der Rubrik Produktionsdokumente zugeordnet werden. Des Weiteren sind die Daten für die Produkthaftung (siehe Abschnitt 3.2) relevant, wodurch eine Nachweispflicht entsteht. Aus diesem Grund gehören Qualitätsdaten auch zu der Rubrik produkthaftungsrelevante Geschäftsunterlagen.

Die Qualitätsdaten sind für die Produkthaftung wichtig, da sie gem. § 1 Abs. 2 Nr. 2 ProdHaftG (Produkthaftungsgesetz), „Die Ersatzpflicht des Herstellers ist ausgeschlossen, wenn nach den

[1] „use cases“

Umständen davon auszugehen ist, daß das Produkt den Fehler, der den Schaden verursacht hat, noch nicht hatte, als der Hersteller es in den Verkehr brachte“, mögliche Produkthaftungsansprüche, von 85 Millionen Euro bei Personenschäden und keine Obergrenze bei Sachschäden, (§ 10 ProdHaftG) entgegenwirken können.

2.3. CAQ-System

Unter einem Computer Aided Quality-System (CAQ-System)[2] versteht man ein System, welches bei allen Qualitätsmanagementaufgaben unterstützt [HPJS08]. Aufgabenbereiche sind unter anderem das Risikomanagement, die Prüfplanung und die Prüfmittelverwaltung [BB15]. Ein Bestandteil des CAQ-Systems ist das Qualitätsinformationssystem (QIS), welches die Daten sammelt, speichert, verarbeitet, verdichtet und zuletzt archiviert [Ger08]. Der Aufbau und die Einordnung in den Qualitätsprozess ist in Abbildung 2.2 zu sehen, wobei ersichtlich wird, dass das CAQ-System den größten Teil des Prozesses ausmacht. Die *Soll-Anforderungen* machen aus den Qualitätsdaten die Qualitätsinformationen, womit die aktuelle Qualitätslage in der Fertigung ermittelt werden kann. Des Weiteren können Fehler in der Fertigung gefunden werden, welche anschließend behoben werden können und damit steigt die Qualität der Produkte.

Das Archivierungssystem wird im QIS angesiedelt, direkt nach den *Soll-Anforderungen*. Eine direkte Archivierung der Daten hätte die Vorteile, dass die Daten gem. § 239 Abs. 2 HGB zeitgerecht archiviert werden und die Daten direkt in einem Archivierungssystem, welche alle rechtlichen Anforderungen erfüllt, gespeichert werden. Dadurch muss die temporäre Zwischenspeicherung der Daten nicht die gesetzlichen Anforderungen der Archivierung erfüllen, wie beispielsweise die Unveränderbarkeit des Datenbestandes gem. § 146 Abs. 4 AO und § 239 Abs. 3 HGB. Die identifizierten Gesetze wurden von Dr. Kampffmeyer [Kam03] entnommen.

2.4. CAP-Theorem

Das CAP-Theorem [FBU+99] wurde Ende der 1990er von *Eric Brewer* als Vermutung aufgestellt. Diese Vermutung wurde im Jahr 2002 durch einen Beweis von *Seth Gilbert und Nancy Lynch* bestätigt und dadurch als Theorem etabliert. Das Theorem besagt, dass in einem verteilten System nur maximal zwei von den drei Eigenschaften Konsistenz (consistency, C), Verfügbarkeit (availability, A) und Ausfalltoleranz (partition tolerance, P) gleichzeitig gelten können. Dabei sind die Eigenschaften als graduelle Größen zu betrachten.

[2] Deutsch: Computerunterstützendes Qualitätssystem

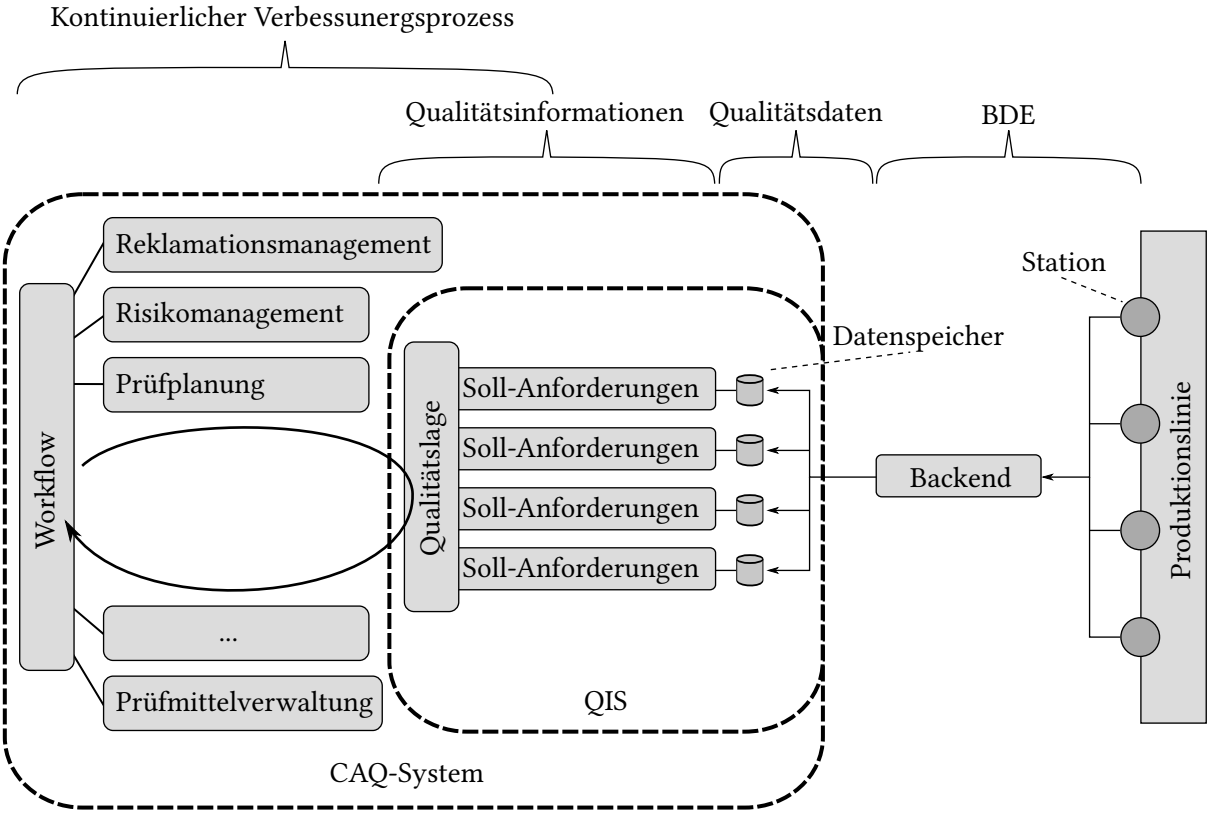

Abbildung 2.2.: Entstehung, Speicherung, Anreicherung und Nutzen von Qualitätsdaten – Nach Brüggemann und Bremer [BB15]

- **Konsistenz -** Alle Knoten sehen zum selben Zeitpunkt dieselben Daten. Daher muss nach jeder abgeschlossenen Transaktion sichergestellt werden, dass alle Replikationen auf den anderen Knoten aktualisiert werden. Dadurch wird erreicht, dass jeder Klient die gleichen Daten sieht. Die Konsistenz ist hoch, falls sofort nach Änderung des Datenbestandes die Konsistenz sichergestellt wird.
- **Verfügbarkeit -** Jede Anfrage an das System wird beantwortet. Dabei ist die Höhe der Verfügbarkeit abhängig von der Antwortzeit, wobei hohe Verfügbarkeit eine schnelle Antwortzeit bedeutet.
- **Ausfallsicherheit -** Das System läuft nach Ausfall von Netzknoten und Nachrichten fehlerfrei weiter. Dabei ist die Art des Ausfalls irrelevant, welche unter anderem eine Netzwerkstörung, ein Fehler auf dem Knoten oder eine Wartung sein kann.

Das CAP-Theorem wird oft als Dreieck dargestellt, wobei jede Kante eine der drei Eigenschaften darstellt. Dieses Dreieck mit den evaluierten Datenbanken ist in Abbildung 4.2 zusehen. Dabei werden die Anwendungen bzw. Datenbanken auf den Kanten, der jeweils erfüllenden Eigenschaften, geschrieben. Beispielsweise sind die traditionellen relationalen Datenbankmanagementsysteme (RDBMS) auf der Kante Verfügbarkeit und Konsistenz anzusiedeln, bei

diesen RDBMS ist die Ausfalltoleranz weniger relevant [HHGJ11]. Das liegt daran, dass viele RDBMS auf einem Knoten laufen und diese meistens hochverfügbare Infrastrukturen besitzen [HJ11].

2.5. PACELC

Eine Erweiterung von CAP ist PACELC, wobei das CAP-Theorem um die Eigenschaften aus einem partitionsfreien Betrieb ergänzt wird. Die ersten drei Buchstaben PAC definieren: Entscheidet sich ein System bei einer Netzwerkpartition (partition tolerance, P) für die Verfügbarkeit (availability, A) oder die Konsistenz (consistency, C). Die hinteren drei Buchstaben stehen für einen partitionsfreien Betrieb (else, E) mit der Eigenschaft, dass die Latenz (latency, L) gesenkt werden kann, wenn die Konsistenz abgeschwächt wird oder aber die Konsistenz (consistency, C) kann sichergestellt werden, wodurch sich die Latenz verschlechtern kann [Aba12; Ste14].

Das CAP-Theorem besagt nicht, was in einem partitionsfreien Betrieb passiert. Durch PACELC lässt sich definieren, ob ein System im Standardfall, den partitionsfreien Betrieb, die Konsistenz oder die Latenz bevorzugt. Dieser Kompromiss tritt bei verteilten Datenbanksystemen mit Replikationen auf [Aba12].

2.6. CAQ-Systeme zur Qualitätsverbesserung

In Abschnitt 2.3 wurden bereits CAQ-Systeme eingeführt. Deshalb werden in diesem Abschnitt verwandte Arbeiten vorgestellt, welche sich mit der Qualitätsverbesserung bei der Produktion beschäftigen.

Schwerdtfeger, Alt und Klinker [SAK07] beschäftigen sich mit der Informationsdarstellung mit *Augmented Reality* und *Mobile Vision*, wodurch Informationen direkt vor einem Mitarbeiter sichtbar werden. Dabei geht es um eine frühzeitige Fehlererkennung bei der Produktion, welche eine wichtige Rolle spielt, da so weiter die Produktion eines Produktes vorangeschritten ist desto mehr Kosten fallen an. Deshalb trägt jeder Mitarbeiter zur Qualitätsverbesserung bei. Diese Informationsdarstellung ist durch eine Interaktion mit dem CAQ-System möglich. Dadurch können Fehler früh einem Mitarbeiter sichtbar gemacht werden und die Tätigkeiten eines Mitarbeiters können durch zusätzliche Informationen ergänzt und verbessert werden.

Eine flexible Produktion mit kommunizierenden Produkten ermöglicht nach Prof. Dr. Glück [Glü12], im Jahre 2020, eine Optimierung der Produktion und Reduzierung der Kosten. Das wird durch verbaute Komponenten, welche kommunizieren können, oder durch Laserscanner erreicht. Dadurch können Maschinen unterschiedliche Aufgaben ausführen und individuell auf jedes Produkt reagieren. Erfüllt ein Produkt nicht die Soll-Anforderungen, so kann dieses dementsprechend angepasst werden, sodass die Soll-Anforderungen erfüllt werden.

Baku und Beus [BB05] beschreiben, wie schwierig es ist ein passendes CAQ-System zu finden, welches die jeweiligen Anforderungen erfüllt. Nicht jedes CAQ-System eignet sich für das Qualitätsmanagement einer bestimmten Branche, wie beispielsweise in der Medizintechnik. Deshalb muss aus den vielen Anbietern von CAQ-Systemen der passende Anbieter gefunden werden. Anschließend sollte eine Testphase mit dem CAQ-System gemacht werden, sodass geprüft werden kann, ob alle Anforderungen erfüllt werden.

Hannus et al. [HPJS08] weißen daraufhin, dass im Bereich der Lebensmittelindustrie die gestiegenen Anforderungen an das Qualitätsmanagement ein Grund zum Einsatz von CAQ-Systemen sind. Des Weiteren wird durch den Einsatz von CAQ-Systemen eine Rückverfolgung von Produkten ermöglicht.

Klöden et al. [KBSQ98] beschreiben wie Daten in der Lebensmittelindustrie durch den Einsatz von Qualitätsmanagementsystemen erfasst und ausgewertet werden können. Dadurch ist eine durchgängige Dokumentation der Qualität gesichert und es kann beispielsweise flexibler auf Prozessveränderungen reagiert werden.

3. Anforderungen an das Archivierungssystem

In diesem Kapitel werden die Anforderungen an ein effizientes und gesetzeskonformes Archivierungssystem aufgelistet und erklärt. Als Erstes wird das aktuelle Mengengerüst aus dem bestehenden Archivierungssystem in Abschnitt 3.1 aufgestellt und die resultierenden Anforderungen genannt. Anschließend werden die Anforderungen aus rechtlicher Seite und Kundenseite in Abschnitt 3.2 gegenübergestellt und zusammengefasst. Da das Archivierungssystem viele Daten aus unterschiedlichen Formaten speichern soll, wird in Abschnitt 3.3 Bezug auf Big Data und dessen charakterisierenden Vs gemacht. Das Archivierungssystem besteht aus zwei Datenbanken, einer Meta-Datenbank und einer Archivierungsdatenbank. Die Archivierungsdatenbank ist das Herzstück des Systems, da diese alle wertvollen Qualitätsdaten verwaltet. Dieser Aufbau wird in Abschnitt 3.4 definiert, wobei die zuvor genannten Anforderungen berücksichtigt wurden. Durch die unterschiedlichen Zwecke der Datenbanken, besitzen diese unterschiedliche Anforderungen und damit unterschiedliche Kriterien bei der Evaluation. Die Kriterien der Archivierungsdatenbank sind in Abschnitt 3.5 und die der Meta-Datenbank in Abschnitt 3.6 aufgelistet. In Abschnitt 3.7 werden alle Kriterien aufgelistet und in die Vs von Big Data, aus Abschnitt 3.3, und in die Kategorien von *Sichere Archivierung*, aus Abschnitt 2.1, eingeordnet.

3.1. Anforderungen aus dem Mengengerüst

Im Folgenden werden ein aktuelles Archivierungssystem und alle dazugehörigen Leistungsangaben von einem der weltweit größten Zulieferer der Automobilbranche für diese Arbeit zu Grunde gelegt.

Um ein aussagekräftiges Mengengerüst bestimmen zu können, wird als Erstes die Funktionsweise des aktuellen Archivierungssystems beschrieben. Damit kann im zweiten Schritt das aktuelle Archivierungssystem evaluiert werden, um Daten für das Mengengerüst zu erhalten. Nach der Evaluation sind die Mindestanforderungen bekannt, sodass das Archivierungssystem auch in der Zukunft seine Aufgabe erfüllen kann.

3.1.1. Aktuelles Archivierungssystem

Vor der Archivierung werden die Qualitätsdaten eines Produktes in einem komprimierten Archiv zusammengefasst, wodurch eine geringere Netzwerkbelastung entsteht und weniger Speicherplatz für die Archivierung benötigt wird. Anschließend sendet ein Service die Daten an das Archivierungssystem und löscht anschließend die Qualitätsdaten aus der Qualitätsdatenbank. Dieses komprimierte Archiv wird in einem Dateiverzeichnis zu den anderen archivierten Qualitätsdaten abgelegt, hierbei werden keine Metadaten separat abspeichert. Daher existiert ein Dateiverzeichnis mit vielen komprimierten Dateien und somit ist nicht klar in welcher Datei welcher Inhalt ist, ausgenommen das Produkt, da das Archiv nach dessen ID benannt ist.

Ein Beispiel, wird nach einer bestimmten Komponente gesucht, so müssen alle Archive entpackt werden und anschließend in die alte Form umgewandelt werden, sodass diese in die Qualitätsdatenbank gespeichert werden können. Anschließend kann nach der jeweiligen Komponente gesucht werden. Mit der steigenden Zahl an Archiven steigt auch die Zeit bis zum Auffinden der Komponente.

Das liegt auch daran, da die Qualitätsdaten über die ganze Archivierungsdauer im Dateiverzeichnis bleiben. Erst am Ende der Archivierungsdauer werden die jeweiligen Qualitätsdaten mit einem eigenständigen Service aus dem Archive gelöscht [PD16]. Das Auffinden von Daten führt zu einer neuen Anforderung des zu entwickelnden Archivierungssystem, wobei diese Anforderung durch die rechtlichen Anforderungen (siehe Abschnitt 3.2 Punkt 6) abgedeckt ist.

3.1.2. Mengengerüst

Das Mengengerüst der Archivierung ist abhängig vom Kunden, da dieser viele Konfigurationen vornehmen kann, da unterschiedlich viele Qualitätsdaten anfallen können. Der Unterschied kommt durch die Anzahl der Produkte, Komponenten, Linien, Stationen, gemessene Maschinenergebnisdaten und der Taktzeit, welche die Zeit definiert in der ein Produkt über die Linie läuft, zustande. Des Weiteren wird es beeinflusst, ob Daten nach einem bestimmten Zeitraum gelöscht werden oder nicht. Daher werden im Folgenden die Eigenschaften zu den leistungsintensivsten Kundenanforderungen betrachtet.

Daten

Für die Archivierung fallen unterschiedliche Dateien an, beispielsweise PDF, TXT und XML, welche zusammen die Qualitätsdaten eines Produktes bilden. Dabei bildet eine XML-Datei den Grundbaustein, welche Messwerte und eine hierarchische Gliederung, unter anderem mit den Komponenten und den durchlaufenen Stationen, enthält. Des Weiteren verweist die XML-Datei auf die zusätzlichen Dateien, welche beispielsweise bestimmten Komponenten

zuzuordnen sind. Nach dem Zippen der Qualitätsdaten eines Produktes liegt die Dateigröße der Archivdatei generell unter einem Megabyte. Da die Dateigröße in der Zukunft nicht gleich bleiben muss, sondern steigen kann, muss mit größeren Dateien gerechnet werden, wodurch das Archivierungssystem auch Daten über einem Megabyte archivieren können muss.

Datenmenge

Wie bereits erwähnt ist die Menge der Qualitätsdaten von vielen Parametern abhängig, so werden aktuell maximal $1\,\mathrm{TB}$ pro Monat an Daten erzeugt und archiviert. Eine Aussage, ob diese Datenmengen steigen oder sinken wird, kann nicht gemacht werden, da die Datenmenge abhängig vom Kunden ist [PD16]. Eine Umrechnung auf eine Sekunde ergibt eine erzeugte Datenmenge von circa $0{,}4\,\mathrm{MB}$. Wird über die unternehmensinterne Archivierungsdauer (siehe Abschnitt 3.2.1) jeden Monat $1\,\mathrm{TB}$ gespeichert, so sind das nach 15 Jahren $180\,\mathrm{TB}$. Definiert der Kunde eine eigene Archivierungsdauer, welche höher ist als die gesetzliche und interne Archivierungsdauer (siehe Abschnitt 3.2.1), dann kann mehr Speicherplatz benötigt werden.

Datenerzeugung

Die Taktzeit ist unter anderem dafür verantwortlich wie schnell ein komprimiertes Archiv erstellt werden soll, wenn der Kunde direkt am Ende der Linie die Qualitätsdaten archivieren lässt, wie in Abbildung 2.1 verdeutlicht wird. Eine Produktion kann mehrere Linien besitzen, wobei die Daten in dieselbe Qualitätsdatenbank gespeichert werden. Dabei können Taktzeiten von $0{,}3\,\mathrm{s}$ für eine Linie existieren [PD16]. Besitzt man mehrere Linien, so müssen die Daten in einem schnelleren Zyklus archiviert werden, wobei eine parallele Archivierung aktuell nicht vorgenommen wird.

Wählt der Kunde die Möglichkeit aus, dass Daten zuerst in die Qualitätsdatenbank gespeichert werden sollen und dass die Daten nach bestimmter Zeit archiviert werden sollen, so müssen in einem Zyklus die Qualitätsdaten ins Archivierungssystem archiviert und die Daten aus der Qualitätsdatenbank gelöscht werden. Daher muss das System eine hohe Ausführungsgeschwindigkeit haben um einen reibungslosen Archivierungsprozess gewährleisten zu können.

Bei zu geringer Ausführungsgeschwindigkeit werden die zu archivierten Daten in eine Warteschlange hinzugefügt und bei konstanter Taktzeit kann die Warteschlange nicht vollständig abgebaut werden, wodurch sich bei langer Laufzeit des Archivierungssystems viele Daten in der Warteschlange befinden. Das könnte dazu führen, dass Daten verloren gehen, wenn das Archivierungssystem unvorhergesehen abstürzt. Des Weiteren kann das Archivierungssystem nicht auf mehr Linien erweitert werden. Diese Probleme können durch eine hohe Ausführungsgeschwindigkeit und eine Skalierbarkeit des Archivierungssystems gelöst werden.

3.1.3. Anforderungen

Horizontale Skalierbarkeit

Das Datenaufkommen und die Taktzeit führen dazu, dass eine horizontale Skalierbarkeit von Nöten ist. Ansonsten können die unterschiedlichen Datenmengen nicht gespeichert werden, da zu wenig Speicherplatz zur Verfügung stehen würde und die Leistung für niedrige Taktzeiten nicht ausreichen würde. Eine reine vertikale Skalierung ist aus Kostengründen nicht empfehlenswert, da die Kosten nicht linear steigen [JF15]. Dazu kommt, dass sich die Rechenleistung eines einzelnen Servers nicht schneller steigern lässt, als sich die Rechenleistung verbessert, welche sich nach dem Mooreschen Gesetz alle zwei Jahre verdoppelt [Moo06]. Ein weiterer Nachteil ist die Erhöhung der Rechenleistung zur Laufzeit, da dies nicht ohne Probleme durchzuführen ist.

Bei der horizontalen Skalierung kann ein neuer Server leichter zur Laufzeit hinzugefügt werden. Des Weiteren kann die Verfügbarkeit des Systems bei Ausfall eines Knotens sichergestellt werden. Aus diesen Gründen kann eine ausschließlich vertikale Skalierung nicht diese Anforderung aus dem Mengengerüst erfüllen und damit wird eine horizontale Skalierbarkeit benötigt [Kau11; MMSW07].

Speicherkonzept für Dateigrößen

Die zu archivierenden Dateien müssen effizient gespeichert werden, wodurch die Anforderung entsteht, dass eine Archivierungsdatenbank ein geeignetes Speicherkonzept oder einen Datentyp für die jeweilige Dateigröße besitzen muss. Da die Qualitätsdaten laut Mengengerüst (siehe Abschnitt 3.1.2) generell unter einem Megabyte Speicher benötigen, muss die Archivierungsdatenbank kleine Dateien verwalten können. Um die Archivierungsdatenbank bei geänderten Dateigrößen der Qualitätsdaten weiterhin nutzen zu können, müssen auch Dateigrößen, über einem Megabyte, verwaltet werden können.

Die Anforderung ist beispielsweise nicht erfüllt, wenn die Archivierungsdatenbank eine minimale Blockgröße von einem Megabyte besitzt, wodurch jede Datei unter einem Megabyte trotzdem $1\,\mathrm{MB}$ Speicher benötigt, obwohl diese eventuell nur $1\,\mathrm{kB}$ besitzt.

3.2. Anforderungen von rechtlicher Seite und Kundenseite

Die Gesetze und Richtlinien für eine elektronische Archivierung werden in unterschiedlichen Büchern geregelt, welche wiederum in Grundsätzen zusammengefasst werden. Diese Grundsätze sind unter anderem *Grundsätzen zum Datenzugriff und zur Prüfbarkeit digitaler Unterlagen*

(GDPdU), *Grundsätzen ordnungsmäßiger Buchführung* (GoB) und *Grundsätzen ordnungsmäßiger DV-gestützter Buchführungssysteme* (GoBS) [VOI09].

Diese Grundsätze können nach *voice of information (VOI)* in zehn Merksätzen zusammengefasst werden. Dabei können nicht alle Merksätze auf das zu entwickelnde Archivierungssystem angewendet werden, deswegen wird bei den Merksätzen ergänzt, ob diese als Anforderungen für das Archivierungssystem relevant sind.

1. „**Jedes Dokument muss nach Maßgabe der rechtlichen und organisationsinternen Anforderungen ordnungsgemäß aufbewahrt werden.**“ [VOI09]: Es werden die internen Richtlinien abgedeckt, welche sich von den anderen Merksätzen nicht unterscheiden.

2. „**Die Archivierung hat vollständig zu erfolgen – kein Dokument darf auf dem Weg ins Archiv oder im Archiv selbst verloren gehen.**“ [VOI09]: Das Verlorengehen von Daten führt zu einem neuen Kriterium, welches durch die verteilte Datenspeicherung erreicht wird. Diese Anforderung wird bei den internen Anforderungen wie folgt definiert: „Die elektronische Datenhaltung muss redundant an zwei räumlich getrennten Brandabschnitten erfolgen.“ [Rob15a] Durch die verteilte Datenhaltung kann sichergestellt werden, dass der Datenbestand auch nach Ausfall eines Knotens durch beispielsweise Hardwareprobleme oder Naturkatastrophen nicht verloren geht.

3. „**Jedes Dokument ist zum organisatorisch frühestmöglichen Zeitpunkt zu archivieren**“ [VOI09]: Die frühestmögliche Archivierung ist individuell vom Kunden konfigurierbar und ist von der Umgebung des bestehenden Archivierungssystems geregelt. Das zu entwickelnde Archivierungssystem nutzt dieselbe Umgebung und daher wird dieser Merksatz nicht als Anforderung benötigt.

4. „**Jedes Dokument muss mit seinem Original übereinstimmen und unveränderbar archiviert werden**“ [VOI09]: Dieser Merksatz führt zu neuen Anforderungen, wobei keine Daten bei der Umwandlung des Dokuments in ein Archivierungsdatenformat verloren gehen dürfen. Dieser Merksatz deckt sich mit den internen Anforderungen, welche lautet „Eine notwendige Migration des Archivbestandes muss ohne inhaltliche Veränderungen der Informationen [...] durchführbar sein.“ [Rob15a] Des Weiteren entsteht die Anforderung, dass Daten unverändert archiviert werden müssen, welches mit der internen Anforderung „Die Dokumente müssen unveränderbar über die vorgesehene Archivierungszeit gehalten werden“ [Rob15a] übereinstimmt. Die Unveränderbarkeit wird auch in einer weiteren Anforderung erwähnt „Die eingesetzte Archivierungslösung [...] muss eine Informationsveränderung während der Archivierungsdauer verhindern.“ [Rob15a]

   Festl, Pott und Winzig [FPW93] beschäftigen sich mit dem Schutz der Daten vor Manipulation. Dabei werden unterschiedliche Sicherungsebenen, wie in Abbildung 3.1 zusehen sind, zum Schutz der Daten definiert, wobei die Ebene *Gerät* den größten Schutz bietet, da so weiter die Ebene von *Media* entfernt ist desto einfacher kann die Ebene umgangen

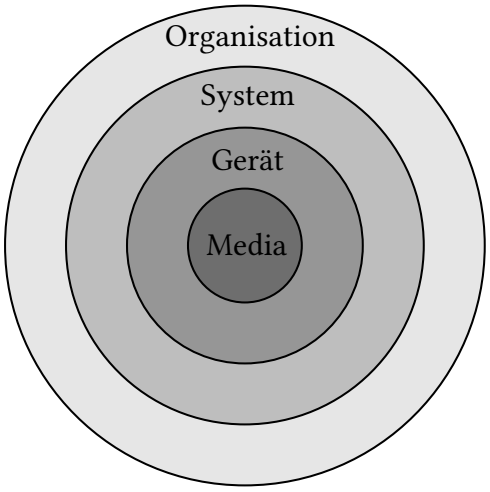

Abbildung 3.1.: Sicherungsebenen zum Schutz gegen Datenmanipulation – Nach Festl, Pott und Winzig [FPW93]

werden [FPW93]. *Gerät* definiert die eingesetzte Hardwarekomponente, wobei die Absicherung der Ebene mit Wirte Once Read Many (WORM) Technologien erfüllt wäre [BP97]. Die hier zuletzt definierte Anforderung „Unveränderbarkeit“ betrifft die Ebenen von *System* und *Organisation*.

5. „**Jedes Dokument darf nur von entsprechend berechtigten Benutzern eingesehen werden**“ [VOI09]: Es wird verlangt, dass Benutzer sich beim Archivierungssystem authentifizieren und autorisieren, wodurch eine neue Anforderung entsteht. Die Nutzung der Dokumente von berechtigten Benutzern wird auch in den internen Anforderungen genannt: „Der Zugriff auf die Archive muss über ein Rechte- und Archivierungskonzept sichergestellt werden.“ [Rob15a]

6. „**Jedes Dokument muss in angemessener Zeit wiedergefunden und reproduziert werden können**“ [VOI09]: „Alle archivierten Informationen müssen [...] in angemessener Zeit wieder verfügbar sein.“ [Rob15a] heißt es in den internen Dokumenten, wodurch eine neue Anforderung entsteht. Dadurch wird sichergestellt, dass benötigte archivierte Daten schnell genutzt werden können, ohne dass lange Zeit vonnöten ist um die Datei zu finden und lesbar zu machen. Dabei ist zu beachten, dass Archivierungssysteme viele Daten, mehrere Terabyte, enthalten können, wodurch höhere Abfragezeiten entstehen können.

7. „**Jedes Dokument darf frühestens nach Ablauf seiner Aufbewahrungsfrist vernichtet, d.h. aus dem Archiv gelöscht werden**“ [VOI09]: Dieser Merksatz ist für die bestehende Umgebung des Archivierungssystems relevant, welche beim neuen Archivierungssystem für die Einhaltung der Aufbewahrungsfrist genutzt werden kann. Dadurch entsteht keine Anforderung.

8. „**Jede ändernde Aktion im elektronischen Archivsystem muss für Berechtigte nachvollziehbar protokolliert werden**“ [VOI09]: Eine Protokollierung findet durch die Umgebung des Archivierungssystems statt, wodurch keine Anforderung an das zu entwickelnde Archivierungssystem entsteht.
9. „**Das gesamte organisatorische und technische Verfahren der Archivierung kann von einem sachverständigen Dritten jederzeit geprüft werden**“ [VOI09]: Der Merksatz dient dazu, dass eine ausführliche Dokumentation nachgewiesen werden kann. Daher führt der Merksatz zu keiner Anforderung an das Archivierungssystem.
10. „**Bei allen Migrationen und Änderungen am Archivsystem muss die Einhaltung aller zuvor aufgeführten Grundsätze sichergestellt sein**“ [VOI09]: Eine erneute Evaluation der genannten Merksätze bedeutet keine neue Anforderung für das Archivierungssystem.

Zusammengefasst ergeben sich aus den Merksätzen und internen Anforderungen folgende verpflichtende Anforderungen, welche das zu entwickelnde Archivierungssystem mindestens erfüllen muss:

1. Verteilte Datenspeicherung zur verlustfreien Datenhaltung über die Archivierungsdauer
2. Migration der Daten ohne inhaltliche Änderung
3. Unveränderbarkeit der Daten über die ganze Archivierungsdauer
4. Berechtigungsverwaltung für Benutzer
    4.1. Authentifizierung eines Benutzers
    4.2. Autorisierung eines Benutzers
5. Recherchemechanismus zur Datenabfrage in angemessener Zeit

Extra Anforderung: Mehrwertgewinn

Eine weitere Anforderung, welche aus rechtlicher Seite nicht relevant ist, jedoch aus Kundenseite, ist eine Verbesserung des aktuellen Archivierungssystem (siehe Abschnitt 3.1.1). Dabei soll ein Mehrwert aus den archivierten Daten gewonnen werden, wodurch unter anderem eine verbesserte Qualitätskontrolle entsteht. Aus diesem Grund soll das Archivierungssystem ein Mechanismus besitzen, um Auswertungen über die archivierten Daten zu ermöglichen. Diese Anforderung unterscheidet sich von der Anforderung „Recherchemechanismus zur Datenabfrage in angemessener Zeit“ darin, dass der eingesetzte Mechanismus dieser Anforderung länger benötigen darf, aber trotzdem mehr Ressourcen, beispielsweise CPU, in Anspruch nehmen darf. Existiert ein Mechanismus, welcher einen Mehrwert gewinnen lässt und die Daten in angemessener Zeit abfragen kann, dann sind beide Anforderungen erfüllt.

3.2.1. Archivierungsdauer

Das deutsche Gesetz schreibt Aufbewahrungsfristen für unterschiedliche Dokumente vor. Dabei stehen die rechtlichen Anforderungen der Dokumente in unterschiedlichen Gesetzbüchern, welche im Folgenden aufgelistet werden. Die relevanten Gesetze für die Archivierung von Qualitätsdaten (siehe Abschnitt 2.2), werden entsprechend angemerkt.

§ 257 HGB (Handelsgesetzbuch) – Kaufmänner sind verpflichtet bestimmte Dokumente für sechs und zehn Jahre aufzubewahren.

§ 14b UStG (Umsatzsteuergesetz) – Rechnungen müssen maximal zehn Jahre aufbewahrt werden.

§ 147 AO (Abgabenordnung) – Jahresabschlüsse, Geschäftsbriefe und weitere Unterlagen, welche bei der Besteuerung wichtig sind, müssen maximal 10 Jahre aufbewahrt werden.

§ 13 Abs. 1 ProdHaftG (Produkthaftungsgesetz) – Der Produkthaftungsanspruch erlischt zehn Jahre nachdem das Produkt in den Verkehr gebracht wurde. Dieses Gesetz betrifft die Qualitätsdaten.

Interne Regelung – Produkte, welche der Produkthaftung unterliegen, müssen 15 Jahre nach der Nutzung aufbewahrt werden [Rob15b]. Dieses Regelung betrifft die Qualitätsdaten.

3.3. Big Data und Archivierung

Big Data bedeutet, dass viele komplexe Daten in unterschiedlichen Formen, wie beispielsweise Bilder, Internetbeiträge oder Qualitätsdaten, auftreten und verarbeitet, analysiert und gespeichert werden müssen. Um mit solchen Daten umgehen zu können werden Big Data Technologien benötigt, welche sich im Laufe der Zeit vermehrt und verbessert haben. Dabei reichen die Technologien von der Datenhaltung in der Datenbank bis zur Visualisierung [Ba14].

Bei der Charakterisierung von Big Data helfen verschiedene Vs [KPM15]. Jedoch sind nicht alle für die Archivierung von Daten relevant. Im Folgenden sind die Vs aufgelistet und definiert, ob diese relevant sind.

- Die Menge[1] der zu archivierenden Daten. Aus dem Mengengerüst (siehe Abschnitt 3.1) wird klar, dass viele Daten anfallen, welche lange verwaltet werden müssen. Aus diesem Grund ist die Betrachtung der Menge von Daten äußerst wichtig [KPM15].

[1] „volume“

- Die Geschwindigkeit[2] wie Daten archiviert werden sollen. Die Geschwindigkeit der aufkommenden Daten kann laut Mengengerüst (siehe Abschnitt 3.1) sehr hoch sein. Daher ist die Geschwindigkeit ein wichtiges Big Data V [KPM15].
- Die Vielfalt[3] der Daten, wobei un-, semi- und strukturierte Daten, aber auch unterschiedliche Datenformate anfallen können. Diese Vielfalt ist beim Archivierungssystem durch die strukturierten Daten aus der Datenbank, PDF-Dokumente und binären Dateien gegeben. Die Vielfalt wird auch im Kontext von vielfältigen Informationsquellen verwendet, was hier nicht der Fall ist. Trotzdem ist die Vielfalt ein wichtiges V beim Archivierungssystem [KPM15].
- Der Wert[4] der Daten, wodurch Mehrwert entsteht. Die zu archivierenden Daten sind wie bereits in Abschnitt 3.2 zu sehen sind, von großer Wichtigkeit. Der meiste Mehrwert ist durch die Vermeidung von Produkthaftungsansprüchen gegeben, wodurch es das wichtigste V des Archivierungssystems ist [KPM15].
- Die Aufrichtigkeit[5] der Daten, wobei die Glaubwürdigkeit der Daten in Frage gestellt wird. Diese Glaubwürdigkeit ist bei den zu archivierenden Daten irrelevant. Das Archivierungssystem übernimmt selbst nicht die Prüfung der Daten, sondern ist nur für die Datenspeicherung zuständig. Die Glaubwürdigkeit der Daten muss vorher geprüft werden oder durch wohldefinierte Prozesse sicher gestellt werden [KPM15].

3.4. Aufbau des Archivierungssystems

Das bestehende Archivierungssystem (siehe Abschnitt 3.1.1) muss aus betrieblichen Gründen durch ein neues Archivierungssystem ersetzt werden. Dabei soll das neue Archivierungssystem eine Speicherung von Metadaten erlauben, sodass nach Qualitätsdaten schneller und effizienter, im Vergleich zu einer Suche über alle Qualitätsdaten, gesucht werden kann. Im zweiten Schritt können über die verbleibenden Daten, welche nicht in den Metadaten enthalten sind, gesucht werden. Dieser Aufbau des Archivierungssystems würde das verpflichtende Kriterium „Recherchemechanismus zur Datenabfrage in angemessener Zeit“ erfüllen.

Aus diesem Grund wird für das Archivierungssystem eine Datenspeicherlösung gesucht, welche Metadaten und Qualitätsdaten speichern kann oder zwei unterschiedliche Datenspeicherlösungen, einer Meta-Datenbank und einer Archivierungsdatenbank. Dabei dient die Archivierungsdatenbank zur eigentlichen Speicherung aller Daten, wobei nicht nur strukturierte Daten anfallen, sondern auch semi- und unstrukturierte Daten.

[2] „velocity“
[3] „variety“
[4] „value“
[5] „veracity“

Die Meta-Datenbank speichert redundant die wichtigsten Bestandteile der Archivierungsdatenbank. Dabei werden ausschließlich strukturierte Daten gespeichert. Durch die Meta-Datenbank ist es möglich effizient die Daten in der Archivierungsdatenbank zu finden, da ausschließlich auf einem kleinen Teil der Daten Operationen ausgeführt werden müssen, welche zudem in strukturierter Form vorliegen. Um diese Beziehung zwischen den Datenbanken herstellen zu können, wird in der Meta-Datenbank eine Referenz auf die Daten aus der Archivierungsdatenbank benötigt. Das hat den weiteren Vorteil, dass weniger Lesezugriffe auf die Archivierungsdatenbank durchgeführt werden müssen.

Da der Zweck der beiden Datenbanken unterschiedlich ist, gelten für Beide unterschiedliche Kriterien für die Wahl der Datenbankprodukte, wobei beides mal die verpflichtenden Kriterien aus den „Anforderungen von rechtlicher Seite und Kundenseite“ zu Grunde gelegt werden. Die Kriterien der Archivierungsdatenbank werden in Abschnitt 3.5 und die Kriterien für die Meta-Datenbank in Abschnitt 3.6 aufgelistet. Für eine einzelne Datenspeicherlösung müssen alle Kriterien beider Datenbanken erfüllt sein.

3.5. Kriterien zur Evaluation von Archivierungsdatenbanken

Für die Auswahl der richtigen Archivierungsdatenbank müssen unterschiedliche Kriterien, welche im Folgenden aufgelistet sind, erfüllt sein. Dadurch ist gegeben, dass alle Anforderungen aus rechtlicher Sicht erfüllt sind.

1. **Authentifizierung des Nutzers** Den Zugriff auf die Datenbank dürfen ausschließlich authentifizierten Nutzern gestattet werden. Daher muss ein Mechanismus zur Authentifizierung existieren. Die zu Grunde liegende verpflichtende Anforderung ist „Authentifizierung eines Benutzers“.
2. **Autorisierung des Nutzers** Eine Rechteverwaltung muss Nutzern oder Nutzergruppen Berechtigungen für Ressourcen geben können. Sodass Nutzer oder Nutzergruppen nur berechtigte Aktionen, wie beispielsweise lesen und schreiben ausführen dürfen. Dieses Kriterium stellt sicher, dass die verpflichtende Anforderung „Autorisierung eines Benutzers“ betrachtet wird.
3. **Speicherung von Dokumenten** Dateien und Dokumente sollen abgespeichert werden können, welche unter anderem semi- und unstrukturierte Daten sind oder enthalten können.
4. **Migration ohne Datenverlust** Die zu archivierenden Daten dürfen bei der Umwandlung des Datenformates in ein mögliches Datenbankformat keine Informationen verlieren. Dieses Kriterium stellt sicher, dass die verpflichtende Anforderung „Migration der Daten ohne inhaltliche Änderung“ betrachtet wird.

5. **Datenmodell für kleine Daten** Kleine Daten, weniger als 1 MB (siehe Abschnitt 3.1.2), müssen durch vorhandene Datentypen abgespeichert werden können. Es muss für jede Dateigröße einen Datentypen geben, welcher laut Dokumentation keine negativen Auswirkungen auf das System besitzt und für die jeweilige Dateigröße geeignet ist. Negative Auswirkungen sind unter anderem, dass mehr Speicherplatz allokiert wird durch definierte Größe des Clusters und Arbeitsspeicher kontinuierlich für jede Datei allokiert wird. Diese Datentypen müssen Kriterium 3 (*Speicherung von Dokumenten*) erfüllen. Die Einhaltung dieses Kriteriums bewirkt, dass die Anforderung „Speicherkonzept für Dateigrößen“ aus dem Mengengerüst erfüllt wird.

6. **Datenmodell für große Daten** Nicht nur kleine Daten können bei der Archivierung anfallen, da das System erweitert werden könnte, wodurch größere Daten, zwischen 1 MB (siehe Abschnitt 3.1.2) und 10 GB anfallen könnten. Die Obergrenze von 10 GB wurde auf Basis der Erfahrungen der zuständigen Mitarbeiter gewählt [PD16]. Es muss für jede Dateigröße einen Datentypen geben, welcher laut Dokumentation keine negativen Auswirkungen auf das System besitzt und für die jeweilige Dateigröße geeignet ist. Negative Auswirkungen sind unter anderem, dass mehr Speicherplatz allokiert wird durch definierte Größe des Clusters und Arbeitsspeicher kontinuierlich für jede Datei allokiert wird. Diese Datentypen müssen Kriterium 3 (*Speicherung von Dokumenten*) erfüllen. Die Einhaltung dieses Kriteriums bewirkt, dass die Anforderung „Speicherkonzept für Dateigrößen“ aus dem Mengengerüst erfüllt wird.

7. **Skalierbarkeit** Die Datenbank muss sich individuell an die Menge der Daten anpassen lassen, indem der Speicherplatz erweitert wird und die Leistung erhöht wird. Somit wird eine horizontale Skalierbarkeit der Datenbank vorausgesetzt, welche ermöglicht, dass weitere Server dem bestehenden Gesamtsystem hinzugefügt werden können. Dieses Kriterium soll sicherstellen, dass die Anforderung „Horizontale Skalierbarkeit“ aus dem Mengengerüst und teilweise die verpflichtende Anforderung „Recherchemechanismus zur Datenabfrage in angemessener Zeit“ erfüllt wird.

8. **Verteilte Replizierung** Für eine sichere Archivierung müssen Daten redundant und verteilt auf verschiedenen Knoten gespeichert werden. So wird der Datenbestand bei einem Ausfall, temporär oder permanent, eines Knotens, beispielsweise durch Auswirkungen von Netzwerkstörungen oder Naturkatastrophen, nicht beeinträchtigt. Dieses Kriterium stellt sicher, dass die verpflichtende Anforderung „Verteilte Datenspeicherung zur verlustfreien Datenhaltung über die Archivierungsdauer“ betrachtet wird.

9. **Verteilter Datenauswertungsmechanismus** Ein verteilter Abfragemechanismus zur Datenauswertung muss existieren, sodass parallel Operationen auf unterschiedlichen Knoten durchgeführt werden können. Dieses Kriterium deckt die Anforderung „Mehrwertgewinn“ ab.

10. **Unveränderbarkeit** Archivierte Daten dürfen über die komplette Zeit der Archivierung nicht verändert werden. Daher müssen Mechanismen zur Verhinderung von Veränderungen des Archivierungsbestandes existieren. Dieses Kriterium stellt sicher, dass die verpflichtende Anforderung „Unveränderbarkeit der Daten über die ganze Archivierungsdauer“ betrachtet wird.

3.6. Kriterien zur Evaluation von Meta-Datenbanken

Die Meta-Datenbank ist Teil des Archivierungssystems und dieses muss als Ganzes den Sicherheitsanforderungen erfüllen. Daher müssen ähnliche Kriterien, wie bei der Archivierungsdatenbank in Abschnitt 3.5 erfüllt sein.

1. **Authentifizierung des Nutzers** Die Metadaten sind wichtiger Bestandteil der Daten und deshalb dürfen diese ausschließlich von authentifizierten Personen erstellt und genutzt werden. Aus diesem Grund gilt Kriterium 1 (*Authentifizierung des Nutzers*) der Archivierungsdatenbank.
2. **Autorisierung des Nutzers** Die Meta-Datenbank kann von verschiedenen authentifizierten Benutzern genutzt werden, wodurch Berechtigungen benötigt werden, sodass nur bestimmte Benutzer Aktionen ausführen dürfen. Ansonsten könnten Benutzer ohne Erlaubnis Daten modifizieren oder einfügen. Für die Meta-Datenbank gilt ebenso Kriterium 2 (*Autorisierung des Nutzers*) der Archivierungsdatenbank.
3. **Speicherung von Dokumenten** Dieses Kriterium wird durch das speziellere Kriterium 11 (*Hierarchische Datenspeicherung*) ersetzt. Womit dieses Kriterium für die Evaluation der Meta-Datenbank nicht benötigt wird.
4. **Migration ohne Datenverlust** Falsche oder unvollständige Informationen in der Meta-Datenbank wären fatal für das Archivierungssystem, da falsch positive und falsch negative Resultate entstehen können. Falsch positive Resultate machen unnötige Arbeit, während falsch negative Resultate dazu führen, dass bestimmte Daten nicht gefunden werden [LL13; Wag14]. Gleiches gilt auch für die Referenz, welche auf die Qualitätsdaten in der Archivierungsdatenbank zeigt. Aus diesem Grund muss die Meta-Datenbank Kriterium 4 (*Migration ohne Datenverlust*) der Archivierungsdatenbank erfüllen.
5. **Datenmodell für kleine Daten** Das Kriterium wird nicht benötigt, da sich dieses auf die Datentypen von Kriterium 3 (*Speicherung von Dokumenten*) bezieht, wobei diese Datentypen bei der Speicherung der Daten in der Meta-Datenbank nicht eingesetzt werden dürfen, siehe Kriterium 11 (*Hierarchische Datenspeicherung*). Da die Metadaten trotzdem klein sind, übernimmt Kriterium 11 (*Hierarchische Datenspeicherung*) die Prüfung auf kleine Daten.

6. **Datenmodell für große Daten** Dieses Kriterium findet bei der Meta-Datenbank keine Anwendung, da es sich bei Metadaten um kleine Daten handelt und damit wird kein Datenmodell für große Daten benötigt.

7. **Skalierbarkeit** Wie bereits aus dem Mengengerüst (siehe Abschnitt 3.1.2) bekannt ist, existiert ein hohes Datenaufkommen. Für alle Qualitätsdaten fallen Metadaten an, welche in der Meta-Datenbank zu speichern sind. Aus diesem Grund muss genügend Speicherplatz für die Daten zur Verfügung stehen, dies wird durch eine horizontale Skalierbarkeit erreicht. Somit muss Kriterium 7 (*Skalierbarkeit*) aus der Archivierungsdatenbank ebenfalls erfüllt sein.

8. **Verteilte Replizierung** Die Metadaten sind wegen der verpflichtenden Anforderung „Recherchemechanismus zur Datenabfrage in angemessener Zeit“ sicher zu speichern. Daher müssen Metadaten verteilt und repliziert gespeichert werden, wodurch beispielsweise Auswirkungen von Serverausfällen gering bleiben, da alle betroffenen Metadaten auf anderen Servern verfügbar sind. Deswegen gilt Kriterium 8 (*Verteilte Replizierung*) der Archivierungsdatenbank.

9. **Verteilter Datenauswertungsmechanismus** Dieses Kriterium wird nicht benötigt, da Kriterium 12 (*Abfragesprache für hierarchische Daten*) für einen Mehrwert der archivierten Daten sorgt und daher kein weiterer Mechanismus zur Datenabfrage existieren muss.

10. **Unveränderbarkeit** Für die Metadaten gilt ebenso Kriterium 10 (*Unveränderbarkeit*), da das Archivierungssystem als Gesamtsystem betrachtet wird, womit alle Daten unverändert bleiben müssen.

11. **Hierarchische Datenspeicherung** Hierarchisch strukturierte Daten müssen abgespeichert werden können, wobei die Daten klein sein können. Die Daten dürfen vor der Speicherung nicht in einen anderen Datentyp umgewandelt werden, wie beispielsweise in ein Binary Large Object (BLOB). Dabei ist zu beachten, dass unterschiedliche Datenstrukturen vorkommen können, da beispielsweise Daten unvollständig oder nicht existieren können.

12. **Abfragesprache für hierarchische Daten** Eine Abfragesprache muss existieren mit der hierarchische Daten geschrieben und ausgewertet werden können. So soll beispielsweise die Gleichheit von Werten überprüft werden können. Dadurch können mit Filtereinstellungen bestimmte Datensätze ermittelt und zurückgegeben werden.

3.7. Zusammenfassung der Evaluierungskriterien

Die Kriterien der Archivierungsdatenbank aus Abschnitt 3.5 und der Meta-Datenbank aus Abschnitt 3.6 werden in der Tabelle 3.1 zusammengefasst. Wird das Kriterium für die jeweilige

Evaluation der Datenbank benötigt, so weißt ✓in der Tabelle daraufhin. Wird das Kriterium nicht genutzt, ist ein ✗gesetzt. Mit den aufgelisteten Kriterien wird jede Datenbank evaluiert.

	Archivierungs-DB	Meta-DB
Kriterium 1 (*Authentifizierung des Nutzers*)	✓	✓
Kriterium 2 (*Autorisierung des Nutzers*)	✓	✓
Kriterium 3 (*Speicherung von Dokumenten*)	✓	✗
Kriterium 4 (*Migration ohne Datenverlust*)	✓	✓
Kriterium 5 (*Datenmodell für kleine Daten*)	✓	✗
Kriterium 6 (*Datenmodell für große Daten*)	✓	✗
Kriterium 7 (*Skalierbarkeit*)	✓	✓
Kriterium 8 (*Verteilte Replizierung*)	✓	✓
Kriterium 9 (*Verteilter Datenauswertungsmechanismus*)	✓	✗
Kriterium 10 (*Unveränderbarkeit*)	✓	✓
Kriterium 11 (*Hierarchische Datenspeicherung*)	✗	✓
Kriterium 12 (*Abfragesprache für hierarchische Daten*)	✗	✓

Tabelle 3.1.: Zusammenfassung der Kriterien von Archivierungs- und Meta-Datenbank

In der Tabelle 3.2 werden die Kriterien durch die Big Data Vs (siehe Abschnitt 3.3) charakterisiert und in den internen bzw. externen Schutz des Datenbestandes (siehe Abschnitt 2.1) eingeordnet.

Hierbei wird deutlich, dass der Wert der Daten an erster Stelle steht, da sechs von zwölf Kriterien den Wert der Daten sicherstellen sollen. Zwar sind die anderen verbleibenden Kriterien für die Einhaltung des Wertes der Daten auch relevant, jedoch nicht so stark wie die genannten Charakterisierungen. Beispielsweise dient Kriterium 7 (*Skalierbarkeit*) in erster Linie dazu, dass viele Daten gespeichert werden können und die Abfrage der Daten in angemessener Zeit Daten zurückliefert. Jedoch steht erst an zweiter Stelle, dass dadurch der Wert der Daten sichergestellt wird und somit ist für das Kriterium der Wert nicht das Hauptmerkmal.

Die Einordnung in den internen bzw. externen Schutz des Datenbestandes zeigt, dass Kriterien aus beiden Bereichen existieren müssen, sodass zusammen alle verpflichteten Kriterien erfüllt werden können.

	Schutz der Datenhaltung	Big Data Vs
Kriterium 1 (*Authentifizierung des Nutzers*)	extern	Wert
Kriterium 2 (*Autorisierung des Nutzers*)	extern	Wert
Kriterium 3 (*Speicherung von Dokumenten*)		Vielfalt
Kriterium 4 (*Migration ohne Datenverlust*)	intern	Wert
Kriterium 5 (*Datenmodell für kleine Daten*)		Größe
Kriterium 6 (*Datenmodell für große Daten*)		Größe
Kriterium 7 (*Skalierbarkeit*)		Geschwindigkeit, Größe
Kriterium 8 (*Verteilte Replizierung*)	intern	Wert
Kriterium 9 (*Verteilter Datenauswertungsmechanismus*)		Wert
Kriterium 10 (*Unveränderbarkeit*)	intern	Wert
Kriterium 11 (*Hierarchische Datenspeicherung*)		Vielfalt
Kriterium 12 (*Abfragesprache für hierarchische Daten*)		Vielfalt

Tabelle 3.2.: Einordnung der Kriterien in den internen bzw. externen Schutz des Datenbestandes und in die Big Data V Charakterisierung

4. Evaluation von Datenbanken

In diesem Kapitel soll das Herzstück des Archivierungssystems, das Modell und die Software der Datenhaltung, gefunden werden. Aus diesem Grund werden Datenbanken und Dateisysteme nach den Kriterien aus Tabelle 3.1 evaluiert. Die Evaluation wird mit technischen Dokumentationen zu dem jeweiligen Produkt durchgeführt, dabei werden keine Erweiterungen betrachtet, da viele Erweiterungen zu einem Produkt existieren können. Anschließend werden die evaluierten Produkte verglichen um die Produkte zu finden, welche die meisten Kriterien der Archivierungsdatenbank und der Meta-Datenbank erfüllen. Da bereits Archivierungssysteme existieren, werden einige von diesen auch nach den Kriterien evaluiert um am Ende diese mit den zuvor evaluierten Produkten zu vergleichen.

Die Datenbanken und Dateisysteme werden in Abschnitt 4.1 ausgewählt. Die ausgewählten Produkte werden anschließend in den unterteilten Kategorien evaluiert, wobei die Evaluation der Datenbanken in den Abschnitte 4.2 bis 4.6 stattfindet und die Evaluation der Dateisystemen in Abschnitt 4.7. Anschließend werden die existierenden Archivierungssysteme in Abschnitt 4.8 evaluiert. Zum Schluss werden die Ergebnisse der gesamten Evaluation in Abschnitt 4.9 zusammengefasst und ausgewertet.

4.1. Auswahl der Evaluationsobjekte

Daten können in unterschiedlichen Systemen abgespeichert werden. So können Daten, wie beim aktuellen Archivierungssystem, direkt auf einem Dateisystem abgelegt werden oder es können Datenbanken genutzt werden. Eine andere Alternative wäre ein existierendes Archivierungssystem zu nutzen.

Es stehen mehrere Dateisysteme und Datenbanken zur Auswahl, welche sich beispielsweise bei der Datenstruktur und bei der Datenspeicherung unterscheiden [Gar10]. Als Erstes wird auf die Datenbanken in Abschnitt 4.1.1 eingegangen und anschließend in Abschnitt 4.1.2 auf die Dateisysteme, wobei jeweils Evaluationsobjekte ausgewählt werden. Für die Alternativlösung werden existierende Archivierungssysteme in Abschnitt 4.1.3 ausgewählt. Zusammengefasst werden die Evaluationsobjekte in Abschnitt 4.1.4.

4.1.1. Datenbanken

Bei den Datenbanken wird zwischen den Kategorien relationale Datenbanken, welche größtenteils mit der Datenbanksprache Structured Query Language (SQL) abgefragt werden können, und Not only SQL (NoSQL)-Datenbanken unterschieden [Gar10]. NoSQL-Datenbanken können wiederum in Kategorien eingeteilt werden, wobei die folgenden Kategorien laut Hecht und Jablonski [HJ11] existieren.

- **Dokumentenorientierte Datenbanken**: In diesen Datenbanken können hierarchische Dokumente mit bestimmten Datenformaten verwaltet werden. Datentypen können unter anderem JSON und XML sein [HJ11].
- **Spaltenorientierte Datenbanken**: Die gespeicherten Daten werden nicht zeilenweise wie bei den relationalen Datenbanken abgespeichert [Gar10], sondern spaltenweise. Dadurch werden bei Schreib- und Lesezugriffen auf einzelne Spalten nur benötigte Daten ausgewertet [HJ11].
- **Key-Value Datenbanken**: Die Datenspeicherung erfolgt mit einem eindeutigen Schlüssel, welcher auf die Daten verweist. Diese Speicherung ist identisch mit dem Datentyp *Map* [HJ11].
- **Graphenorientierte Datenbanken**: Bei vielen Beziehungen zwischen Datensätzen können die Daten effizient verwaltet werden [HJ11]. Operationen auf Graphen, wie beispielsweise der Dijkstra-Algorithmus [Dij59], können optimal eingesetzt werden, da ein Graph bereits existiert und nicht wie bei anderen Datenbanken erstellt werden muss.

Um einen möglichst großen Bereich bei der Evaluation abzudecken, werden aus jeder dieser Kategorien mindestens eine Datenbank, sowie mindestens eine relationale Datenbank ausgewählt. Unterschiedliche Dateisysteme vervollständigen die Evaluation.

Da ein großes Spektrum an Datenbanken existiert, müssen geeignete Datenbanken für die Evaluation ausgesucht werden. Aus diesem Grund wurde eine Google-Suche durchgeführt, welche in Anhang A.1 mit den Resultaten aufgeführt ist, um existierende Datenbanken zu finden.

Die Vielzahl an Datenbanken führt dazu, dass nur die besten Datenbanken aus jeder Kategorie evaluiert werden können. Diese Auswahl wird anhand von unterschiedlichen Ranglisten und Empfehlungen getroffen.

G2 Crowd, Inc. [GC15] hat eine Rangliste zu den besten NoSQL-Datenbanken anhand von Benutzerkritiken erstellt, wobei das Resultat in Abbildung 4.1 zu sehen ist. Laut QuinStreet Inc. [QS16] sollen Entwickler mit den folgenden NoSQL-Datenbanken umgehen können, MongoDB, Redis, Cassandra, CouchDB und HBase. Die Resultate aus den beiden Ranglisten wurden in Tabelle 4.1 zusammengefasst, wobei ein + bei einer genannten Datenbanken steht. Des Weiteren wurde bei aufgelisteten Datenbanken die Popularität durch *DB-Engines* [SI16b] bestimmt, dazu wurde jeweils die Datenbank in ihre zugehörige Kategorie eingeteilt und

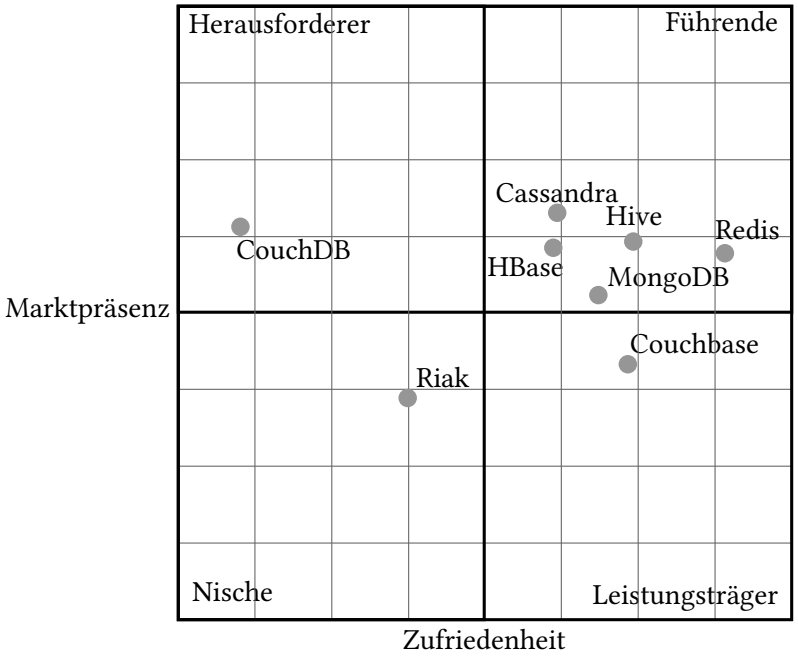

Abbildung 4.1.: G2 Crowd Grid – Beste NoSQL-Datenbanken – Nach G2 Crowd, Inc. [GC15]

anschließend wurde dessen Ranglistenplatz, in Tabelle 4.1, notiert. *DB-Engines* ermittelt die Popularität unter anderem mit Google Trends und den Anzahl der Suchergebnisse auf Google und Bing [SI16a].

Tabelle 4.1 zeigt, dass alle drei Quellen die populärsten Datenbanken der jeweiligen Kategorie übereinstimmend ausgewählt haben, welche Redis, MongoDB und Cassandra sind. Aber auch HBase und CouchDB, welche von [GC15] und [QS16] genannt wurden, schneiden laut DB-Engine gut ab. Hive und Riak scheiden aus, da sie weder in [QS16] genannt werden und laut DB-Engine nicht auf einem besseren Platz als die Konkurrenz, der jeweiligen Kategorie, sind. Couchbase hat einen besseren Ranglistenplatz als CouchDB bei DB-Engine erhalten, wird jedoch im Gegensatz zu CouchDB nicht in [QS16] aufgeführt. Da DB-Engine einen historischen Verlauf der Popularität besitzt [SI16c], erkennt man, dass der Unterschied zwischen Couchbase und CouchDB nicht groß ist. Aus diesem Grund scheidet die in [QS16] nicht genannte Couchbase aus. Daraus folgt, dass Redis, MongoDB, Cassandra, HBase und CouchDB evaluiert werden. Da jedoch keine graphenorientierte Datenbank dabei ist, wird die bestplatzierte Datenbank aus der DB-Engine Rangliste genommen [SI16d], welche Neo4j ist, aber durch die kommerzielle Lizenz ausscheidet. Die nächste reine graphenorientierte Datenbank ist Titan, welche zur Evaluation genutzt wird.

Bei den relationalen Datenbanken wird für die Evaluation PostgreSQL verwendet, da diese auf der Rangliste [SI16e] ganz oben steht, abgesehen von kommerziellen Varianten von anderen Datenbanken.

Datenbank	[GC15]	[QS16]	Kategorie	DB-Engines Rang [SI16b]
Apache Cassandra	+	+	Spaltenorientierte DB	1
Apache CouchDB	+	+	Dokumentenorientierte DB	4
Apache HBase	+	+	Spaltenorientierte DB	2
Apache Hive	+		Relationale DB	11
Couchbase	+		Dokumentenorientierte DB	2
MongoDB	+	+	Dokumentenorientierte DB	1
Redis	+	+	Key-Value-DB	1
Riak	+		Key-Value-DB	4

Tabelle 4.1.: Datenbanken im Vergleich

4.1.2. Dateisysteme

Es gibt verschiedene verteilte Dateisysteme, deshalb wurde eine Google-Suche (siehe Anhang A.2.1) durchgeführt. Unter den ersten zehn Resultaten ist eine Übersichtsseite mit Dateisystemen und Datenbanken [Mos16], woraus einige Dateisysteme beispielhaft notiert wurden, da nicht alle existierenden Dateisysteme im Folgenden aufgelistet werden können, da die Menge der Dateisysteme zu groß ist und ständig neue Dateisysteme hinzukommen.

- Amazon Elastic File System [AWS16b]
- Apache HDFS [ASFH16g]
- BeeGFS [TFB16a]
- Ceph Filesystem [RH16]
- Disco DDFS [NCD14]
- GlusterFS [RHG16f]
- Google GFS [GOO03]

Zur Evaluation werden Dateisysteme benötigt, welche zur Verfügung stehen und genutzt werden können, aus diesem Grund fällt beispielsweise Google GFS raus. Die Speicherung der Daten soll nicht durch ein Online Dateisystem übernommen werden, daher fällt beispielsweise Amazon Elastic File System raus.

Evaluiert wird Apache HDFS/Hadoop, da Apache HBase evaluiert wird und dessen Architektur baut auf Hadoop und damit auf HDFS auf. Dadurch können beide Komponenten separat evaluiert werden und Zusatzfunktionen durch Apache HBase dargestellt werden. Weitere Dateisysteme sind BeeGFS und GlusterFS, da diese von den Betreuern empfohlen wurden.

4.1.3. Existierende Archivierungssysteme

Ein breites Spektrum an existierenden Archivierungssysteme existiert auf dem Markt. Dabei haben diese unterschiedliche Einsatzzwecke und sind damit spezialisiert auf die Verwaltung von bestimmten Dokumenten. Beispielsweise ist *MailStore Server* spezialisiert auf die Archivierung von E-Mails [MSS16], da E-Mails auch rechtssicher archiviert werden müssen [MSS15] oder Archivierungssysteme für das Gesundheitswesen [BBS16].

Um weitere Archivierungslösungen zu finden, wurde eine Google-Suche (siehe Anhang A.2.2) durchgeführt, welche einige Resultate geliefert hat. Ein Ausschnitt der Resultate, welche Geschäftsunterlagen in einem Langzeitarchive aufbewahren können, sind in der folgenden Liste dargestellt.

- CIB doXima [CIB16]
- Doxis4 [SHI16]
- EASY ARCHIVE [ES16a]
- IBM Content Manger Version 8 (CM8) [IBM16]
- OpenText Livelink ECM [OTC16]
- scanview [AIS16]
- TMS Archiv [TMS16]

Da die Dokumentationen einiger Archivierungssysteme nicht öffentlich zugänglich sind, weil diese evtl. nur kommerziell angeboten werden, können diese nicht evaluiert werden. Damit fallen CIB doXima, OpenText Livelink ECM, scanview und TMS Archiv raus.

Für Qualitätsdaten gibt es weitere Archivierungslösungen, wobei diese Lösungen nicht ausschließlich für die Archivierung genutzt werden können, sondern auch für die Erfassung und Visualisierung. Eine Google-Suche (siehe Anhang A.2.3) ergab mehrere Treffer, welche unter anderem im Folgenden aufgelistet sind:

- apromaceMES [ADS16]
- ARBURG Leitrechnersystem (ALS) [ARG16]
- HYDRA-FEP [MMG16]
- MEVInet-Q [IMG16]
- Q-DAS Archivierungssystem [QDG16]

Bei diesen Lösungen werden alle Prozesse der Qualitätsdaten, von der Erzeugung bis zur Archivierung, abgedeckt. Es wird aber nur die Archivierung benötigt, wodurch die Lösungen nicht genutzt werden können, da die Prozesse fortlaufend sind und daher einzelne Prozesse, wie die Archivierung, nicht genutzt werden können. Des Weiteren konnten keine detaillierten Dokumentationen gefunden werden, da diese wahrscheinlich nicht öffentlich zugänglich sind.

Für die Evaluation verbleiben somit die folgenden drei existierenden Archivierungssysteme.

Kategorie	Name
Dokumentenorientierte DB	Apache CouchDB
Dokumentenorientierte DB	MongoDB
Spaltenorientierte DB	Apache Cassandra
Spaltenorientierte DB	Apache HBase
Key-Value-DB	Redis
Graphenorientierte DB	Titan
Relationale DB	PostgreSQL
Dateisystem	Apache Hadoop Framework
Dateisystem	BeeGFS
Dateisystem	GlusterFS
Exist. Archivierungssystem	Doxis4
Exist. Archivierungssystem	EASY ARCHIVE
Exist. Archivierungssystem	IBM CM8

Tabelle 4.2.: Datenbanken, Dateisysteme und existierende Archivierungssysteme, welche evaluiert werden

- Doxis4 [SHI16]
- EASY ARCHIVE [ES16a]
- IBM Content Manger Version 8 (CM8) [IBM16]

4.1.4. Evaluationsobjekte

In Abschnitte 4.1.1 und 4.1.2 wurden die Datenbanken und Dateisysteme ausgewählt, welche evaluiert werden sollen. Anschließend wurden die existierenden Archivierungssysteme in Abschnitt 4.1.3 ausgewählt.

In Tabelle 4.2 werden diese Evaluationsobjekte mit ihrer jeweiliger Kategorie zusammen aufgelistet. Nach dieser Kategorisierung und Reihenfolge werden die Objekte evaluiert.

4.2. Dokumentenorientierte Datenbanken

In diesem Abschnitt werden die dokumentenorientierten Datenbanken evaluiert. Angefangen mit Apache CouchDB in Abschnitt 4.2.1 und anschließend MongoDB in Abschnitt 4.2.2.

4.2.1. Apache CouchDB

„Cluster of unreliable commodity hardware Data Base“ (kurz: CouchDB) [ASFCO16e] ist eine dokumentenorientierte Datenbank und eine freie Software von Apache. Sie kann zum Speichern von JSON-Dokumenten genutzt werden [ASFCO16e]. Apache CouchDB ist beim Dreieck des CAP-Theorems auf der Kante Verfügbarkeit-Ausfalltoleranz [ALN10a] anzuordnen und besitzt nach PACELC die Einordnung PA/EL [Aba11].

Apache CouchDB wird im Folgenden mit Hilfe der Kriterien aus Abschnitt 3.7 evaluiert.

- Jeder Benutzer ist bei CouchDB standardmäßig ein Admin, jedoch können neue Benutzer mit Nutzername und Passwort hinzugefügt werden [ASFCO16b]. Kriterium 1 (*Authentifizierung des Nutzers*) wird damit erfüllt.
- Die Benutzer können Berechtigungen, beispielsweise lesen und schreiben, für bestimmte Datenbankentabellen erhalten [ASFCO16c]. Kriterium 2 (*Autorisierung des Nutzers*) wird damit erfüllt.
- CouchDB bietet die Möglichkeit an einen Anhang an ein JSON-Dokument hinzuzufügen. Dieser Anhang ist nicht limitiert auf ein Dateiformat, wodurch alle beliebigen Dateiformate abgespeichert werden können [ASFCO16a]. Kriterium 3 (*Speicherung von Dokumenten*) wird damit erfüllt.
- Um Dateien und Dokumente als Anhang speichern zu können, muss keine Umwandlung stattfinden. Kriterium 4 (*Migration ohne Datenverlust*) wird damit erfüllt.
- Mit dem Anhang können beliebige Dateigrößen abgespeichert werden [ASFCO16a]. Kriterium 5 (*Datenmodell für kleine Daten*) und Kriterium 6 (*Datenmodell für große Daten*) werden damit erfüllt.
- CouchDB kann individuell skaliert werden [ALN10b; ALN10c]. Kriterium 7 (*Skalierbarkeit*) wird damit erfüllt.
- Daten können auf unterschiedliche Knoten verteilt gespeichert werden [ALN10c]. Kriterium 8 (*Verteilte Replizierung*) wird damit erfüllt.
- MapReduce Aufträge können erstellt werden, womit Views, mit gleicher Funktion wie SQL-Views, erstellt werden können. Jedoch ist der MapReduce von CouchDB nicht für eine Volltextsuche geeignet, sondern für JSON-Dokumente. Des Weiteren besteht keine Möglichkeit einen MapReduce Auftrag über die Anhänge von Dokumenten zu erstellen [ALN10d]. Kriterium 9 (*Verteilter Datenauswertungsmechanismus*) wird damit nicht erfüllt.
- CouchDB besitzt Filterfunktionen mit denen es möglich ist, Aktionen von Benutzern auf Datenbanktabellen zu erlauben oder zu verweigern. Diese Filterfunktionen müssen in JavaScript geschrieben werden, wodurch eigene Konfigurationen beliebig eingestellt werden können, so können beispielsweise allen Benutzern das Lesen erlaubt werden,

jedoch nur je einem Nutzer das Schreiben und Löschen der Daten. Diese Funktionalität bietet den Schutz gegen Datenmanipulation auf der Ebene System. Werden diese Filterfunktionen mit der Autorisierung verbunden, so wird ein höherer Schutz gegen Datenmanipulation erreicht [ASFCO16c; ASFCO16d]. Kriterium 10 (*Unveränderbarkeit*) wird damit erfüllt.

- Wie bereits erwähnt ist CouchDB eine dokumentenorientierte Datenbank, welche JSON-Dokumente abspeichern kann. JSON-Dokumente enthalten hierarchische Daten und die Daten werden in CouchDB schemafrei abspeichert [ALN10e]. Kriterium 11 (*Hierarchische Datenspeicherung*) wird damit erfüllt.
- Eine Abfragesprache für JSON-Dokumente ermöglicht bestimmte Resultate durch Filtereinstellungen zu erhalten [ALN10f]. Kriterium 12 (*Abfragesprache für hierarchische Daten*) wird damit erfüllt.

4.2.2. MongoDB

MongoDB [Mon16e] ist eine dokumentenorientierte Open Source NoSQL-Datenbank. Mit dieser können Dokumente im Datenformat JSON-Format abgespeichert, verwaltet und abgefragt werden. MongoDB besitzt eine ausführliche Dokumentation und verschiedene Programme zur Visualisierung der vorhandenen Daten [Mon16e]. MongoDB ist beim Dreieck des CAP-Theorems auf der Kante Konsistenz-Ausfalltoleranz [AB13] anzuordnen und besitzt nach PACELC die Einordnung PA/EC [Aba12].

MongoDB wird im Folgenden mit Hilfe der Kriterien aus Abschnitt 3.7 evaluiert.

- MongoDB kann mit einer Benutzeranmeldung ausgestattet werden, wobei der Nutzer Anmeldename und Passwort benötigt. Außerdem können andere Mechanismen, wie beispielsweise Kerberos, verwendet werden [Mon16a]. Kriterium 1 (*Authentifizierung des Nutzers*) wird damit erfüllt.
- Die Benutzerverwaltung, welche mit der Benutzeranmeldung verbunden ist, kann um Rechte für Nutzer ergänzt werden, um den Zugriff auf Operationen und Ressourcen zu erlauben oder zu verweigern [Mon16g]. Kriterium 2 (*Autorisierung des Nutzers*) wird damit erfüllt.
- Dateien und Dokumente können in Bytes umgewandelt werden, wodurch die Bytes in ein JSON- oder Binary JSON (BSON)-Dokument eingefügt werden können [Mon16d]. Dadurch können semi- und unstrukturierte Daten gespeichert werden. Kriterium 3 (*Speicherung von Dokumenten*) wird damit erfüllt.
- Mit der Umwandlung der zu archivierenden Daten in ein binäres Datenformat gehen keine Daten verloren. Kriterium 4 (*Migration ohne Datenverlust*) wird damit erfüllt.

- Die Speicherung mit JSON- und BSON-Formaten ermöglicht, dass kleine Daten effizient gespeichert werden, da das Limit von BSON-Dokumenten $16\,\mathrm{MB}$ ist [Mon16c]. Kriterium 5 (*Datenmodell für kleine Daten*) wird damit erfüllt.
- Mit dem BSON Format können Daten bis zu $16\,\mathrm{MB}$ gespeichert werden. Für größere Daten kann Grid File System (GridFS) verwendet werden, welches die Daten in gleich große Stücke zerlegt, um die Daten effizient speichern zu können [Mon16c]. Kriterium 6 (*Datenmodell für große Daten*) wird damit erfüllt.
- MongoDB lässt dich individuell an die Bedürfnisse des Kunden horizontal skalieren [Mon16e]. Kriterium 7 (*Skalierbarkeit*) wird damit erfüllt.
- Replikationen auf verteilten Servern ermöglicht, dass der Datenbestand mehrfach existiert [Mon16e]. Kriterium 8 (*Verteilte Replizierung*) wird damit erfüllt.
- MongoDB ermöglicht über bestimmte Daten einen MapReduce Auftrag auszuführen. Kriterium 9 (*Verteilter Datenauswertungsmechanismus*) wird damit erfüllt.
- Der Schutz gegen Datenmanipulation findet auf der Ebene Organisation statt, dabei hilft eine umfangreiche Autorisierung, wodurch jeder Benutzer nur die Berechtigungen erhalten kann, welche er für seine Aktionen benötigt. Dabei können unterschiedliche Privilegien, wie beispielsweise *insert*, *update* und *delete* für Datensätze, aber auch *dropDatabase*, vergeben werden [Mon16b]. Diese Vielzahl an Privilegien ermöglicht, dass der archivierte Datenbestand nicht modifiziert oder gelöscht werden kann, wobei der Administrator von MongoDB nicht betrachtet wird. Kriterium 10 (*Unveränderbarkeit*) wird damit erfüllt.
- Mit einem JSON-Format lassen sich hierarchisch angeordnete Daten darstellen. Dieses JSON-Dokument kann direkt abgespeichert werden. Außerdem hat ein JSON-Dokument kein spezielles Datenschema in MongoDB, wodurch Flexibilität in der Datenstruktur entsteht [Mon16e]. Kriterium 11 (*Hierarchische Datenspeicherung*) wird damit erfüllt.
- MongoDB stellt eine eigene Abfragesprache zur Verfügung, mit der Abfragen auf JSON-Dokumenten ausgeführt werden können. Dadurch können die hierarchischen Daten gefiltert werden [Mon16f]. Kriterium 12 (*Abfragesprache für hierarchische Daten*) wird damit erfüllt.

4.3. Spaltenorientierte Datenbanken

In diesem Abschnitt werden die spaltenorientierten Datenbanken evaluiert. Angefangen mit Apache Cassandra in Abschnitt 4.3.1 und anschließend Apache HBase in Abschnitt 4.3.2.

4.3.1. Apache Cassandra

Apache Cassandra [ASFC15] ist eine spaltenorientierte NoSQL-Datenbank. Mit Cassandra können große Datenmengen verteilt verwaltet werden [DSC16a]. Apache Cassandra ist beim Dreieck des CAP-Theorems auf der Kante Verfügbarkeit-Ausfalltoleranz [AB13; Aba12] anzuordnen und besitzt nach PACELC die Einordnung PA/EL [Aba12].

Apache Cassandra wird im Folgenden mit Hilfe der Kriterien aus Abschnitt 3.7 evaluiert.

- Eine interne Passwort Authentifizierung ermöglicht, dass Cassandra nur von berechtigten Personen genutzt werden kann [DSC16b]. Kriterium 1 (*Authentifizierung des Nutzers*) wird damit erfüllt.
- Authentifizierte Benutzer können Berechtigungen für Befehle, beispielsweise CREATE, DROP und SELECT, erhalten [ASFC16a; DSC16b]. Kriterium 2 (*Autorisierung des Nutzers*) wird damit erfüllt.
- Dateien und Dokumente können in Bytes umgewandelt werden [DSC16c]. Kriterium 3 (*Speicherung von Dokumenten*) wird damit erfüllt.
- Die Umwandlung von Daten in Bytes bewirkt keinen Datenverlust. Kriterium 4 (*Migration ohne Datenverlust*) wird damit erfüllt.
- Kleine Dokumente, kleiner als $1\,\mathrm{MB}$, können in Bytes umgewandelt und als BLOB abgespeichert werden, ohne dass die Performance der Datenbank schlechter wird [DSC16c]. Kriterium 5 (*Datenmodell für kleine Daten*) wird damit erfüllt.
- Cassandra kann theoretisch einen BLOB mit $2\,\mathrm{GB}$ Speicher verwalten, jedoch ist ein BLOB bei Daten, größer als $1\,\mathrm{MB}$, nicht mehr geeignet, da ein BLOB bei kleinen Daten effizienter ist [DSC16c]. Kriterium 6 (*Datenmodell für große Daten*) wird damit nicht erfüllt.
- Cassandra kann durch Hinzufügen von neuen Knoten die Leistung und den Speicherplatz der Datenbank erhöhen [DSC16a], womit eine horizontale Skalierbarkeit möglich ist. Kriterium 7 (*Skalierbarkeit*) wird damit erfüllt.
- Daten können abhängig von Replikationsfaktor auf unterschiedlichen Knoten verteilt gespeichert werden [DSC16a]. Kriterium 8 (*Verteilte Replizierung*) wird damit erfüllt.
- Apache Cassandra besitzt keinen eigenen Mechanismus um die Daten parallel zu verarbeiten. Es kann aber Apache Spark genutzt werden um die Daten parallel und effizient auszuwerten [ASFS16]. Jedoch sind Erweiterungen von Datenbanken zur Erfüllung der Kriterien ausgeschlossen. Kriterium 9 (*Verteilter Datenauswertungsmechanismus*) wird damit nicht erfüllt.

- Einen unveränderbaren Datenbestand kann auf der Ebene Organisation durch die Berechtigungsverwaltung ermöglicht werden. Dabei erhalten die Benutzer, welche die Daten abspeichern und löschen sollen, die Berechtigung *MODIFY*, wodurch die Ausführung der Befehle *INSERT, UPDATE, DELETE* und *TRUNCATE* erlaubt werden [ASFC16a]. Die beiden Benutzer können mehr Operationen ausführen als für sie notwendig sind, jedoch wird von validierten Benutzern ausgegangen und damit gehen mehr erlaubte Operationen in Ordnung. Die anderen Benutzer erhalten diese Berechtigung nicht, wodurch keine Datenmanipulation auf dieser Ebene stattfinden kann. Kriterium 10 (*Unveränderbarkeit*) wird damit erfüllt.
- Seit Version 2.2 kann Cassandra JSON-Dokumente speichern und bei einer Anfrage zurückliefern. Dabei benötigen die Dokumente kein Schema [ASFC16b]. Kriterium 11 (*Hierarchische Datenspeicherung*) wird damit erfüllt.
- Cassandra besitzt eine eigene Abfragesprache, die „Cassandra Query Language“ (kurz: CQL), welche der SQL-Syntax und dem SQL-Funktionsumfang ähnelt. Diese Abfragesprache unterstützt das Erstellen und Abfragen von JSON-Dokumenten in der Datenbank [ASFC16b]. Kriterium 12 (*Abfragesprache für hierarchische Daten*) wird damit erfüllt.

4.3.2. Apache HBase

Die „Apache Hadoop Database“ (kurz: Apache HBase) [ASFB16] ist eine Erweiterung des Hadoop Projekts von Abschnitt 4.7.1. Diese Erweiterung bildet die Hadoop Datenbank, welche eine spaltenorientierte NoSQL-Datenbank ist. Das zugrunde liegende Hadoop Projekt ermöglicht, dass sämtliche Hadoop Funktionalitäten unterstützt werden. Somit kann zur Speicherung der Daten das Dateisystem Hadoop Distributed File System (HDFS) genutzt werden. Des Weiteren können durch die geerbten Funktionalitäten große Datenmengen verteilt und skalierbar gespeichert werden [ASFB16]. Apache HBase ist beim Dreieck des CAP-Theorems auf der Kante Konsistenz-Ausfalltoleranz [AHT16e] anzuordnen und besitzt nach PACELC die Einordnung PC/EC [Aba12].

Apache HBase wird im Folgenden mit Hilfe der Kriterien aus Abschnitt 3.7 evaluiert.

- Wie bereits bei Hadoop kann HBase eine Authentifizierung mit Kerberos durchführen [AHT16f]. Des Weiteren kann sich ein Benutzer mit SSH anmelden [AHT16b]. Kriterium 1 (*Authentifizierung des Nutzers*) wird damit erfüllt.
- Eine Zugangsbeschränkung ermöglicht, dass Nutzer Berechtigungen, wie beispielsweise lesen und schreiben, auf zugewiesene Tabellen erhalten können [AHT16a]. Kriterium 2 (*Autorisierung des Nutzers*) wird damit erfüllt.
- Die Speicherung der Daten wird durch unterschiedliche Datenformate ermöglicht. So eignet sich für Daten, kleiner als $10\,\mathrm{MB}$, ein binäres Datenformat BLOB oder Medium-sized Object (MOB). Zur Abspeicherung von Dateien, größer als $10\,\mathrm{MB}$, nutzt man das

HDFS, wobei eine Referenz auf die Datei in HBase hinterlegt wird [AHT16g]. Kriterium 3 (*Speicherung von Dokumenten*) wird damit erfüllt.

- Bei der Umwandlung in ein binäres Datenformat gehen keine Daten verloren. Kriterium 4 (*Migration ohne Datenverlust*) wird damit erfüllt.
- Die unterschiedlichen binären Datenformate erlauben eine effiziente Datenspeicherung. Die Speicherung als BLOB ist bis $100\,\mathrm{kB}$ zu empfehlen und bis $10\,\mathrm{MB}$ das Datenformat MOB. Dateien größer als $10\,\mathrm{MB}$ können direkt auf das HDFS mit entsprechender Referenz in HBase gespeichert werden [AHT16g; AHT16h]. Kriterium 5 (*Datenmodell für kleine Daten*) und Kriterium 6 (*Datenmodell für große Daten*) werden damit erfüllt.
- Die Rechenleistung und der Speicher kann durch Hinzufügen von Knoten erhöht werden [ASFB16]. Kriterium 7 (*Skalierbarkeit*) wird damit erfüllt.
- Apache HBase kann Daten verteilt auf verschiedenen Knoten speichern. Die Daten werden je nach Replikationsfaktor mehrfach abgespeichert [AHT16d]. Kriterium 8 (*Verteilte Replizierung*) wird damit erfüllt.
- Apache HBase besitzt ein Software Framework MapReduce mit dem große Datenmengen parallel verarbeitet werden können [AHT16c]. Kriterium 9 (*Verteilter Datenauswertungsmechanismus*) wird damit erfüllt.
- Einen unveränderbaren Datenbestand kann auf der Ebene Organisation durch die Berechtigungsverwaltung ermöglicht werden. Dabei erhalten die Benutzer, welche die Daten abspeichern und löschen sollen die Berechtigung *Write*, wodurch Daten geschrieben und gelöscht werden können [AHT16a]. Die beiden Benutzer können mehr Operationen ausführen als für sie notwendig sind, jedoch wird von validierten Benutzern ausgegangen und damit gehen mehr erlaubte Operationen in Ordnung. Die anderen Benutzer erhalten diese Berechtigung nicht, wodurch keine Datenmanipulation auf dieser Ebene stattfinden kann. Kriterium 10 (*Unveränderbarkeit*) wird damit erfüllt.
- Die Speicherung von hierarchischen Daten wird durch den Datentyp String oder BLOB ermöglicht. Jedoch entspricht diese Umwandlung nicht den Anforderungen. Kriterium 11 (*Hierarchische Datenspeicherung*) wird damit nicht erfüllt.
- Eine Abfragesprache für hierarchische Daten existiert nicht. Kriterium 12 (*Abfragesprache für hierarchische Daten*) wird damit nicht erfüllt.

4.4. Key-Value Datenbanken

In Abschnitt 4.4.1 wird Redis eine Key-Value Datenbank evaluiert.

4.4.1. Redis

Der „remote dictionary server“ (kurz: Redis) [San16c] ist eine Open Source In-Memory Datenbank und gehört zu der Kategorie Key-Value Datenbanken. Sie speichert alle Daten im Hauptspeicher und persistent auf der Festplatte [San16c]. Redis ist beim Dreieck des CAP-Theorems auf der Kante Konsistenz-Ausfalltoleranz [HHGJ11] anzuordnen. Bei der Einordnung in PACELC wird die Annahme getroffen, dass Redis in die Kategorie PA/EC einzuordnen ist. Diese Annahme beruht darauf, dass bei einer Netzpartitionierung zwei Master-Knoten entstehen können, wodurch das System verfügbar ist, aber die Konsistenz verloren geht [Kin13].

Redis wird im Folgenden mit Hilfe der Kriterien aus Abschnitt 3.7 evaluiert.

- Der Schutz vor unberechtigten Servern oder Computern wird durch einen Netzwerkmechanismus ermöglicht, wobei von allen Clients, welche Redis nutzen wollen, die IP-Adresse notiert werden muss. Nur diese Clients haben anschließend Zugriff auf Redis. Die Sicherheit kann durch ein Passwort verstärkt werden, wobei ein Client das Passwort an Redis mitsenden muss [San16d]. Diese zwei Mechanismen bieten einen Schutz gegen Angreifer von einem anderen Rechner mit einer anderen IP-Adresse, jedoch können trotzdem Benutzer desselben Rechners die Dienste von Redis in Anspruch nehmen. Kriterium 1 (*Authentifizierung des Nutzers*) und Kriterium 2 (*Autorisierung des Nutzers*) werden damit nicht erfüllt.
- Dokumente und Dateien können in einem String abgespeichert werden, wobei die Daten davor in einen BLOB umgewandelt werden müssen [San16a]. Kriterium 3 (*Speicherung von Dokumenten*) wird damit erfüllt.
- Bei der Umwandlung von Dokumenten und Dateien in einen binären Datentyp gehen keine Daten verloren. Kriterium 4 (*Migration ohne Datenverlust*) wird damit erfüllt.
- Daten, bis zu $512\,\mathrm{MB}$, können als String in der In-Memory Datenbank abgespeichert werden [San16a]. Dabei ist zu berücksichtigen, dass bei vielen Daten viel Hauptspeicher benötigt wird, wodurch neue Knoten hinzugefügt werden müssen und hohe Kosten für den Hauptspeicher anfallen können. Aus dem Mengengerüst (siehe Abschnitt 3.1) geht hervor, dass diese Umsetzung nicht mehr wirtschaftlich ist, da zu viel Hauptspeicher benötigt wird. Kriterium 5 (*Datenmodell für kleine Daten*) wird damit nicht erfüllt.
- Zu dem Problem mit dem Hauptspeicher kommt dazu, dass nur Daten bis $512\,\mathrm{MB}$ abgespeichert werden können, nicht wie verlangt $10\,\mathrm{GB}$ [San16a]. Kriterium 6 (*Datenmodell für große Daten*) wird damit nicht erfüllt.
- Redis kann horizontal skaliert werden. Beim Hinzufügen von neuen Knoten, ist der Hauptspeicher und das Betriebssystem von hoher Bedeutung. Das liegt daran, dass die Daten im Hauptspeicher gespeichert werden und je nach 32- oder 64-Bit Betriebssystem unterschiedlich viel Hauptspeicher angesteuert werden kann [San16b]. Kriterium 7 (*Skalierbarkeit*) wird damit erfüllt.

- Die gespeicherten Daten können auf unterschiedlichen Servern abgespeichert werden, wobei je nach Master Knoten beliebig viele Slave Knoten eingerichtet werden können. Das liegt daran, dass bei jedem Slave Knoten definiert werden muss welcher Master Knoten repliziert werden muss. Dadurch kann ein beliebig hoher Replikationsfaktor erstellt werden. Nachteil ist, dass durch Kriterium 7 (*Skalierbarkeit*) neue Master Knoten hinzukommen können und für jeden neuen Master Knoten müssen neue Slave Knoten definiert werden um eine Replikation zu ermöglichen [San16e]. Kriterium 8 (*Verteilte Replizierung*) wird damit erfüllt.
- Redis besitzt keinen eigenen Mechanismus um die Daten parallel zu verarbeiten. Es kann r^3 genutzt werden, um die Daten parallel und effizient auszuwerten [Hey12]. Jedoch sind Erweiterungen von Datenbanken zur Erfüllung der Kriterien ausgeschlossen. Kriterium 9 (*Verteilter Datenauswertungsmechanismus*) wird damit nicht erfüllt.
- Der Schutz gegen Datenmanipulation kann durch die Ebene Organisation nicht sichergestellt werden, da beide Kriterien „Authentifizierung“ und „Autorisierung“ nicht erfüllt sind. Des Weiteren wurden keine Informationen zur Ebene System gefunden. Eine Aussage zur Erfüllung von Kriterium 10 (*Unveränderbarkeit*) kann damit nicht getroffen werden.
- Redis besitzt keinen hierarchischen Datentyp oder ähnliche Möglichkeiten hierarchische Daten abzuspeichern. Einzige Möglichkeit ist es die Daten in ein String umzuwandeln und abzuspeichern, wodurch eine Umwandlung stattfindet [San16a]. Kriterium 11 (*Hierarchische Datenspeicherung*) wird damit nicht erfüllt.
- Es können weder hierarchische Daten geschrieben noch gelesen werden [San16a]. Des Weiteren können nur auf Schlüssel und Indizes Operationen ausgeführt werden, jedoch nicht auf den Wert selber. Kriterium 12 (*Abfragesprache für hierarchische Daten*) wird damit nicht erfüllt.

4.5. Graphenorientierte Datenbanken

In Abschnitt 4.5.1 wird Titan eine graphenorientierte Datenbank evaluiert.

4.5.1. Titan

Titan [DST16h] ist eine graphenorientierte Open Source NoSQL-Datenbank. Sie ist eine skalierbare und verteilte Datenbank mit der Möglichkeit Graphen zu verwalten und abzufragen. Dabei können die Speichertechnologien von Apache Cassandra, Apache HBase und Oracle BerkeleyDB genutzt werden, wobei sich die jeweiligen Eigenschaften unterscheiden [DST16h]. Die unterschiedlichen Speichertechnologien bewirken, dass die Einordnungen in CAP und PACELC verschieden sind [DST16a].

Titan wird im Folgenden mit Hilfe der Kriterien aus Abschnitt 3.7 evaluiert.

- Eine Standardauthentifizierung mit dem Gremlin Server ist möglich [DST16g]. Damit können Benutzer mit Name und Passwort authentifiziert werden [ASFT15b]. Kriterium 1 (*Authentifizierung des Nutzers*) wird damit erfüllt.
- Informationen zu einer Autorisierung von Benutzern konnten nicht gefunden werden. Eine Aussage zur Erfüllung von Kriterium 2 (*Autorisierung des Nutzers*) kann damit nicht getroffen werden.
- Eine Abspeicherung von Dokumenten ist mit einem String möglich [DST16e]. Kriterium 3 (*Speicherung von Dokumenten*) wird damit erfüllt.
- Eine Umwandlung von unstrukturierten Daten in einen String kann zu Datenverlusten führen. Kriterium 4 (*Migration ohne Datenverlust*) wird damit nicht erfüllt.
- Es wurden keine Informationen gefunden, welche Eigenschaften die Datenbank Titan bei unterschiedlichen Dateigrößen besitzt. Eine Aussage zur Erfüllung von Kriterium 5 (*Datenmodell für kleine Daten*) und Kriterium 6 (*Datenmodell für große Daten*) kann damit nicht getroffen werden.
- Eine horizontale Skalierung kann mit den Speichertechnologien Apache HBase und Apache Cassandra erstellt werden [DST16b; DST16c]. Kriterium 7 (*Skalierbarkeit*) wird damit erfüllt.
- Eine verteilte Datenspeicherung und ein frei wählbarer Replikationsfaktor existiert [DST16b; DST16c]. Kriterium 8 (*Verteilte Replizierung*) wird damit erfüllt.
- Der Vorteil von Titan-Hadoop ist, dass MapReduce genutzt werden kann [DST16d]. Des Weiteren wird durch Gremlin die Nutzung von MapReduce angeboten [ASFT15a]. Kriterium 9 (*Verteilter Datenauswertungsmechanismus*) wird damit erfüllt.
- Der Schutz gegen Datenmanipulation kann durch die Ebene Organisation nicht sichergestellt werden, da keine Informationen zur „Autorisierung“ vorliegen. Des Weiteren wurden keine Informationen zur Ebene System gefunden. Eine Aussage zur Erfüllung von Kriterium 10 (*Unveränderbarkeit*) kann damit nicht getroffen werden.
- Die Datenspeicherung von hierarchischen Daten kann mit Kanten und Knoten funktionieren, jedoch müssen die Daten erst zerlegt werden und dann abgespeichert werden. Kriterium 11 (*Hierarchische Datenspeicherung*) wird damit nicht erfüllt.
- Die Abfragesprache Gremlin ermöglicht, dass Daten geschrieben und abgefragt werden können [DST16f]. Hierarchisch strukturierte Daten ist eine Art der topologischen Sortierung von einem Graph. Daher könnte Gremlin als Abfragesprache genutzt werden. Kriterium 12 (*Abfragesprache für hierarchische Daten*) wird damit erfüllt.

4.6. Objektrelationale Datenbanken

In Abschnitt 4.6.1 wird PostgreSQL eine objektrelationale Datenbank evaluiert.

4.6.1. PostgreSQL

PostgreSQL [PGD16h] ist eine objektrelationale Datenbank, welche Open Source ist. Der SQL-Standard wird größtenteils unterstützt und durch neue Funktionalitäten erweitert [PGD16i]. Die Einordnung ins CAP-Theorem (siehe Abschnitt 2.4) ergibt Verfügbarkeit und Konsistenz [HHGJ11]. Eine Einordnung in PACELC (siehe Abschnitt 2.5) kann nicht gemacht werden, da PostgreSQL keine Netzwerkpartitionierung aufweist, aber PACELC für die Einordnung der ersten drei Buchstaben eine Netzwerkpartitionierung benötigt.

PostgreSQL wird im Folgenden mit Hilfe der Kriterien aus Abschnitt 3.7 evaluiert.

- Der Benutzer kann sich über eine Passwort Authentifizierung bei PostgreSQL anmelden. Eine Anmeldung mit Kerberos ist auch möglich. Des Weiteren kann für jede Datenbank ein eigenes Passwort vergeben werden [PGD16a]. Kriterium 1 (*Authentifizierung des Nutzers*) wird damit erfüllt.
- Benutzer können Berechtigungen für Befehle, beispielsweise SELECT, INSERT und DELETE, für unterschiedliche Tabellen erhalten [PGD16d]. Kriterium 2 (*Autorisierung des Nutzers*) wird damit erfüllt.
- Dokumente können als binär Daten mit dem Datentyp *bytea* abgespeichert werden [PGD16c]. Kriterium 3 (*Speicherung von Dokumenten*) wird damit erfüllt.
- Die Umwandlung der Dokumente erfolgt ohne Datenverlust. Kriterium 4 (*Migration ohne Datenverlust*) wird damit erfüllt.
- Der Datentyp *bytea* kann binär Daten bis zu einem GB enthalten, jedoch benötigen große Daten viel Speicher und somit sind große Dateien damit ineffizient zu speichern. Die Grenze von einem Megabyte ist weit von einem Gigabyte entfernt. Kriterium 5 (*Datenmodell für kleine Daten*) wird damit erfüllt.
- Für Daten, größer als $1\,\mathrm{MB}$, kann der Datentyp *bytea* bis zu einem GB genutzt werden, wobei große Daten viel Speicher benötigen und dadurch sind große Dateien ineffizient zu speichern. Deshalb sollten *Large Objects* verwendet werden, sodass die Performance nicht zu stark beeinträchtigt wird. Mit diesen *Large Objects* können Daten bis zu einer Größe von $4\,\mathrm{TB}$ verwaltet werden [PGD16g]. Kriterium 6 (*Datenmodell für große Daten*) wird damit erfüllt.

- Eine horizontale Skalierbarkeit wird von PostgreSQL nicht unterstützt. Stattdessen wird eine Erweiterung, beispielsweise *Citus Data*, benötigt, welche mehrere Server verbinden kann um eine horizontale Skalierbarkeit zu ermöglichen [Cit15]. Jedoch sind Erweiterungen von Datenbanken zur Erfüllung der Kriterien ausgeschlossen. Kriterium 7 (*Skalierbarkeit*) wird damit nicht erfüllt.
- PostgreSQL muss bei einer verteilten Datenspeicherung mit Replikationen erweitert werden, mit beispielsweise *Citus Data*, wodurch mehrere Knoten das Speichern der Daten übernehmen können und Replikationen auf den Knoten verteilt werden [Cit15]. Jedoch sind Erweiterungen von Datenbanken zur Erfüllung der Kriterien ausgeschlossen. Kriterium 8 (*Verteilte Replizierung*) wird damit nicht erfüllt.
- Ein verteilter Abfragemechanismus existiert bei PostgreSQL nicht. Mit der Erweiterung *Pgpool-II* kann eine Abfrage auf mehreren Servern gleichzeitig ausgeführt werden [PGD16b]. Jedoch sind Erweiterungen von Datenbanken zur Erfüllung der Kriterien ausgeschlossen. Kriterium 9 (*Verteilter Datenauswertungsmechanismus*) wird damit nicht erfüllt.
- Der Schutz gegen Datenmanipulation auf der Ebene Organisation kann durch die hohe Granularität der Rechte ermöglicht werden, wodurch nur bestimmte Benutzer die Rechte *INSERT* und *DELETE* erhalten können. Kriterium 10 (*Unveränderbarkeit*) wird damit erfüllt.
- PostgreSQL unterstützt die Speicherung von JSON-Dokumenten. Das wird durch einen eigenen Datentyp *json* möglich [PGD16e; PGD16f]. Kriterium 11 (*Hierarchische Datenspeicherung*) wird damit erfüllt.
- Es wird eine Abfragesprache angeboten mit der gespeicherte JSON-Dokumente ausgewertet werden können. So können Vergleiche und Aggregationen durchgeführt werden [PGD16e; PGD16f]. Kriterium 12 (*Abfragesprache für hierarchische Daten*) wird damit erfüllt.

4.7. Dateisysteme

In diesem Abschnitt werden die ausgewählten Dateisysteme (siehe Abschnitt 4.1.2) evaluiert. Angefangen mit Apache Hadoop Framework in Abschnitt 4.7.1 und anschließend BeeGFS in Abschnitt 4.7.2. Als letztes Dateisystem wird GlusterFS in Abschnitt 4.7.3 evaluiert.

4.7.1. Apache Hadoop Framework

Das Apache Hadoop Projekt [ASFH16g] ist keine Datenbank, sondern ein Framework und eine Open Source Bibliothek für verteilte Datenspeicherung und Verarbeitung. Die Leistung und der

Datenspeicher kann durch eine horizontale Skalierbarkeit reguliert werden [ASFH16g]. Hadoop besitzt ein eigenes Dateisystem, welches zur Datenspeicherung genutzt werden kann.

Hadoop besteht aus folgenden Modulen [ASFH16g]:

- „Hadoop Distributed File System“ (HDFS): Ein verteiltes Dateisystem mit einfacherer Skalierbarkeit, welches aus beliebig vielen Knoten bestehen kann.
- Hadoop MapReduce: Erlaubt die parallele Verarbeitung von großen Datenmengen.
- YARN (Yet Another Resource Negotiator): Ein Framework für Ressourcen- und Aufgabenmanagement, wobei diese von MapReduce entkoppelt wurden. Das ermöglicht neue Anwendungsgebiete. Daher wird es auch als MapReduce 2.0 bezeichnet.
- Hadoop Common: Eine Standard Bibliothek für weitere Module, welche nützliche Funktionen für den Umgang mit Hadoop enthält.

Zu den genannten Modulen kommen Erweiterungen mit unterschiedlichen Funktionalitäten dazu, wodurch neue Anwendungsgebiete abgedeckt werden können. Eine Erweiterung ist Apache HBase, welches in Abschnitt 4.3.2 evaluiert wurde [ASFH16g].

Die Speicherung der Dokumente bei Hadoop übernimmt das HDFS, welches die Dokumente mit einem Namenode und mehreren Datanodes verwaltet. Die Datanodes speichern die Dokumente, wobei die Dokumente in einstellbare Block Größen unterteilt werden. Der Namenode speichert die Referenzen auf die Dateien mit Verzeichnisnamen, Dateinamen und Anzahl der Replikate [ASFH16e]. Maximal können zwei Namenodes konfiguriert werden, wobei einer von beiden aktiv und der andere passiv ist. Dadurch kann ein Single Point of Failure vermieden werden [ASFH16c].

Apache Hadoop Framework wird im Folgenden mit Hilfe der Kriterien aus Abschnitt 3.7 evaluiert.

- Hadoop läuft standardmäßig im nicht sicheren Modus, wodurch keine Authentifizierung benötigt wird. Jedoch kann auf einen sicheren Modus umgestellt werden, wodurch jeder Nutzer und jeder Service sich mit Kerberos authentifizieren muss, um Hadoop Services nutzen zu können [ASFH16a]. Kriterium 1 (*Authentifizierung des Nutzers*) wird damit erfüllt.
- Ein Berechtigungsmodel ermöglicht, dass der Zugriff auf Daten und Ordner im HDFS nur von berechtigten Personen und Nutzergruppen gewährt wird [ASFH16d]. Kriterium 2 (*Autorisierung des Nutzers*) wird damit erfüllt.
- Wie bereits erwähnt besitzt Hadoop ein HDFS, welches ein Dateisystem ist [ASFH16e]. Kriterium 3 (*Speicherung von Dokumenten*) wird damit erfüllt.
- HDFS ist ein Dateisystem, wodurch Dokumente ohne Umwandlung abgespeichert werden können. Kriterium 4 (*Migration ohne Datenverlust*) wird damit erfüllt.

- Dateien werden beim Speichern in gleich große Blöcke geteilt, wobei die Blöcke abgespeichert werden [ASFH16b]. Die kleinstmögliche Blockgröße ist dabei $1\,\mathrm{MB}$, wodurch bei Dateien, kleiner als $1\,\mathrm{MB}$, nicht die Vorteile von Hadoop genutzt werden und damit die Performance reduziert wird. Dazu kommt, dass beim Speichern von Dateien, Verzeichnissen und Blöcken jeweils 150 Bytes an Speicherplatz beim Namenode benötigt wird. Das hat zur Folge, dass bei vielen kleinen Daten viel Speicherplatz benötigt wird. Beispielsweise wird bei zehn Million Dateien $3\,\mathrm{GB}$ Speicherplatz benötigt [Whi09a]. Dieses Speicherproblem kann durch ein Hadoop Archive gelöst werden, wodurch mehrere Dateien zu einem großen Archiv zusammengefasst werden und damit weniger Speicher benötigt wird [Whi09b]. Jedoch lassen sich die Daten einzeln oder als Archiv nicht effizient auswerten, da ein MapReduce Job alle Dateien im Archiv als einzelne Dateien wahrnimmt. Das hat zur Folge, dass jeder MapReduce Job nur kleine Dateien als Input erhält, wodurch viele MapReduce Jobs gestartet werden müssen. Beispielsweise müssen bei einer einzigen $1\,\mathrm{GB}$ Datei und $64\,\mathrm{MB}$ Block Größe 16 map Aufträge ausgeführt. Besitzt man nur $100\,\mathrm{kB}$ Dateien, so würde man 10.000 Dateien besitzen, wodurch 10.000 map Aufträge gestartet werden. Dabei kann jeder map Auftrag eine JVM starten, was zur Folge hat, dass die Performance von Hadoop schlechter wird [Whi09c]. Kriterium 5 (*Datenmodell für kleine Daten*) wird damit nicht erfüllt.
- Die richtige Blockgröße für unterschiedlich große Dateien zu finden, ist schwer. Betrachtet man Dateien, welche größer sind als die Blockgröße, so können diese effizient gespeichert werden, jedoch nicht unbedingt die Dateien, welche kleiner als die Blockgröße sind [Whi09a]. Da der Übergang der Dateigröße, kleiner und größer als $1\,\mathrm{MB}$, nicht vernachlässigt werden darf, ist eine effiziente Abspeicherung von großen Daten nicht möglich. Kriterium 6 (*Datenmodell für große Daten*) wird damit nicht erfüllt.
- Wie bereits erwähnt kann Hadoop horizontal skaliert werden, wobei mehrere tausend Knoten zu einem Gesamtsystem kombiniert werden können [ASFH16e]. Kriterium 7 (*Skalierbarkeit*) wird damit erfüllt.
- HDFS ist ein verteiltes Dateisystem [ASFH16e], wobei ein Replikationsfaktor die Anzahl der Replikate angibt. Dadurch können Replikate auf verschiedenen Knoten und somit Datanodes gespeichert werden. Der Replikationsfaktor kann beliebig eingestellt werden [ASFH16b]. Kriterium 8 (*Verteilte Replizierung*) wird damit erfüllt.
- Apache Hadoop besitzt ein Software Framework MapReduce mit dem große Datenmengen parallel verarbeitet werden können [ASFH16f]. Kriterium 9 (*Verteilter Datenauswertungsmechanismus*) wird damit erfüllt.
- Einen unveränderbaren Datenbestand kann auf der Ebene Organisation durch die Berechtigungsverwaltung ermöglicht werden. Dabei erhalten die Benutzer, welche die Daten abspeichern und löschen sollen die Berechtigung *Write*, wodurch Daten geschrieben und gelöscht werden können [ASFH16d]. Die beiden Benutzer können mehr Operationen ausführen als für sie notwendig sind, jedoch wird von validierten Benutzern ausgegangen und damit gehen mehr erlaubte Operationen in Ordnung. Die anderen Benutzer

erhalten diese Berechtigung nicht, wodurch keine Datenmanipulation auf dieser Ebene stattfinden kann. Kriterium 10 (*Unveränderbarkeit*) wird damit erfüllt.

- Durch das Dateisystem HDFS können alle Dateiformate abgespeichert werden, somit auch hierarchische Dateiformate, wie beispielsweise XML oder JSON [ASFH16e]. Diese Daten unterliegen keinem Schema, da alle Dateien separat abgespeichert werden. Kriterium 11 (*Hierarchische Datenspeicherung*) wird damit erfüllt.
- Hadoop fehlt eine Abfragesprache mit der es möglich wäre hierarchische Daten auszuwerten. Das existierende MapReduce Framework könnte die Daten parallel verarbeiten, jedoch ist MapReduce keine Abfragesprache [ASFH16f]. Kriterium 12 (*Abfragesprache für hierarchische Daten*) wird damit nicht erfüllt.

4.7.2. BeeGFS

BeeGFS [TFB16a] ist ein frei erhältliches Dateisystem, welches auf verteilte Dateiverwaltung setzt [TFB16a].

BeeGFS wird im Folgenden mit Hilfe der Kriterien aus Abschnitt 3.7 evaluiert.

- Eine sichere Verbindung zwischen Client und Server soll existieren [TP16], weitere Informationen dazu konnten nicht gefunden werden. Des Weiteren existieren zwei Benutzer, ein Administrator und ein Informationsbenutzer [TFB16c; TFB16d], wobei keine Informationen zu neuen Benutzer noch zu Privilegien gefunden werden konnten. Eine Aussage zur Erfüllung von Kriterium 1 (*Authentifizierung des Nutzers*) und Kriterium 2 (*Autorisierung des Nutzers*) kann damit nicht getroffen werden.
- BeeGFS ist ein Dateisystem, wodurch Dokumente und Dateien direkt abgespeichert werden können [TFB16e]. Kriterium 3 (*Speicherung von Dokumenten*) wird damit erfüllt.
- Es erfolgt keine Umwandlung, da Dokumente direkt abgespeichert werden können [TFB16e]. Kriterium 4 (*Migration ohne Datenverlust*) wird damit erfüllt.
- Daten, kleiner als $1\,\mathrm{MB}$, können abgespeichert werden. Jedoch können die häufigen kleinen Schreib- und Lesezugriffe die Leistung beeinträchtigen. Die Auswirkungen können durch bessere CPU und Festplatten verringert werden, wodurch aber höhere Kosten bei der Skalierung anfallen [TFB16e]. Diese negativen Auswirkungen, auf die Leistung des Dateisystems, dürfen nicht vernachlässigt werden. Kriterium 5 (*Datenmodell für kleine Daten*) wird damit nicht erfüllt.
- Daten, zwischen $1\,\mathrm{MB}$ und $10\,\mathrm{GB}$, können abgespeichert werden [TFB16e]. Kriterium 6 (*Datenmodell für große Daten*) wird damit erfüllt.
- Neue Knoten können dem Gesamtsystem hinzugefügt werden um den Speicherplatz und die Leistung zu erhöhen [TFB16f]. Kriterium 7 (*Skalierbarkeit*) wird damit erfüllt.

- Daten können auf zwei verteilten Knoten redundant gespeichert werden, wobei die Knoten dynamisch oder manuell zugewiesen werden können. Dabei gibt es ein Master Knoten und einen gespiegelten sekundär Knoten, welche eine Gruppe bilden. Fällt der Master Knoten aus, wird er durch den sekundären Knoten ersetzt. Der maximale Replikationsfaktor liegt bei zwei, da nur zwei Knoten einer Gruppe hinzugefügt werden können [TFB16b]. Kriterium 8 (*Verteilte Replizierung*) wird damit erfüllt.
- BeeGFS ist nur ein Dateisystem und bietet keine Abfragesprache an. Kriterium 9 (*Verteilter Datenauswertungsmechanismus*) wird damit nicht erfüllt.
- Die Unveränderbarkeit wird durch WORM Bänder ermöglicht, jedoch existieren keine Informationen zur Unveränderbarkeit durch die Ebene der Organisation oder des Systems. Eine Aussage zur Erfüllung von Kriterium 10 (*Unveränderbarkeit*) kann damit nicht getroffen werden.
- Wie bereits bei der Evaluation von Kriterium 3 (*Speicherung von Dokumenten*) angemerkt wurde, können alle Dateitypen abgespeichert werden, somit auch hierarchische Dateiformate [TFB16e]. Kriterium 11 (*Hierarchische Datenspeicherung*) wird damit erfüllt.
- BeeGFS ist nur ein Dateisystem und bietet keine Abfragesprache an. Kriterium 12 (*Abfragesprache für hierarchische Daten*) wird damit nicht erfüllt.

4.7.3. GlusterFS

GlusterFS [RHG16f] ist ein Open Source Dateisystem. Dieses lässt sich über verteilte Knoten skalieren um mehr Speicherplatz für die Datenhaltung bereitzustellen [RHG16f].

GlusterFS wird im Folgenden mit Hilfe der Kriterien aus Abschnitt 3.7 evaluiert.

- GlusterFS kann per SSL und IP-Tabelle, mit erlaubten und nicht erlaubten IP-Adressen, geschützt werden. Dadurch existiert nicht die Möglichkeit spezielle Benutzer zu registrieren und damit existiert kein Berechtigungsmanagement [RHG16b; RHG16e]. Kriterium 1 (*Authentifizierung des Nutzers*) und Kriterium 2 (*Autorisierung des Nutzers*) werden damit nicht erfüllt.
- GlusterFS ist ein Dateisystem, wodurch Dokumente und Dateien direkt abgespeichert werden können [RHG16d]. Kriterium 3 (*Speicherung von Dokumenten*) wird damit erfüllt.
- Es erfolgt keine Umwandlung, da Dokumente direkt abgespeichert werden können [RHG16d]. Kriterium 4 (*Migration ohne Datenverlust*) wird damit erfüllt.
- Das Dateisystem kann durch Features für den Umgang mit kleinen Daten optimiert werden, wobei die eigentliche Speicherung von Dateien, kleiner als $1\,\mathrm{MB}$, möglich ist [GHG16]. Kriterium 5 (*Datenmodell für kleine Daten*) wird damit erfüllt.

- Eine Datenspeicherung mit Dateigrößen bis zu 10 GB und größer wird unterstützt [RHG16b]. Kriterium 6 (*Datenmodell für große Daten*) wird damit erfüllt.
- Eine horizontale Skalierung kann mit Standard Hardware durchgeführt werden [RHG16a]. Kriterium 7 (*Skalierbarkeit*) wird damit erfüllt.
- Individuelle Konfigurationen erlauben es, dass ein frei wählbarer Replikationsfaktor definiert werden kann und dass GlusterFS verteilt auf unterschiedlichen Knoten laufen kann [RHG16a]. Kriterium 8 (*Verteilte Replizierung*) wird damit erfüllt.
- GlusterFS ist kompatibel mit Apache Hadoop, wodurch MapReduce genutzt werden kann [RHG16c]. Kriterium 9 (*Verteilter Datenauswertungsmechanismus*) wird damit erfüllt.
- Das Feature WORM ermöglicht, dass Daten nicht verändert und gelöscht werden können, aber es können neue Daten am Ende der Datei hinzugefügt werden, wodurch eine Modifizierung des Datenbestandes entsteht. Mit einem Zeitstempel oder einer anderen Markierung könnte am Ende jedes neu erstellten Dokuments sichtbar gemacht werden, wo das Originaldokument aufgehört hat, da nur danach Daten hinzugefügt werden können. Trotzdem wird der Datenbestand modifiziert [RHG16g]. Kriterium 10 (*Unveränderbarkeit*) wird damit nicht erfüllt.
- Wie bereits bei der Evaluation von Kriterium 3 (*Speicherung von Dokumenten*) angemerkt wurde, können alle Dateitypen abgespeichert werden, somit auch hierarchische Dateiformate [RHG16d]. Kriterium 11 (*Hierarchische Datenspeicherung*) wird damit erfüllt.
- GlusterFS ist nur ein Dateisystem und bietet keine Abfragesprache an. Durch eine mögliche Beziehung mit Apache Hadoop kann MapReduce verwendet werden, dies ist aber keine Abfragesprache [RHG16c]. Kriterium 12 (*Abfragesprache für hierarchische Daten*) wird damit nicht erfüllt.

4.8. Existierende Archivierungssysteme

In diesem Abschnitt werden die ausgewählten existierenden Archivierungssysteme (siehe Abschnitt 4.1.3) evaluiert. Angefangen mit Doxis4 in Abschnitt 4.8.1 und anschließend EASY ARCHIVE in Abschnitt 4.8.2. Als letztes Archivierungssystem wird IBM CM8 in Abschnitt 4.8.3 evaluiert.

4.8.1. Doxis4

Doxis4 ist eine Archivierungssoftware der Firma *SERgroup Holding International GmbH* und bietet die Möglichkeit an, Dokumente abzuspeichern und zu archivieren [SHI16].

Doxis4 wird im Folgenden mit Hilfe der Kriterien aus Abschnitt 3.7 evaluiert.

- Eine Benutzerauthentifizierung kann mit einer SSL-Verschlüsselung für die Server-Client Kommunikation eingerichtet werden [SHI16]. Kriterium 1 (*Authentifizierung des Nutzers*) wird damit erfüllt.
- Eine Autorisierung mit verschiedenen Berechtigungen kann eingerichtet werden [SHI16]. Kriterium 2 (*Autorisierung des Nutzers*) wird damit erfüllt.
- Es können beliebige Dokumente abgespeichert werden [SHI16]. Kriterium 3 (*Speicherung von Dokumenten*) wird damit erfüllt.
- Eine Umwandlung der Dokumente findet nicht statt. Kriterium 4 (*Migration ohne Datenverlust*) wird damit erfüllt.
- Einschränkungen bei der Dateigröße konnten keine festgestellt werden. Kriterium 5 (*Datenmodell für kleine Daten*) und Kriterium 6 (*Datenmodell für große Daten*) werden damit erfüllt.
- Eine horizontale Skalierung kann eingerichtet werden [SHI16]. Kriterium 7 (*Skalierbarkeit*) wird damit erfüllt.
- Daten können repliziert auf verteilten Knoten gespeichert werden [SHI16]. Kriterium 8 (*Verteilte Replizierung*) wird damit erfüllt.
- Eine Volltextsuche über die archivierten Daten ist möglich, jedoch existiert kein Datenauswertungsmechanismus [SHI16]. Kriterium 9 (*Verteilter Datenauswertungsmechanismus*) wird damit nicht erfüllt.
- Die Unveränderbarkeit wird durch WORM Bänder ermöglicht, jedoch existieren keine Informationen zur Unveränderbarkeit durch die Ebene des Systems oder der Organisation, da die Granularität der Rechte unbekannt ist [SHI16]. Eine Aussage zur Erfüllung von Kriterium 10 (*Unveränderbarkeit*) kann damit nicht getroffen werden.
- Durch das Dateisystem können alle Dateiformate abgespeichert werden, somit auch hierarchische Dateiformate, wie beispielsweise XML oder JSON [SHI16]. Diese Daten unterliegen keinem Schema, da alle Dateien separat abgespeichert werden. Kriterium 11 (*Hierarchische Datenspeicherung*) wird damit erfüllt.
- Doxis4 fehlt eine Abfragesprache mit der es möglich wäre hierarchische Daten auszuwerten. Kriterium 12 (*Abfragesprache für hierarchische Daten*) wird damit nicht erfüllt.

4.8.2. EASY ARCHIVE

EASY ARCHIVE ist eine Archivierungslösung der Firma *EASY SOFTWARE AG* und bietet die Möglichkeit an, Dokumente abzuspeichern und zu archivieren [ES16a]. In Kombination von weiteren EASY-Produkten soll eine revisionssichere Archivierung von Dokumenten stattfinden [ES16c].

EASY ARCHIVE wird im Folgenden mit Hilfe der Kriterien aus Abschnitt 3.7 evaluiert.

- Informationen zur Authentifizierung konnten nicht gefunden werden. Eine Aussage zur Erfüllung von Kriterium 1 (*Authentifizierung des Nutzers*) kann damit nicht getroffen werden.
- Eine Autorisierung ist durch eine Rechtestruktur gegeben [ES16a]. Kriterium 2 (*Autorisierung des Nutzers*) wird damit erfüllt.
- Dokumente können direkt abgespeichert werden [ES16a]. Kriterium 3 (*Speicherung von Dokumenten*) wird damit erfüllt.
- Eine Umwandlung der Dokumente erfolgt nicht, da diese direkt abgespeichert werden können. Kriterium 4 (*Migration ohne Datenverlust*) wird damit erfüllt.
- Einschränkungen bei der Dateigröße konnten keine festgestellt werden. Kriterium 5 (*Datenmodell für kleine Daten*) und Kriterium 6 (*Datenmodell für große Daten*) werden damit erfüllt.
- Eine Skalierbarkeit ist gegeben, jedoch ist nicht bekannt ob es sich um eine horizontale Skalierbarkeit handelt [ES16b]. Eine Aussage zur Erfüllung von Kriterium 7 (*Skalierbarkeit*) kann damit nicht getroffen werden.
- Eine verteilte redundante Speicherung der Dokumente findet nicht statt, sondern die Dokumente werden redundant auf unterschiedlichen Medien abgespeichert [ES16b]. Kriterium 8 (*Verteilte Replizierung*) wird damit nicht erfüllt.
- Ein Mechanismus zur Datenauswertung konnte nicht gefunden werden. Eine Aussage zur Erfüllung von Kriterium 9 (*Verteilter Datenauswertungsmechanismus*) kann damit nicht getroffen werden.
- Die Unveränderbarkeit wird durch WORM Bänder ermöglicht, jedoch existieren keine Informationen zur Unveränderbarkeit durch die Ebene des Systems oder der Organisation, da die Granularität der Rechte unbekannt ist [ES16a]. Eine Aussage zur Erfüllung von Kriterium 10 (*Unveränderbarkeit*) kann damit nicht getroffen werden.
- Durch das Dateisystem können alle Dateiformate abgespeichert werden, somit auch hierarchische Dateiformate, wie beispielsweise XML oder JSON [ES16a]. Diese Daten unterliegen keinem Schema, da alle Dateien separat abgespeichert werden. Kriterium 11 (*Hierarchische Datenspeicherung*) wird damit erfüllt.
- EASY ARCHIVE fehlt eine Abfragesprache mit der es möglich wäre hierarchische Daten auszuwerten. Kriterium 12 (*Abfragesprache für hierarchische Daten*) wird damit nicht erfüllt.

4.8.3. IBM Content Manager Version 8 (CM8)

IBM Content Manager ist keine Archivierungslösung, sondern eine Lösung zur Speicherung von Dokumenten um Prozesse effizienter zu gestalten. Durch eine Erweiterung von Tivoli Storage Manager for Content Manager werden Archivierungslösungen bereitgestellt, wobei eine Langzeitarchivierung für Daten ermöglicht werden. Die *IBM Content Manager Enterprise Edition*–Software ermöglicht die Speicherung von Dateien, beispielsweise Bilder, Office-Dokumente und XML-Dateien [IBM16].

IBM Content Manager Version 8 wird im Folgenden mit Hilfe der Kriterien aus Abschnitt 3.7 evaluiert.

- Der *Content Manager* verfügt über eine Server Authentifizierung mit Benutzer ID und einem Passwort, dadurch wird nur berechtigten Personen der Zutritt zur Software gewährt [IBM06]. Kriterium 1 (*Authentifizierung des Nutzers*) wird damit erfüllt.
- Sicherheitsfunktionen ermöglichen, dass bestimmte Dokumente vor Personenkreisen geschützt werden können. Diese Autorisierung wird durch unterschiedliche Mechanismen, beispielsweise Nutzer und Nutzergruppen, sowie Privilegien erreicht. Das ermöglicht, dass Personen für Dokumente autorisiert werden können [IBM06]. Kriterium 2 (*Autorisierung des Nutzers*) wird damit erfüllt.
- Dokumente können direkt abgespeichert werden [IBM06]. Kriterium 3 (*Speicherung von Dokumenten*) wird damit erfüllt.
- Eine Umwandlung der Dokumente erfolgt nicht, da diese direkt abgespeichert werden können. Kriterium 4 (*Migration ohne Datenverlust*) wird damit erfüllt.
- Einschränkungen bei der Dateigröße konnten keine festgestellt werden. Kriterium 5 (*Datenmodell für kleine Daten*) und Kriterium 6 (*Datenmodell für große Daten*) werden damit erfüllt.
- Der *Content Manager* lässt sich individuell skalieren [IBM06; IBM16]. Kriterium 7 (*Skalierbarkeit*) wird damit erfüllt.
- Mit dem *Replication Service* können Dokumente repliziert auf verteilten Knoten gespeichert werden [IBM16]. Kriterium 8 (*Verteilte Replizierung*) wird damit erfüllt.
- Ein Mechanismus zur Datenauswertung konnte nicht gefunden werden. Eine Aussage zur Erfüllung von Kriterium 9 (*Verteilter Datenauswertungsmechanismus*) kann damit nicht getroffen werden.
- Durch Kriterium 2 (*Autorisierung des Nutzers*) lassen sich Dokumente vor Veränderungen von bestimmten Personen schützen, jedoch ist die Granularität der Rechte nicht bekannt. Des Weiteren lassen sich durch eine Versionssteuerung mehrere Versionen eines Dokuments abspeichern. Wird die maximale Anzahl an Versionen erreicht, so wird

die erste Version überschrieben [IBM16]. Kriterium 10 (*Unveränderbarkeit*) wird damit nicht erfüllt.

- Durch das Dateisystem können alle Dateiformate abgespeichert werden, somit auch hierarchische Dateiformate, wie beispielsweise XML oder JSON [IBM06; IBM16]. Diese Daten unterliegen keinem Schema, da alle Dateien separat abgespeichert werden. Kriterium 11 (*Hierarchische Datenspeicherung*) wird damit erfüllt.
- Dem *Content Manager* fehlt eine Abfragesprache mit der es möglich wäre hierarchische Daten auszuwerten. Kriterium 12 (*Abfragesprache für hierarchische Daten*) wird damit nicht erfüllt.

4.9. Ergebnisse der Evaluation

In den Abbildungen 4.2 und 4.3 sind die Einordnungen der Datenbanken in das CAP und PACELC zusammengefasst. Titan ist nicht mit aufgeführt, da die Einordnung abhängig ist von der eingesetzten Speichertechnologie. PostgreSQL ist in Abbildung 4.3 nicht aufgelistet, da es keine Partitionierung ermöglicht und somit ist PACELC nicht anwendbar.

Die Ergebnisse der Evaluation sind in der Tabelle 4.3 zusammengefasst. Dabei ist zusehen, welches evaluierte Produkt welches Kriterium erfüllt und welches Kriterium für die Meta-Datenbank und die Archivierungsdatenbank relevant ist. Ein erfülltes Kriterium wird mit einem „✓“ und ein nicht erfülltes Kriterium mit einem „✗“ dargestellt. Kann keine Aussage über die Erfüllung des Kriteriums getroffen werden, so ist „–“ dargestellt.

Bei den Datenbanken hat die dokumentenorientierte Datenbank MongoDB alle Kriterien erfüllt, somit ist sie als Archivierungs- und als Meta-Datenbank geeignet. Die spaltenorientierte Datenbank Apache HBase erfüllt alle Kriterien der Archivierungsdatenbank. Die dokumentenorientierte Datenbank Apache CouchDB und die spaltenorientierte Datenbank Apache Cassandra erfüllen alle Kriterien der Meta-Datenbank. Die anderen evaluierten Datenbanken erfüllen mindestens ein Kriterium der Archivierungs- und Meta-Datenbank nicht.

Bei den Dateisystemen und den existierenden Archivierungssystemen erfüllt kein evaluiertes Produkt alle Kriterien der Archivierungs- und der Meta-Datenbank. Bei den existierenden Archivierungssystemen kommen die Nachteile dazu, dass die Archivierungslösung komplett von einem Hersteller abhängig ist, wodurch keine Anpassung der Software möglich ist und es muss darauf vertraut werden, dass der Hersteller die Daten wirklich revisionssicher abspeichert.

Aus diesen Ergebnissen folgt, dass die Datenbanken Apache Cassandra, Apache CouchDB und MongoDB als Meta-Datenbank geeignet sind. Als Archivierungsdatenbank eignet sich Apache HBase und MongoDB. Daraus folgt, dass sechs Kombinationen der Datenbanken als Aufbau des Archivierungssystem existieren. Um die optimale Kombination zu erhalten, wird die Performance der Datenbanken verglichen. Dabei sind die Lese- und Schreibzugriffe wichtig

und nicht die Ausführung von *Update* oder *Delete* Befehlen, da diese beim Archivierungssystem nur in Ausnahmefällen möglich sein sollen.

Zunächst werden die Meta-Datenbanken verglichen. Im Vergleich von den dokumentenorientierten Datenbanken Apache CouchDB und MongoDB hat MongoDB die deutlich bessere Performance und kann somit schneller Daten schreiben und abfragen [Hen11; LM13]. Somit wird Apache CouchDB nicht mehr als mögliche Meta-Datenbank betrachtet. Im zweiten Vergleich wird die spaltenorientierte Datenbank Apache Cassandra mit der dokumentenorientierten Datenbank MongoDB verglichen. Apache Cassandra besitzt laut einigen Experimenten eine bessere Performance [AB13; Bus12], jedoch nicht in allen Experimenten schneidet Apache Cassandra besser ab als MongoDB [LM13]. Zu berücksichtigen ist, dass sich die Performance der Datenbanken über die Zeit verbessert hat, wie man beispielsweise an MongoDB sehen kann [Mon15b]. Für den Einsatz von Apache Cassandra im Archivierungssystem ist die Performance mit JSON-Dokumenten relevant. Jedoch kann keine Aussage über die Performance mit JSON-Dokumenten getroffen werden, da keine Experimente von Apache Cassandra gefunden werden konnten, wobei die Performance anhand von JSON-Dokumenten ermittelt wurde. Aus diesem Grund wird die Annahme gemacht, dass die Performance der dokumentenorientierten Datenbank MongoDB zur Speicherung und Abfrage von JSON-Dokumenten besser geeignet ist. Somit wird MongoDB als Meta-Datenbank ausgewählt.

Als Archivierungsdatenbank stehen die spaltenorientierte Datenbank Apache HBase und die dokumentenorientierte Datenbank MongoDB zur Auswahl. Laut Experimenten hat MongoDB eine bessere Performance bei Lesezugriffen, aber eine schlechtere Performance bei Schreibzugriffen als Apache HBase [ABF14; Bus12]. Da die Archivierungsdatenbank hauptsächlich zum Schreiben benutzt werden soll und weniger zum Lesen, ist die Performance beim Schreiben wichtiger. Aus diesem Grund wird Apache HBase als Archivierungsdatenbank genommen.

Somit besitzt das Archivierungssystem als Meta-Datenbank MongoDB und als Archivierungsdatenbank Apache HBase. Beide Datenbanken geben die Latenz im partitionsfreien Betrieb für die Konsistenz auf. Tritt eine Partitionierung auf, so gibt die Archivierungsdatenbank Apache HBase die Verfügbarkeit für die Konsistenz auf hingegen die Meta-Datenbank MongoDB die Konsistenz für die Verfügbarkeit aufgibt.

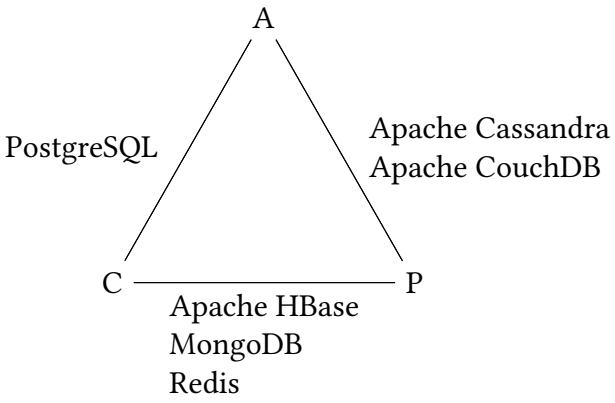

Abbildung 4.2.: Einordnung der evaluierten Datenbanken in CAP

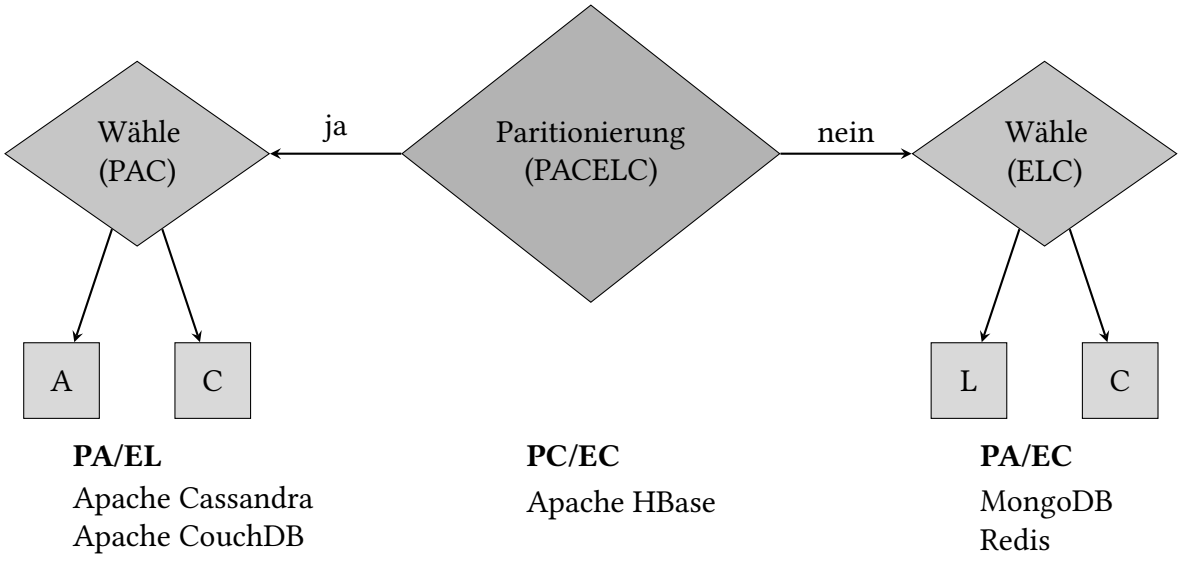

Abbildung 4.3.: Einordnung der evaluierten Datenbanken in PACELC

		Datenbanken							Dateisysteme			Existierende		
	Einordnung[1]	Apache CouchDB	MongoDB	Apache Cassandra	Apache HBase	Redis	Titan	PostgreSQL	Apache Hadoop Framework	BeeGFS	GlusterFS	Doxis4	EASY ARCHIVE	IBM CM8
Kriterium 1 (*Authentifizierung des Nutzers*)	B	✓	✓	✓	✓	✗	✓	✓	✓	–	✗	✓	–	✓
Kriterium 2 (*Autorisierung des Nutzers*)	B	✓	✓	✓	✓	✗	–	✓	✓	–	✗	✓	✓	✓
Kriterium 3 (*Speicherung von Dokumenten*)	A	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Kriterium 4 (*Migration ohne Datenverlust*)	B	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
Kriterium 5 (*Datenmodell für kleine Daten*)	A	✓	✓	✓	✓	✗	–	✓	✗	✗	✓	✓	✓	✓
Kriterium 6 (*Datenmodell für große Daten*)	A	✓	✓	✗	✓	✗	–	✓	✗	✓	✓	✓	✓	✓
Kriterium 7 (*Skalierbarkeit*)	B	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	–	✓
Kriterium 8 (*Verteilte Replizierung*)	B	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓
Kriterium 9 (*Verteilter Datenauswertungsmechanismus*)	A	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	–	–
Kriterium 10 (*Unveränderbarkeit*)	B	✓	✓	✓	✓	–	–	✓	✓	–	✗	–	–	✗
Kriterium 11 (*Hierarchische Datenspeicherung*)	M	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓
Kriterium 12 (*Abfragesprache für hierarchische Daten*)	M	✓	✓	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗

Tabelle 4.3.: Ergebnisse der Evaluation

[1] Kriterium benötigt für Archivierungsdatenbank (A), Meta-Datenbank (M), beide Datenbanken (B)

5. Implementierung

In Abschnitt 4.9 wurde die beste Kombination von Datenbanken gefunden, um ein Archivierungssystem mit den Anforderungen aus Abschnitt 3.1 und Abschnitt 3.2 zu erstellen. Die Kombination aus der Meta-Datenbank MongoDB und der Archivierungsdatenbank Apache HBase wird in diesem Kapitel verwendet um einen Prototypen eines Archivierungssystems zu erstellen. Ein Prototyp wird erstellt, um den Aufbau, die Verbindung der Datenbanken untereinander, die enthaltenen Daten und die Anbindung des Archivierungssystems an ein System zu veranschaulichen. Apache HBase wird mit Version 1.2.2 und MongoDB mit Version 3.2.8 verwendet.

Eine Evaluation des Prototypen findet im Rahmen dieser Arbeit nicht statt, da der implementierte Prototyp nur eine vergleichsweise geringe Funktionalität gegenüber den Anforderungen aufweist, da es sich hierbei um einen Prototypen handelt. Ein vollständiges und funktionierendes Archivierungssystem, welches alle Anforderungen erfüllt, kann in dieser Arbeit nicht erstellt werden, da es den Rahmen dieser Arbeit übersteigt. Des Weiteren soll in dieser Arbeit ein funktionsfähiger Prototyp entwickelt werden und kein kommerzielles Produkt.

In diesem Kapitel wird die Auswahl der Metadaten in Abschnitt 5.1 beschrieben. Anschließend wird der Aufbau des Archivierungssystems in Abschnitt 5.2 erklärt. Die Verbindung der Datenbanken untereinander wird durch einen Verweis ermöglicht, welcher in Abschnitt 5.3 näher erklärt wird.

5.1. Auswahl der Metadaten

Im Archivierungssystem sollen unterschiedliche Dateien archiviert werden, wobei eine XML-Datei (siehe Abschnitt 3.1.2) enthalten ist. Diese XML-Datei enthält eine hierarchische Gliederung aller verbauter Komponenten und weitere Qualitätsdaten. Daher werden die Metadaten aus der XML-Datei gewonnen.

Zunächst wird der Aufbau der XML-Datei erklärt, wobei der schematische Aufbau in Listing 5.1 zusehen ist. Die Tags `<UNIQUEPART_DATA>`, `<ADD_DATA>`, `<RESULT_DATA>`, `<COMPONENT_DATA>` wiederholen sich rekursiv. `<PACKAGING_DATA>` wird hingegen nur einmal benötigt. Die verwendeten Tags haben die folgenden Bedeutungen:

Listing 5.1 Schematischer Aufbau der XML-Datei mit Qualitätsdaten

```
<DATA>
      <UNIQUEPART_DATA>...</UNIQUEPART_DATA>
      <PACKAGING_DATA>...</PACKAGING_DATA>
      <ADD_DATA>...</ADD_DATA>
      <RESULT_DATA>...</RESULT_DATA>
      <COMPONENT_DATA>
            <COMPONENT>
                  <UNIQUEPART_DATA>...</UNIQUEPART_DATA>
                  <ADD_DATA>...</ADD_DATA>
                  <RESULT_DATA>...</RESULT_DATA>
                  <COMPONENT_DATA>...</COMPONENT_DATA>
            </COMPONENT>
            <COMPONENT>...</COMPONENT>
            ...
      </COMPONENT_DATA>
</DATA>
```

- `<UNIQUEPART_DATA>` enthält die aktuell betrachtete Komponente mit Ausnahme des ersten Vorkommens, hierbei bezieht es sich auf das Produkt.
- `<PACKAGING_DATA>` enthält eine Liste an Informationen zur Logistik, wie beispielsweise die Box oder Paletten Nummer.
- `<ADD_DATA>` enthält eine Liste an zusätzlichen Informationen der aktuellen Komponente bzw. dem Produkt, welche sich in den separaten Dateien befinden.
- `<RESULT_DATA>` enthält eine Liste an durchlaufenen Stationen, wobei die Stationen wiederum Messwerte mit Toleranzen enthalten können.
- `<COMPONENT_DATA>` enthält eine Liste an Komponenten, welche in der aktuellen Komponenten bzw. im Produkt direkt enthalten sind. Beispielsweise besteht Komponente A aus den Komponenten B und C, wobei B aus D und E besteht, somit besteht Komponente A direkt aus B und C und indirekt aus D und E.
- `<COMPONENT>` definiert eine Komponente, welche weitere Komponenten, Stationen und Zusatzinformationen enthalten kann.

Der Inhalt der Metadaten ist von den Anwendungsfällen abhängig. Die zwei Anwendungsfälle der Qualitätsdaten (siehe Abschnitt 2.2) werden um einen dritten Anwendungsfall ergänzt. Diese drei Anwendungsfälle sind im Folgenden dargestellt:

Listing 5.2 Schematischer Aufbau der Metadaten als JSON-Format

```
{
      "UNIQUEPART_DATA":{
            ...
      },
      "COMPONENT_DATA":[
            {
                  ...
                  "ID":"1"
                  ...
            },
            {
                  ...
                  "PARENT":"1"
            }
            ...
      ],
      "PACKAGING_DATA":[
            ...
      ]
}
```

- Bei defekten Komponenten sollen alle Produkte gefunden werden können, worin die Komponenten verbaut wurden (siehe Abschnitt 2.2).
- Bei defekten Produkten sollen alle verbauten Komponenten gefunden werden können, um die fehlerhafte Komponente oder Station zu finden (siehe Abschnitt 2.2).
- Bei Problemen in der Logistik, bspw. falsches Ziel oder ein Unfall beim Transport, sollen alle betroffenen Produkte ermittelt werden können [PB16].

Aus diesen Anwendungsfällen folgt, dass Informationen zum Produkt, den Komponenten und der Logistik in die Metadaten gehören. Alle restlichen Informationen werden nicht als Metadaten gebraucht. Eine weitere Änderung der Metadaten gegenüber den Qualitätsdaten ist, dass eine Strukturveränderung vorgenommen wird, sodass eine geringere Verschachtelungstiefe entsteht und dadurch eine einfachere Abfrage möglich wird. Dazu werden alle `<COMPONENT>` auf derselben Ebene gelistet und um ein *PARENT* Attribut ergänzt. Dadurch fällt die Verschachtelung von `<COMPONENT_DATA>` weg.

Da die dokumentenorientierte Datenbank MongoDB (siehe Abschnitt 4.9) verwendet wird, welche JSON-Dokumente speichern kann, wird kein XML-Format benötigt, sondern ein JSON-Format. Daher werden die Metadaten nicht als XML-Format erstellt, sondern als JSON-Format. Die Metadaten als JSON-Format mit der Strukturveränderung sind in Listing 5.2 zusehen.

5.2. Aufbau

Ein eigenständiges Archivierungssystem, welches wenige Abhängigkeiten zu anderen Services besitzt, kann beliebig zugeschalten werden und hat keine Auswirkungen auf die anderen Services. Das hat den Vorteil, dass das aktuelle Archivierungssystem weitergenutzt werden kann, aber parallel das neue Archivierungssystem eingebunden werden kann, ohne dass das aktuelle Archivierungssystem beeinflusst wird. Dieser schematische Aufbau ist in Abbildung 5.1 zusehen, welcher im Folgenden näher beschrieben wird.

Um das neue Archivierungssystem nutzen zu können, wird als einziges ein Dateiverzeichnis benötigt, welches vom aktuellen und neuen Archivierungssystem zugreifbar ist. Das aktuelle Archivierungssystem muss um einen Service (*Archiving Control*) ergänzt werden, welcher die zu archivierenden Hauptdaten im Dateiverzeichnis ablegt. Dabei ist zu beachten, dass die Daten immer die gleiche Struktur haben müssen, sodass keine Konvertierung der Struktur benötigt wird. Beispielsweise sollte die Ordnerstruktur und die angebotene Form (Ordner oder komprimiertes Archiv) immer gleich sein. Ein zweiter Service (*Archiving Service*), welcher vom neuen Archivierungssystem bereit gestellt wird, prüft zyklisch das Dateiverzeichnis nach neuen Hauptdaten und filtert diese anschließend, sodass Metadaten und die Hauptdaten archiviert werden können.

Um die archivierten Hauptdaten oder Metadaten abzufragen, wird ein Service (*Archiving Query Service*) vom Archivierungssystem angeboten, welcher die Meta-Datenbank und die Archivierungsdatenbank ansteuert und die jeweils benötigten Informationen zurückliefert. Dieser Service kann über einen eigenen Service (*Web Browser*) beispielsweise mit HTTP angesteuert werden, sodass keine Abhängigkeit zwischen den Services entsteht.

Da es sich hierbei um einen schematischen Aufbau handelt, wird im Folgenden der Aufbau des entwickelnden Prototyps vom Archivierungssystem anhand von Abbildung 5.2 beschrieben. Die Hauptdaten des Archivierungssystems sind Qualitätsdaten. Die eingesetzten Datenbanktechnologien sind als Archivierungsdatenbank Apache HBase und als Meta-Datenbank MongoDB (siehe Abschnitt 4.9). Zur Datenabfrage wird ein REST-Service verwendet, sodass verschiedene Datenabfragen mit unterschiedlicher Anzahl an Parametern ermöglicht wird. Bei der Abbildung 5.2 ist zusehen, dass die Datenbanken nicht verteilt arbeiten, wie in Abschnitt 3.2 gefordert ist, das liegt daran, dass es sich um einen Prototypen handelt. Um eine verteilte Datenspeicherung zu ermöglichen, sollte der *Database Server* mehrfach dupliziert werden und anschließend eine Anpassung der Konfiguration von Apache HBase und MongoDB vorgekommen werden.

Bei dem hohen Datenaufkommen (siehe Abschnitt 3.1.2) ist ein hoher Durchsatz des *Archiving Service* wichtig, daher wird Apache Storm [ASF16] eingesetzt, sodass die Daten parallel und verteilt verarbeitet werden können. Existiert ein hohes Datenaufkommen, können neue Knoten zugeschalten werden, wodurch mehr Durchsatz erreicht wird.

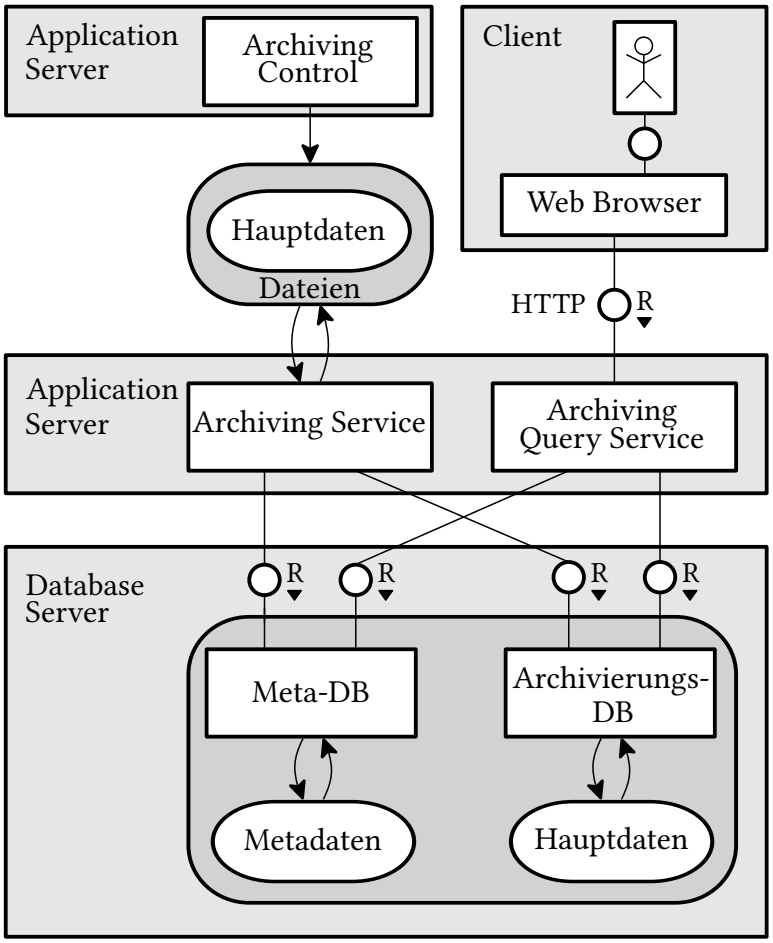

Abbildung 5.1.: Schematischer Aufbau des Prototyps vom Archivierungssystem

5.3. Verweise von den Metadaten auf die Qualitätsdaten

Die archivierten Metadaten verweisen auf die archivierten Qualitätsdaten. Dieser Verweis wird durch einen eindeutigen Datenbanklink realisiert. Die Metadaten erhalten ein extra Attribut mit dem Datenbanklink, wie in Listing 5.3 zusehen ist. Die Archivierungsdatenbank wird um eine weitere Spalte mit dem Datenbanklink erweitert. Somit ist der Datenbanklink ein Verweis von den Metadaten auf die Qualitätsdaten.

Da der Datenbanklink eindeutig sein muss, muss der Aufbau des Datenbanklinks so gewählt werden, dass keine Duplikate vorkommen können oder es muss nach der Erstellung des Datenbanklinks geprüft werden, ob dieser bereits verwendet wird.

Im entwickelnden Archivierungssystem wird versucht einen eindeutigen Datenbanklink zu erstellen, wobei anschließend geprüft wird, ob dieser bereits verwendet wird. Um mögliche Mehrfacherstellungen und Mehrfachprüfungen des Datenbanklinks zu vermeiden, wird ein

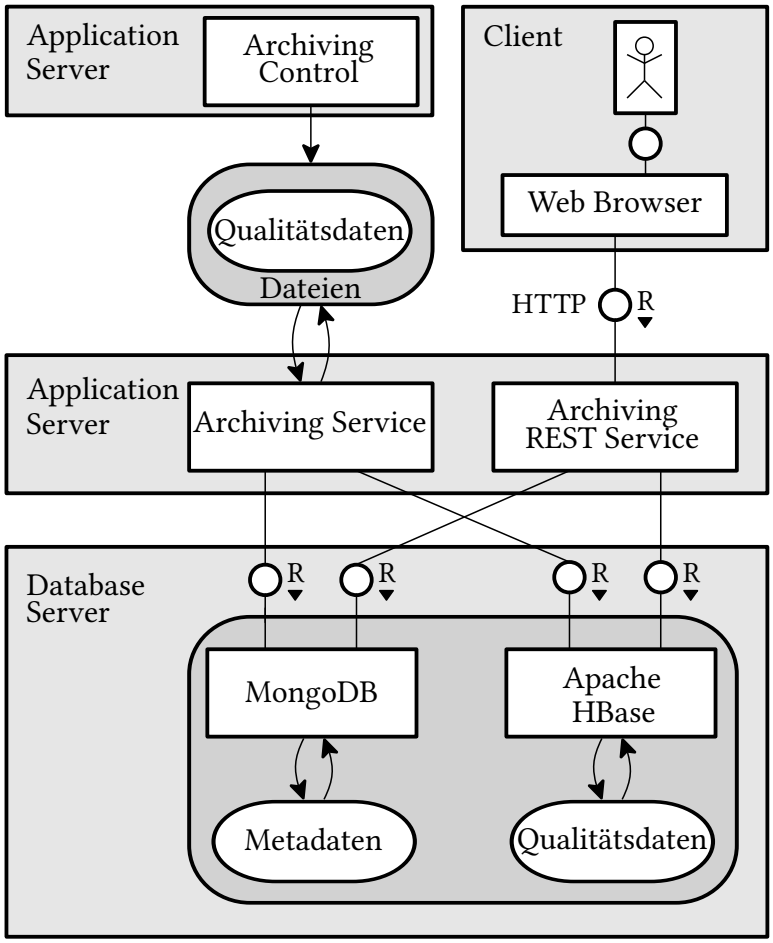

Abbildung 5.2.: Aufbau des Prototyps vom Archivierungssystem

längerer Datenbanklink verwendet. Dabei beträgt die ganze Länge des Datenbanklinks 94 Zeichen.

Der Aufbau eines Datenbanklinks ist in Tabelle 5.1 zusehen, wobei die einzelnen Segmente mit einem Unterstrich verbunden werden. Mit der *Service ID* kann ermittelt werden, welcher Service die Daten archiviert hat. Des Weiteren können unterschiedliche Services gleichzeitig Daten archivieren, ohne ein Duplikat eines Datenbanklinks zu erzeugen.

Listing 5.3 Schematischer Aufbau der Metadaten mit einem Datenbanklink

```
{
      "UNIQUEPART_DATA":{
            ...
      },
      "COMPONENT_DATA":[
            {
                  ...
                  "ID":"1"
                  ...
            },
            {
                  ...
                  "PARENT":"1"
            }
            ...
      ],
      "PACKAGING_DATA":[
            ...
      ],
      "REFERENCE":"2016-07-20T16:42:12.345+0200_005432187654321_d45...144_f4f...641"
}
```

Beschreibung	Zeichenanzahl	Beispiel
Zeitstempel der Archivierung	28	2016-07-20T16:42:12.345+0200
Laufzeit der JVM in Nanosekunden – Letzte 15 Ziffern	15	005432187654321
UUID – Ohne Bindestrich	32	d4565cf9ef4a4c5c9f79638065193144
Service ID	16	f4f855b895a24641

Tabelle 5.1.: Aufbau eines Datenbanklinks

6. Zusammenfassung und Ausblick

Das Thema dieser Arbeit ist die Entwicklung eines Archivierungssystems für Qualitätsdaten. Dazu wurde in Kapitel 2 auf Begriffe eingegangen, welche für Archivierungssysteme wichtig sind. In Kapitel 3 wurden die Anforderungen an ein Archivierungssystem für Qualitätsdaten erhoben. Dabei wurde ein aktuelles Archivierungssystem evaluiert, um Angaben zum Mengengerüst zu erhalten. Aus diesem Mengengerüst wurden Anforderungen gewonnen. Weitere Anforderungen wurden von rechtlicher Seite und von Kundenseite erhoben. Zur Erfüllung der Anforderungen wird ein Aufbau des Systems gewählt, sodass Metadaten und Qualitätsdaten separat archiviert werden. Für diesen Aufbau wurde eine Meta-Datenbank und eine Archivierungsdatenbank mit unterschiedlichen Anforderungen und Kriterien benötigt. Insgesamt entstanden zwölf Kriterien.

In Kapitel 4 wurden Evaluationsobjekte bestimmt und mit Hilfe der zwölf Kriterien evaluiert. Zur Evaluation wurden sieben Datenbanksysteme, drei Dateisysteme und drei Archivierungslösungen genommen. Die Archivierungslösungen wurden evaluiert, sodass geprüft werden konnte, ob es Archivierungslösungen auf dem Markt gibt, welche die geforderten Anforderungen erfüllt. Nach der Evaluierung wurden die Evaluationsobjekte gegenübergestellt, sodass geprüft werden konnte, welches Evaluationsobjekte alle Kriterien der jeweiligen Meta- oder Archivierungsdatenbank erfüllt. Die Kriterien der Meta-Datenbank wurden von drei Datenbanksystemen erfüllt, hingegen die Kriterien der Archivierungsdatenbank von zwei Datenbanksystemen erfüllt wurden. Die Wahl der Datenbanksysteme wurde anhand von Performance Werten ermittelt. Der Aufbau eines optimalen Archivierungssystems besitzt somit als Archivierungsdatenbank Apache HBase und als Meta-Datenbank MongoDB.

In Kapitel 5 wurde ein Prototyp eines Archivierungssystems mit den zuvor festgelegten Datenbanksystemen entwickelt. Zur Bestimmung der Metadaten für die Meta-Datenbank wurden drei Anwendungsfälle in Betracht gezogen, sodass daraus die Metadaten gewonnen werden konnten. Der Aufbau des Archivierungssystems wurde so gewählt, dass wenige Abhängigkeiten existieren, wodurch das Archivierungssystem von jedem anderen Service genutzt werden kann. Der Verweis der Meta-Datenbank auf die Archivierungsdatenbank wird durch einen eindeutigen Datenbanklink ermöglicht, wodurch die Daten in der Archivierungsdatenbank eindeutig zu den Metadaten zuzuordnen sind.

Ausblick

Diese Arbeit diente bereits dazu, Prozesse des aktuellen Archivierungssystems zu prüfen und neue Technologien einzuführen, welche Optimierung und neue Funktionalitäten versprechen.

Im Rahmen dieser Arbeit konnten, bezogen auf den Prototypen, nicht alle Punkte vollständig geprüft werden. Daher bieten sich folgende Punkte für zukünftige Arbeiten an.

Bei der Auswahl der optimalen Meta-Datenbank wurde die Annahme getroffen, dass die Performance der dokumentenorientierten Datenbank MongoDB besser ist als die spaltenorientierte Datenbank Apache Cassandra, da es sich bei den zu speichernden Daten um JSON-Dokumente handelt. Ein Vergleich, ob eine spaltenorientierte Datenbank, welche die Speicherung und Abfrage von JSON-Dokumenten erlaubt, eine bessere Performance aufweist als eine reine dokumentenorientierte Datenbank könnte durch ein Experiment ermittelt werden.

Da kein vollständiges Archivierungssystem entwickelt wurde, sondern ein Prototyp, bestand nicht die Möglichkeit das System zu evaluieren und die Kriterien anhand des Systems zu prüfen. Dadurch bleibt offen, ob alle Kriterien praktisch erfüllt werden und nicht nur theoretisch.

In dieser Arbeit wurden Grundzüge des rechtlichen Rahmens in Abschnitt 2.2 betrachtet. Der Hauptfokus der Arbeit lag auf einer fundierten Auswahl von Datenbanksystemen für ein Archivierungssystem und einer prototypischen Umsetzung eines solchen Systems. Für ein Produktiveinsatz werden zwei Aspekte gesehen: 1. Die Härtung des Systems, so dass es die industriellen Anforderungen an Zuverlässigkeit erfüllt und 2. Die rechtliche Prüfung der Implementierung, sodass das System mit den geltenden Vorschriften konform ist.

A. Anhang

A.1. Liste an Datenbanken

In Abschnitt 4.1.1 werden Datenbanken als Evaluationsobjekte gesucht. Zur Bestimmung von existierenden Datenbanken wird die folgende Suche durchgeführt.

Eine Google-Suche mit den Schlüsselbegriffen *big data, nosql database, wide column, key-value, document* und *graph* lieferte am 07. August 2016 zwischen 18:20 und 18:30 Uhr, Zeitzone Berlin (UTC+01:00), 2.590 Resultate. Die genaue Such-URL lautet https://www.google.de/search?q=%22big%20data%22%20%22nosql%20database%22%20%22wide%20column%22%20%22key-value%22%20%22document%22%20%22graph%22.

Unter den ersten zehn Resultaten sind zwei Übersichtsseiten mit Datenbanken [Edl16; Sar14]. Bei den aufgelisteten Datenbanken, der beiden Übersichtsseiten, wurden die Links geprüft, ob diese auf eine Produktseite oder auf GitHub verweisen. Im Folgenden sind die Datenbanken aufgelistet, welche dieses Kriterium erfüllen.

Dokumentenorientierte Datenbanken

- Amisalabs DBaaS [ALA16]
- Apache Jackrabbit [ASFJ16]
- ArangoDB [Ara16]
- BaseX [BXT15]
- Clusterpoint [CP16]
- Couchbase Server [CBS16]
- CouchDB [ASFCO16e]
- DensoDB [MCD16]
- Djondb [DDB16]
- Elasticsearch [ES16d]
- eXistdb [EXS16]
- JasDB [OBT15]
- gunDB [Gun16]
- MarkLogic [MLC16]
- MongoDB [Mon16e]
- NeDB [GHN16b]

- OrientDB [ODO16]
- RaptorDB [MCR16]
- RavenDB [HRR15]
- RethinkDB [RT16]
- SisoDb [GHN16d]
- Terrastore [GOO16]
- ThruDB [GHT16]
- ToroDB [8K16]
- UnQLite [SSu16]

Spaltenorientierte Datenbanken

- Amazon SimpleDB [AWS16c]
- Apache Accumulo [ASFA16]
- Apache Cassandra [ASFC15]
- Apache HBase [ASFB16]
- Apache Kudu [ASFK16]
- ConcourceDB [Cin16]
- Druid [TDc16]
- Hypertable [HTH14]
- MonetDB [Mon15a]
- Scylla [Scy16]
- Splice [SMS16]

Key-Value Datenbanken

- Aerospike [ASA16]
- Amazon DynamoDB [AWS16a]
- BangDB [IQL16]
- Berkeley DB [OCB16]
- BoltDB [GHN16a]
- c-treeACE [FCC16]
- DynomiteDB [DD16]
- Hibari DB [HD15]
- HyperDex [Cor16]
- KitaroDB [SK16]
- LevelDB [DG16]
- MemcacheDB [Chu09]
- NessDB [GHN16c]
- Oracle NoSQL Database [OCO16]
- PickleDB [Erd16]
- quasardb [QDQ16]

- RaptorDB [MCR16]
- Redis [San16c]
- Riak [BTR16]
- RocksDB [FDET16]
- Scalaris [ZIB16]
- STSdb [SSS16]
- Tarantool [MRG16]
- TIBCO ActiveSpaces DB [TSA16b]
- Tokyo Cabinet [TSA16a]
- upscaledb [Rup16]
- Voldemort [LIV16]

Graphenorientierte Datenbanken

- AllegroGraph [FIA16]
- ArangoDB [Ara16]
- Fallen-8 [Rau16]
- FlockDB [GHF16]
- GraphBase [Gra16]
- gunDB [Gun16]
- HypergraphDB [KSH10]
- InfiniteGraph [OBI16]
- Infogrid [NMI16]
- Neo4J [NTI16]
- OrientDB [ODO16]
- Sparksee (DEX) [STS16]
- Titan [DST16h]
- Trinity [MCD10]
- VertxDB [DC09]
- WhiteDB [WTW13]

A.2. Google-Suchen

A.2.1. Dateisysteme

In Abschnitt 4.1.2 werden Dateisysteme als Evaluationsobjekte gesucht. Zur Bestimmung von existierenden Dateisystemen wird die folgende Suche durchgeführt.

Eine Google-Suche mit den Schlüsselbegriffen *big data, filesystem, ecosystem, scalable* und *parallel* lieferte am 07. August 2016 zwischen 18:50 und 19:00 Uhr, Zeitzone Berlin (UTC+01:00), 4.970

Resultate. Die genaue Such-URL lautet https://www.google.de/search?q=%22Big%20Data%22%20%22filesystem%22%20%22Ecosystem%22%20%22scalable%22%20%22parallel%22.

A.2.2. Existierende Archivierungslösungen

In Abschnitt 4.1.3 werden Archivierungslösungen als Evaluationsobjekte gesucht. Zur Bestimmung von existierenden Archivierungslösungen wird die folgende Suche durchgeführt.

Eine Google-Suche mit dem Schlüsselbegriff *Archivierungslösung* lieferte am 07. August 2016 zwischen 19:10 und 19:20 Uhr, Zeitzone Berlin (UTC+01:00), 18.500 Resultate. Die genaue Such-URL lautet https://www.google.de/search?q=%22Archivierungsl%C3%B6sung%22.

A.2.3. Existierende Archivierungslösungen für Qualitätsdaten

In Abschnitt 4.1.3 werden Archivierungslösungen für Qualitätsdaten als Evaluationsobjekte gesucht. Zur Bestimmung von existierenden Archivierungslösungen wird die folgende Suche durchgeführt.

Eine Google-Suche mit dem Schlüsselbegriff *Qualitätsdaten* und *Archivierung* lieferte am 07. August 2016 zwischen 19:30 und 19:40 Uhr, Zeitzone Berlin (UTC+01:00), 4.080 Resultate. Die genaue Such-URL lautet https://www.google.de/?q=%22Qualit%C3%A4tsdaten%22+%22Archivierung%22.

Abkürzungsverzeichnis

Abkürzung	Bedeutung	Erstes Vorkommen
BDE	Betriebsdatenerfassung	17
BLOB	Binary Large Object	35
BSON	Binary JSON	46
CAQ-System	Computer Aided Quality-System	15
GridFS	Grid File System	47
HDFS	Hadoop Distributed File System	49
MOB	Medium-sized Object	49
NoSQL	Not only SQL	40
QIS	Qualitätsinformationssystem	19
RDBMS	Relationale Datenbankmanagementsysteme	15
SQL	Structured Query Language	40
WORM	Wirte Once Read Many	28

Literaturverzeichnis

[8K16] 8Kdata. *ToroDB*. 2016. URL: http://www.8kdata.com/torodb/ (zitiert auf S. 80).

[AB13] V. Abramova, J. Bernardino. „NoSQL Databases: MongoDB vs Cassandra“. In: *C3S2E '13 Proceedings of the International C* Conference on Computer Science and Software Engineering* (2013), S. 14–22 (zitiert auf S. 46, 48, 65).

[Aba11] D. J. Abadi. *CAP, PACELC, and Determinism*. Jan. 2011 (zitiert auf S. 45).

[Aba12] D. J. Abadi. „Consistency Tradeoffs in Modern Distributed Database System Design“. In: *Computer* 45 (2012), S. 37–42 (zitiert auf S. 21, 46, 48, 49).

[ABF14] V. Abramova, J. Bernardino, P. Furtado. „Experimental evaluation of NoSQL databases“. In: *International Journal of Database Management Systems (IJDMS) Vol.6, No.3* (Juni 2014) (zitiert auf S. 65).

[ADS16] apromace data systems GmbH. *Manufacturing Execution System*. 2016. URL: http://www.apromace.de/loesung.html (zitiert auf S. 43).

[AHT16a] Apache HBase Team. *Apache HBase ™ Reference Guide: Access Control Labels (ACLs)*. 2016. URL: https://hbase.apache.org/book.html#hbase.accesscontrol.configuration (zitiert auf S. 49, 50).

[AHT16b] Apache HBase Team. *Apache HBase ™ Reference Guide: Advanced - Fully Distributed*. 2016. URL: https://hbase.apache.org/book.html#quickstart_fully_distributed (zitiert auf S. 49).

[AHT16c] Apache HBase Team. *Apache HBase ™ Reference Guide: HBase and MapReduce*. 2016. URL: https://hbase.apache.org/book.html#mapreduce (zitiert auf S. 50).

[AHT16d] Apache HBase Team. *Apache HBase ™ Reference Guide: HBase run modes: Standalone and Distributed*. 2016. URL: https://hbase.apache.org/book.html#standalone_dist (zitiert auf S. 50).

[AHT16e] Apache HBase Team. *Apache HBase ™ Reference Guide: Network Consistency and Partition Tolerance*. 2016. URL: https://hbase.apache.org/book.html#perf.network.call_me_maybe (zitiert auf S. 49).

[AHT16f] Apache HBase Team. *Apache HBase ™ Reference Guide: Secure Client Access to Apache HBase*. 2016. URL: https://hbase.apache.org/book.html#hbase.secure.configuration (zitiert auf S. 49).

[AHT16g] Apache HBase Team. *Apache HBase ™ Reference Guide: Storing Medium-sized Objects (MOB)*. 2016. URL: https://hbase.apache.org/book.html#hbase_mob (zitiert auf S. 50).

[AHT16h] Apache HBase Team. *Apache HBase ™ Reference Guide: Table Schema Rules Of Thumb*. 2016. URL: https://hbase.apache.org/book.html#table_schema_rules_of_thumb (zitiert auf S. 50).

[AIS16] Allgeier IT Solutions GmbH. *scanview®*. 2016. URL: http://www.allgeier-it.de/business-solutions/syntona-logic/zusatz-module/prozesse-optimieren/scanview (zitiert auf S. 43).

[ALA16] Amisalabs. *MongoDB-Compatible Database Service*. 2016. URL: http://www.amisalabs.com/ (zitiert auf S. 79).

[ALN10a] J. C. Anderson, J. Lehnardt, S. Noah. *CouchDB: The Definitive Guide*. Hrsg. von M. Loukides. OReilly Media, Jan. 2010, S. 12–13 (zitiert auf S. 45).

[ALN10b] J. C. Anderson, J. Lehnardt, S. Noah. *CouchDB: The Definitive Guide*. Hrsg. von M. Loukides. OReilly Media, Jan. 2010, S. 145–147 (zitiert auf S. 45).

[ALN10c] J. C. Anderson, J. Lehnardt, S. Noah. *CouchDB: The Definitive Guide*. Hrsg. von M. Loukides. OReilly Media, Jan. 2010, S. 165–170 (zitiert auf S. 45).

[ALN10d] J. C. Anderson, J. Lehnardt, S. Noah. *CouchDB: The Definitive Guide*. Hrsg. von M. Loukides. OReilly Media, Jan. 2010, S. 27–31 (zitiert auf S. 45).

[ALN10e] J. C. Anderson, J. Lehnardt, S. Noah. *CouchDB: The Definitive Guide*. Hrsg. von M. Loukides. OReilly Media, Jan. 2010, S. 119–130 (zitiert auf S. 46).

[ALN10f] J. C. Anderson, J. Lehnardt, S. Noah. *CouchDB: The Definitive Guide*. Hrsg. von M. Loukides. OReilly Media, Jan. 2010, S. 179–187 (zitiert auf S. 46).

[Ara16] ArangoDB GmbH. *ArangoDB*. 2016. URL: https://www.arangodb.com/ (zitiert auf S. 79, 81).

[ARG16] ARBURG GmbH + Co KG. *Leitrechnersystem (ALS)*. 2016. URL: https://www.arburg.com/de/de/leistungsspektrum/spritzgiessen/produktionsoptimierung/leitrechnersystem-als/ (zitiert auf S. 43).

[ASA16] Aerospike, Inc. *High performance NoSQL database delivering speed at scale*. 2016. URL: http://www.aerospike.com/ (zitiert auf S. 80).

[ASF16] The Apache Software Foundation. *Apache Storm*. 2016. URL: http://storm.apache.org/ (zitiert auf S. 72).

[ASFA16] The Apache Software Foundation. *accumulo*. 2016. URL: https://accumulo.apache.org/ (zitiert auf S. 80).

[ASFB16] The Apache Software Foundation. *Welcome to Apache HBase™*. 2016. URL: https://hbase.apache.org/ (zitiert auf S. 49, 50, 80).

[ASFC15] The Apache Software Foundation. *Cassandra*. 2015. URL: http://cassandra.apache.org/ (zitiert auf S. 48, 80).

[ASFC16a] The Apache Software Foundation. *Cassandra Query Language (CQL) v3.4.0: Data Control*. 2016. URL: https://cassandra.apache.org/doc/cql3/CQL-3.0.html#dataControl (zitiert auf S. 48, 49).

[ASFC16b] The Apache Software Foundation. *Cassandra Query Language (CQL) v3.4.0: JSON Support*. 2016. URL: https://cassandra.apache.org/doc/cql3/CQL-3.0.html#json (zitiert auf S. 49).

[ASFCO16a] The Apache Software Foundation. *10.4.1. /db/doc: Attachments*. 2016. URL: http://docs.couchdb.org/en/1.6.1/api/document/common.html#attachments (zitiert auf S. 45).

[ASFCO16b] The Apache Software Foundation. *1.6. Security: 1.6.1. Authentication*. 2016. URL: http://docs.couchdb.org/en/1.6.1/intro/security.html#authentication (zitiert auf S. 45).

[ASFCO16c] The Apache Software Foundation. *1.6. Security: 1.6.3. Authorization*. 2016. URL: http://docs.couchdb.org/en/1.6.1/intro/security.html#authorization (zitiert auf S. 45, 46).

[ASFCO16d] The Apache Software Foundation. *6.1. Design Functions: 6.1.5. Filter functions*. 2016. URL: http://docs.couchdb.org/en/1.6.1/couchapp/ddocs.html#filter-functions (zitiert auf S. 46).

[ASFCO16e] The Apache Software Foundation. *A Database for the Web*. 2016. URL: http://couchdb.apache.org/ (zitiert auf S. 45, 79).

[ASFH16a] The Apache Software Foundation. *Hadoop in Secure Mode: Authentication*. 2016. URL: https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-common/SecureMode.html#Authentication (zitiert auf S. 56).

[ASFH16b] The Apache Software Foundation. *HDFS Architecture: Data Replication*. 2016. URL: https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html#Data_Replication (zitiert auf S. 57).

[ASFH16c] The Apache Software Foundation. *HDFS High Availability*. 2016. URL: https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithNFS.html (zitiert auf S. 56).

[ASFH16d] The Apache Software Foundation. *HDFS Permissions Guide*. 2016. URL: https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HdfsPermissionsGuide.html (zitiert auf S. 56, 57).

[ASFH16e] The Apache Software Foundation. *HDFS Users Guide*. 2016. URL: https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html (zitiert auf S. 56–58).

[ASFH16f] The Apache Software Foundation. *MapReduce Tutorial: Overview*. 2016. URL: https://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Overview (zitiert auf S. 57, 58).

[ASFH16g] The Apache Software Foundation. *Welcome to Apache™ Hadoop®!* 2016. URL: http://hadoop.apache.org/ (zitiert auf S. 42, 55, 56).

[ASFJ16] The Apache Software Foundation. *Welcome to Apache Jackrabbit.* 2016. URL: http://jackrabbit.apache.org/ (zitiert auf S. 79).

[ASFK16] The Apache Software Foundation. *Apache Kudu.* 2016. URL: http://kudu.apache.org/ (zitiert auf S. 80).

[ASFS16] The Apache Software Foundation. *APACHE Spark Lightning-fast cluster computing.* 2016. URL: http://spark.apache.org/ (zitiert auf S. 48).

[ASFT15a] The Apache Software Foundation. *MapReduce.* 2015. URL: http://tinkerpop.apache.org/docs/3.0.1-incubating/#mapreduce (zitiert auf S. 53).

[ASFT15b] The Apache Software Foundation. *Security.* 2015. URL: http://tinkerpop.incubator.apache.org/docs/3.0.1-incubating/#_security (zitiert auf S. 53).

[AWS16a] Amazon Web Services, Inc. *Amazon DynamoDB.* 2016. URL: https://aws.amazon.com/de/dynamodb/ (zitiert auf S. 80).

[AWS16b] Amazon Web Services, Inc. *Amazon Elastic File System.* 2016. URL: https://aws.amazon.com/efs/ (zitiert auf S. 42).

[AWS16c] Amazon Web Services, Inc. *Amazon SimpleDB für einfach Datenbanken.* 2016. URL: https://aws.amazon.com/de/simpledb/ (zitiert auf S. 80).

[Ba14] J. Bartel, et al. *Big-Data-Technologien – Wissen für Entscheider.* 2014 (zitiert auf S. 13, 30).

[Bad16] A. Bader. „Comparison of Time Series Databases“. Diplomarbeit. Universität Stuttgart, 2016 (zitiert auf S. 15).

[BB05] K. Baku, H. Beus. *CAQ-Einsatz in der Medizintechnik.* 2005 (zitiert auf S. 22).

[BB15] H. Brüggemann, P. Bremer. *Grundlagen Qualitätsmanagement.* Springer Vieweg, 2015, S. 218–223 (zitiert auf S. 16–20).

[BBS16] BS Bucher Systemlösungen GmbH & Co.KG. *Archivierungslösungen.* 2016. URL: http://www.busys.de/Loesungen/Archivierungsloesungen (zitiert auf S. 43).

[BP97] J. Baader, M. Philipp. „Rechtliche Grundlagen für den Einsatz betrieblicher elektronischer Archivierungssysteme“. In: *Datenbanken in Büro, Technik und Wissenschaft* (1997), S. 299–311 (zitiert auf S. 28).

[BTR16] Basho Technologies, Inc. *Welcome to the Basho Docs.* 2016. URL: http://docs.basho.com/ (zitiert auf S. 81).

[Bus12] S. Bushik. *A Vendor-independent Comparison of NoSQL Databases: Cassandra, HBase, MongoDB, Riak.* Techn. Ber. 2012 (zitiert auf S. 65).

[BXT15] BaseX Team. *BaseX. The XML Database.* 2015. URL: http://basex.org/ (zitiert auf S. 79).

[CBS16] COUCHBASE. *Couchbase Server*. 2016. URL: http://www.couchbase.com/nosql-databases/couchbase-server (zitiert auf S. 79).

[CER12] CERN. *Processing: What to record?* Aug. 2012. URL: http://home.cern/about/computing/processing-what-record (zitiert auf S. 13).

[Chu09] S. Chu. *MemcacheDB*. 2009. URL: http://memcachedb.org/ (zitiert auf S. 80).

[CIB16] CIB software GmbH. *CIB doXima*. 2016. URL: https://www.cib.de/de/produkte/cib-office-module-comod/cib-doxima.html (zitiert auf S. 43).

[Cin16] Cinchapi Inc. *ConcourseDB*. 2016. URL: http://concoursedb.com/ (zitiert auf S. 80).

[Cit15] Citus Data. *Architecture*. 2015. URL: https://www.citusdata.com/docs/citus/5.1/aboutcitus/introduction_to_citus.html (zitiert auf S. 55).

[Cor16] Cornell University. *HyperDex*. 2016. URL: http://hyperdex.org/ (zitiert auf S. 80).

[CP16] Clusterpoint. *Clusterpoint*. 2016. URL: https://www.clusterpoint.com/ (zitiert auf S. 79).

[DC09] S. Dekorte, R. Collins. *VERTEXDB*. 2009. URL: http://www.dekorte.com/projects/opensource/vertexdb/ (zitiert auf S. 81).

[DD16] DynomiteDB.com. *DynomiteDB*. 2016. URL: http://www.dynomitedb.com/ (zitiert auf S. 80).

[DDB16] djondb. *djonDB*. 2016. URL: http://djondb.com/ (zitiert auf S. 79).

[DG16] J. Dean, S. Ghemawat. *LevelDB*. 2016. URL: http://leveldb.org/ (zitiert auf S. 80).

[Dij59] E. W. Dijkstra. „A note on two problems in connexion with graphs“. In: *Numerische Mathematik* 2 (1959), S. 269–271 (zitiert auf S. 40).

[DSC16a] DataStax, Inc. *Apache Cassandra™ 2.1 Documentation*. Apr. 2016, S. 8 (zitiert auf S. 48).

[DSC16b] DataStax, Inc. *Apache Cassandra™ 2.1 Documentation*. Apr. 2016, S. 65–68 (zitiert auf S. 48).

[DSC16c] DataStax, Inc. *CQL for Cassandra 2.0 & 2.1: Blob type*. 2016. URL: http://docs.datastax.com/en/cql/3.1/cql/cql_reference/blob_r.html (zitiert auf S. 48).

[DST16a] DataStax, Inc. *Chapter 1. The Benefits of Titan*. 2016. URL: http://s3.thinkaurelius.com/docs/titan/1.0.0/benefits.html (zitiert auf S. 52).

[DST16b] DataStax, Inc. *Chapter 15. Cassandra*. 2016. URL: http://s3.thinkaurelius.com/docs/titan/1.0.0/cassandra.html (zitiert auf S. 53).

[DST16c] DataStax, Inc. *Chapter 16. HBase*. 2016. URL: http://s3.thinkaurelius.com/docs/titan/1.0.0/hbase.html (zitiert auf S. 53).

[DST16d] DataStax, Inc. *Chapter 32. The Benefits of Titan-Hadoop*. 2016. URL: http://s3.thinkaurelius.com/docs/titan/0.5.0/hadoop-benefits.html (zitiert auf S. 53).

[DST16e] DataStax, Inc. *Chapter 5. Schema and Data Modeling*. 2016. URL: http://s3.thinkaurelius.com/docs/titan/1.0.0/schema.html (zitiert auf S. 53).

[DST16f] DataStax, Inc. *Chapter 6. Gremlin Query Language*. 2016. URL: http://s3.thinkaurelius.com/docs/titan/1.0.0/schema.html (zitiert auf S. 53).

[DST16g] DataStax, Inc. *Chapter 7. Titan Server*. 2016. URL: http://s3.thinkaurelius.com/docs/titan/1.0.0/server.html (zitiert auf S. 53).

[DST16h] DataStax, Inc. *TITAN*. 2016. URL: http://titan.thinkaurelius.com/ (zitiert auf S. 52, 81).

[ED09] R. Eckl, R. e. a. Dr. Vahrmann. *Leitfaden Archivierung von Dokumenten*. Fachverband Electronic Components and Systems im ZVEI, Apr. 2009 (zitiert auf S. 13).

[Edl16] S. Edlich. *Your Ultimate Guide to the Non-Relational Universe!* 2016. URL: http://nosql-database.org/ (zitiert auf S. 79).

[Erd16] H. Erd. *pickleDB*. 2016. URL: https://pythonhosted.org/pickleDB/ (zitiert auf S. 80).

[ES16a] EASY SOFTWARE AG. *EASY ARCHIVE*. 2016. URL: https://easy.de/ecm-modules/easy-archive/ (zitiert auf S. 43, 44, 61, 62).

[ES16b] EASY SOFTWARE AG. *Nie wieder proprietär: (Rechts-)sichere Archivierung durch HSM-Systeme*. 2016. URL: https://easy.de/2015/01/05/nie-wieder-proprietaer-rechts-sichere-archivierung-durch-hsm-systeme/ (zitiert auf S. 62).

[ES16c] EASY SOFTWARE AG. *SICHER IST SICHER*. 2016. URL: https://easy.de/digitale-geschaeftsprozesse/archivierung/ (zitiert auf S. 61).

[ES16d] Elasticsearch. *Elasticsearch | Search & Analyze Data in Real Time*. 2016. URL: https://www.elastic.co/products/elasticsearch (zitiert auf S. 79).

[EXS16] eXist Solutions. *Vitamins for your Applications*. 2016. URL: http://exist-db.org/ (zitiert auf S. 79).

[FBU+99] A. Fox, E. A. Brewer, S. University, U. of California, at Berkeley. „Harvest, Yield, and Scalable Tolerant Systems“. In: *HOTOS '99 Proceedings of the The Seventh Workshop on Hot Topics in Operating Systems*. 1999 (zitiert auf S. 19).

[FCC16] FairCom Corporation. *Introducing c-treeACE V11*. 2016. URL: https://www.faircom.com/products/c-treeace (zitiert auf S. 80).

[FDET16] Facebook Database Engineering Team. *RocksDB*. 2016. URL: http://rocksdb.org/ (zitiert auf S. 81).

[FIA16] Franz Inc. *AllegroGraph*. 2016. URL: http://franz.com/agraph/allegrograph/ (zitiert auf S. 81).

[FPW93] G. Festl, H. Pott, K. Winzig. „Archivierung und Sicherheit“. In: *Infodoc, Heft 2* (1993). Zitiert nach "Rechtliche Grundlagen für den Einsatz betrieblicher elektronischer Archivierungssysteme"von Baader, J. and Philipp, M. aus 1997, S. 16–23 (zitiert auf S. 27, 28).

[Gar10] L. Garber. *Will NoSQL Databases Live Up to Their Promise?* 2010 (zitiert auf S. 39, 40).

[GC15] G2 Crowd, Inc. *G2 Crowd publishes Fall 2015 rankings of the best NoSQL databases, based on user reviews*. Aug. 2015. URL: https://www.g2crowd.com/press-release/best-nosql-databases-fall-2015/ (zitiert auf S. 40–42).

[Gei07] W. Geiger. „Terminus Technicus: Qualitätsdaten“. In: *QZ Qualität und Zuverlässigkeit 52* 11 (2007). Zitiert nach "Grundlagen Qualitätsmanagement"von Brüggemann, H. und Bremer, P. aus 2015, S. 12 (zitiert auf S. 16).

[Ger08] A. Gerber. *Methode zur Konzeption eines unternehmensspezifischen Qualitätsinformationssystems*. 2008, S. 4 (zitiert auf S. 16, 19).

[GHF16] GitHub, Inc. *FlockDB*. 2016. URL: https://github.com/twitter/flockdb (zitiert auf S. 81).

[GHG16] GitHub, Inc. *Feature*. 2016. URL: https://github.com/gluster/glusterfs-specs/blob/master/done/GlusterFS%203.7/Small%20File%20Performance.md (zitiert auf S. 59).

[GHN16a] GitHub, Inc. *Bolt*. 2016. URL: https://github.com/boltdb/bolt (zitiert auf S. 80).

[GHN16b] GitHub, Inc. *NeDB*. 2016. URL: https://github.com/louischatriot/nedb (zitiert auf S. 79).

[GHN16c] GitHub, Inc. *nessDB*. 2016. URL: https://github.com/BohuTANG/nessDB (zitiert auf S. 80).

[GHN16d] GitHub, Inc. *SisoDb*. 2016. URL: https://github.com/danielwertheim/sisodb-provider (zitiert auf S. 80).

[GHT16] GitHub, Inc. *thrudb - document oriented database services*. 2016. URL: https://github.com/tjake/thrudb-java (zitiert auf S. 80).

[Glü12] M. Prof. Dr. Glück. *Die Produktion 2020*. 2012 (zitiert auf S. 21).

[GOO03] Google Inc. *The Google File System*. 2003 (zitiert auf S. 42).

[GOO16] Google Inc. *terrastore*. 2016. URL: https://code.google.com/archive/p/terrastore/ (zitiert auf S. 80).

[Gra16] GraphBase. *GraphBase*. 2016. URL: http://graphbase.net/ (zitiert auf S. 81).

[Gru04] *Elektronische Archivierung von Buchungsbelegen in Kommunalkassen*. 2004 (zitiert auf S. 16).

[Gun16] Gun’s team. *gun*. 2016. URL: http://gun.js.org/ (zitiert auf S. 79, 81).

[HD15] Hibari developers. *Hibari DB*. 2015. URL: http://hibari.readthedocs.io/ (zitiert auf S. 80).

[Hen11] R. Henricsson. „Document Oriented NoSQL Databases“. Bachelorarbeit. Blekinge Institute of Technology, Juni 2011 (zitiert auf S. 65).

[Hey12] B. Heynemann. *r^3 redistrredis reduce reuse*. 2012. URL: https://heynemann.github.io/r3/ (zitiert auf S. 52).

[HHGJ11] J. Han, E. Haihong, L. Guan, D. Jian. „Survey on NoSQL Database“. In: *6th International Conference on Pervasive Computing and Applications (ICPCA)* (2011), S. 363–366 (zitiert auf S. 21, 51, 54).

[HJ11] R. Hecht, S. Jablonski. *NoSQL Evaluation A Use Case Oriented Survey*. 2011 (zitiert auf S. 21, 40).

[Höl14] T. Höllwart. *Cloud Migration*. MITP-Verlags GmbH & Co. KG, 2014, S. 175–181 (zitiert auf S. 15, 16).

[HPJS08] T. Hannus, O. K. Poignée, V. Jahn, G. Schiefer. „IT - Unterstützung im Qualitätsmanagement – Status Quo und Entwicklungsmöglichkeiten“. In: *Unternehmens-IT: Führungsinstrument oder Verwaltungsbürde?* (2008), S. 59–62 (zitiert auf S. 19, 22).

[HR01] T. Härder, E. Rahm. *Datenbanksysteme : Konzepte und Techniken der Implementierung ; mit 14 Tabellen*. Berlin; Heidelberg; New York; Barcelona; Hongkong; London; Mailand; Paris; Tokio: Springer, 2001 (zitiert auf S. 15).

[HRR15] Hibernating Rhinos. *RAVENDB*. 2015. URL: https://ravendb.net/ (zitiert auf S. 80).

[HTH14] Hypertable Inc. *HYPERTABLE*. 2014. URL: http://www.hypertable.com/ (zitiert auf S. 80).

[IBM06] IBM. *Content Manager Implementation and Migration Cookbook*. IBM Redbooks, Apr. 2006 (zitiert auf S. 63, 64).

[IBM16] IBM Deutschland GmbH. *Content Manager Enterprise Edition*. 2016. URL: http://www-03.ibm.com/software/products/de/mpenterprise (zitiert auf S. 43, 44, 63, 64).

[IMG16] IMS Messsysteme GmbH. *Qualitätsmanagementsystem*. 2016. URL: http://www.ims-gmbh.de/produktkatalog/qualitaetsmanagementsystem/ (zitiert auf S. 43).

[Ini14] Initiative Industrie 4.0: Electrosuisse. „Der Industrie 4.0 fehlt die gemeinsame Sprache“. In: *Technische Rundschau* (Apr. 2014), S. 96–98 (zitiert auf S. 13).

[IQL16] IQLECT. *BangDB*. 2016. URL: http://bangdb.com/ (zitiert auf S. 80).

[JF15] G. Jäger, G. Fuhr. „Oracle Coherence: In-Memory-Computing für Einsteiger“. In: *DOAGNews* (2015), S. 41–46 (zitiert auf S. 26).

[Kam03] U. Dr. Kampffmeyer. *GDPdU und elektronische Archivierung*. 2003 (zitiert auf S. 15, 16, 19).

[Kam12] U. Dr. Kampffmeyer. *Elektronische Archivierung Eine "never ending Story"?* 2012 (zitiert auf S. 15).

[Kau11] O. B. Kaul. *Kurz & Gut Skalierbarkeit*. Nov. 2011 (zitiert auf S. 26).

[KBSQ98] W. Klöden, S. Böhlmann, C. Scheiber, H. Quendt. „Erfassung und Auswertung von Prozeß- und Qualitätsdaten in der Lebensmittelindustrie“. In: *Chemie Ingenieur Technik (70)* (1998), S. 734–737 (zitiert auf S. 22).

[Ker96] H. Kernler. *PPS-Controlling*. Wiesbaden: Gabler Verlag, 1996, S. 7–14. ISBN: 978-3-322-89358-1 (zitiert auf S. 16).

[Kie10] C. Kiederer. *Entwicklung von Qualitätsmanagementstrategien in eine KMU*. Diplomica Verlag, 2010, S. 124 (zitiert auf S. 16).

[Kin13] K. Kingsbury. *Jepsen: Redis*. 2013. URL: https://aphyr.com/posts/283-jepsen-redis (zitiert auf S. 51).

[KPM15] KPMG AG. *Mit Daten Werte schaffen*. 2015 (zitiert auf S. 30, 31).

[Kro10] T. Kropp. *Aufbewahrungspflichten für Gewerbetreibende*. Sep. 2010 (zitiert auf S. 13).

[KSH10] Kobrix Software. *What Is It?* 2010. URL: http://www.hypergraphdb.org/ (zitiert auf S. 81).

[LIV16] LinkedIn. *Project Voldemort*. 2016. URL: http://www.project-voldemort.com/ (zitiert auf S. 81).

[LL13] J. Ludewig, H. Lichter. *Software Engineering*. dpunkt.verlag, 2013 (zitiert auf S. 13, 34).

[LM13] Y. Li, S. Manoharan. „A performance comparison of SQL and NoSQL databases“. In: *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)* (2013) (zitiert auf S. 65).

[MCD10] Microsoft Corporation. *Trinity*. 2010. URL: https://www.microsoft.com/en-us/research/project/trinity/ (zitiert auf S. 81).

[MCD16] Microsoft Corporation. *DeNSo DB*. 2016. URL: https://densodb.codeplex.com/ (zitiert auf S. 79).

[MCR16] Microsoft Corporation. *RaptorDB - The Document Store*. 2016. URL: https://raptordb.codeplex.com/ (zitiert auf S. 80, 81).

[Mit13] B. Mitschang. *Datenbanken und Informationssysteme. Vorlesungsfolien*. 2013 (zitiert auf S. 15).

[MLC16] MarkLogic Corporation. *MarkLogic*. 2016. URL: http://de.marklogic.com/ (zitiert auf S. 79).

[MMG16] MPDV Mikrolab GmbH. *Fertigungsprüfung mit HYDRA-FEP*. 2016. URL: http://www.mpdv.com/de/produkte-loesungen/standardloesung-hydra-mes/fertigungspruefung-caq (zitiert auf S. 43).

[MMSW07] M. Maged, J. E. Moreira, D. Shiloach, R. W. Wisniewski. *Scale-up x Scale-out: A Case Study using Nutch/Lucene*. Techn. Ber. März 2007 (zitiert auf S. 26).

[Mon15a] MonetDB B.V. *monetdb*. 2015. URL: https://www.monetdb.org/ (zitiert auf S. 80).

[Mon15b] MongoDB, Inc. *Performance Testing MongoDB 3.0*. 2015. URL: https://www.mongodb.com/blog/post/performance-testing-mongodb-30-part-1-throughput-improvements-measured-ycsb (zitiert auf S. 65).

[Mon16a] MongoDB, Inc. *Authentication*. 2016. URL: https://docs.mongodb.com/manual/core/authentication/ (zitiert auf S. 46).

[Mon16b] MongoDB, Inc. *Built-In Roles*. 2016. URL: https://docs.mongodb.com/manual/reference/built-in-roles/ (zitiert auf S. 47).

[Mon16c] MongoDB, Inc. *GridFS*. 2016. URL: https://docs.mongodb.com/manual/core/gridfs/ (zitiert auf S. 47).

[Mon16d] MongoDB, Inc. *JSON and BSON*. 2016. URL: https://www.mongodb.com/json-and-bson (zitiert auf S. 46).

[Mon16e] MongoDB, Inc. *MongoDB*. 2016. URL: https://www.mongodb.com/de (zitiert auf S. 46, 47, 79).

[Mon16f] MongoDB, Inc. *Query Documents*. 2016. URL: https://docs.mongodb.com/manual/tutorial/query-documents/ (zitiert auf S. 47).

[Mon16g] MongoDB, Inc. *Role-Based Access Control*. 2016. URL: https://docs.mongodb.com/manual/core/authorization/ (zitiert auf S. 46).

[Moo06] G. E. Moore. „Cramming more components onto integrated circuits“. In: *IEEE Solid-State Circuits Society Newsletter* (2006), S. 33–35 (zitiert auf S. 13, 26).

[Mos16] Mostosi, Andrea. *The Big-Data Ecosystem Table*. 2016. URL: http://bigdata.andreamostosi.name/ (zitiert auf S. 42).

[MRG16] Mail.Ru Group. *TARANTOOL*. 2016. URL: http://tarantool.org/ (zitiert auf S. 81).

[MSS15] MailStore Software GmbH. *Rechtssichere E-Mail- Archivierung Der Leitfaden für Deutschland*. 2015 (zitiert auf S. 43).

[MSS16] MailStore Software GmbH. *MailStore Server®*. 2016. URL: http://www.mailstore.com/de/mailstore-server.aspx (zitiert auf S. 43).

[NCD14] Nokia Corporation & The Disco Project. *Disco Distributed Filesystem*. 2014. URL: http://disco.readthedocs.io/en/latest/howto/ddfs.html (zitiert auf S. 42).

[NMI16] NetMesh Inc. *The Web Graph Database*. 2016. URL: http://infogrid.org/ (zitiert auf S. 81).

[NSG+06] *Integration von Qualitätsdaten and für Produktionsanlagen*. 2006 (zitiert auf S. 16).

[NTI16] Neo Technology, Inc. *neo4j*. 2016. URL: https://neo4j.com/ (zitiert auf S. 81).

[OBI16] Objectivity Inc. *INFINITEGRAPH*. 2016. URL: http://www.objectivity.com/products/infinitegraph/ (zitiert auf S. 81).

[OBT15] Oberasoftware. *jasdb*. 2015. URL: http://www.oberasoftware.com/jasdb/ (zitiert auf S. 79).

[OCB16] Oracle Corporation. *Oracle Berkeley DB 12c*. 2016. URL: http://www.oracle.com/technetwork/database/database-technologies/berkeleydb (zitiert auf S. 80).

[OCO16] Oracle Corporation. *Oracle NoSQL Database*. 2016. URL: https://www.oracle.com/database/nosql (zitiert auf S. 80).

[ODO16] OrientDB LTD. *OrientDB*. 2016. URL: http://orientdb.com/ (zitiert auf S. 80, 81).

[OTC16] OpenText Corp. *Livelink*. 2016. URL: http://www.opentext.de/was-wir-tun/produkte/opentext-produktangebotskatalog/produkte-mit-neuer-markenbezeichnung/livelink-gehoert-jetzt-zur-opentext-content-suite (zitiert auf S. 43).

[PB16] Palenga, Manuel und Bosch-Betreuer. *Besprechung am 04.05.2016*. 2016 (zitiert auf S. 71).

[PD16] Palenga, Manuel und Bosch-Datenbankbeauftragter. *Besprechung am 16.05.2016*. 2016 (zitiert auf S. 24, 25, 33).

[PGD16a] The PostgreSQL Global Development Group. *19.3. Authentication Methods*. 2016. URL: https://www.postgresql.org/docs/9.5/static/auth-methods.html (zitiert auf S. 54).

[PGD16b] The PostgreSQL Global Development Group. *25.1. Comparison of Different Solutions*. 2016. URL: https://www.postgresql.org/docs/9.5/static/different-replication-solutions.html (zitiert auf S. 55).

[PGD16c] The PostgreSQL Global Development Group. *31.7. Storing Binary Data*. 2016. URL: https://www.postgresql.org/docs/7.4/static/jdbc-binary-data.html (zitiert auf S. 54).

[PGD16d] The PostgreSQL Global Development Group. *5.6. Privileges*. 2016. URL: https://www.postgresql.org/docs/9.5/static/ddl-priv.html (zitiert auf S. 54).

[PGD16e] The PostgreSQL Global Development Group. *8.14. JSON Types*. 2016. URL: https://www.postgresql.org/docs/9.5/static/datatype-json.html (zitiert auf S. 55).

[PGD16f] The PostgreSQL Global Development Group. *9.15. JSON Functions and Operators*. 2016. URL: https://www.postgresql.org/docs/9.5/static/functions-json.html (zitiert auf S. 55).

[PGD16g] The PostgreSQL Global Development Group. *Chapter 32. Large Objects*. 2016. URL: https://www.postgresql.org/docs/current/static/largeobjects.html (zitiert auf S. 54).

[PGD16h] The PostgreSQL Global Development Group. *PostgreSQL*. 2016. URL: https://www.postgresql.org/ (zitiert auf S. 54).

[PGD16i] The PostgreSQL Global Development Group. *What is PostgreSQL?* 2016. URL: https://www.postgresql.org/docs/9.5/static/intro-whatis.html (zitiert auf S. 54).

[QDG16] Q-DAS GmbH. *Q-DBM Qualitätsdatenbank Management*. 2016. URL: http://www.q-das.de/de/camera/q-dbm/ (zitiert auf S. 43).

[QDQ16] quasardb. *quasardb*. 2016. URL: https://www.quasardb.net/ (zitiert auf S. 80).

[QS16] QuinStreet Inc. *Top Five NoSQL Databases and When to Use Them*. 2016. URL: http://www.itbusinessedge.com/slideshows/top-five-nosql-databases-and-when-to-use-them-03.html (zitiert auf S. 40–42).

[Rau16] H. Rauch. *Fallen-8*. 2016. URL: http://www.fallen-8.com/ (zitiert auf S. 81).

[RH16] Red Hat, Inc. *CEPH STORAGE*. 2016. URL: http://ceph.com/ceph-storage/file-system/ (zitiert auf S. 42).

[RHG16a] Red Hat, Inc. *Architecture*. 2016. URL: http://gluster.readthedocs.io/en/latest/Quick-Start-Guide/Architecture/ (zitiert auf S. 60).

[RHG16b] Red Hat, Inc. *Managing GlusterFS Volumes*. 2016. URL: http://gluster.readthedocs.io/en/latest/Administrator%20Guide/Managing%20Volumes/ (zitiert auf S. 59, 60).

[RHG16c] Red Hat, Inc. *Managing Hadoop Compatible Storage*. 2016. URL: http://gluster.readthedocs.io/en/latest/Administrator%20Guide/Hadoop/ (zitiert auf S. 60).

[RHG16d] Red Hat, Inc. *Quick Start Guide*. 2016. URL: http://gluster.readthedocs.io/en/latest/Install-Guide/Quick_start/ (zitiert auf S. 59, 60).

[RHG16e] Red Hat, Inc. *Setting up GlusterFS with SSL/TLS*. 2016. URL: https://gluster.readthedocs.io/en/latest/Administrator%20Guide/SSL/ (zitiert auf S. 59).

[RHG16f] Red Hat, Inc. *Storage for your Cloud*. 2016. URL: https://www.gluster.org/ (zitiert auf S. 42, 59).

[RHG16g] Red Hat, Inc. *WORM (Write Once Read Many)*. 2016. URL: http://staged-gluster-docs.readthedocs.io/en/release3.7.0beta1/Features/worm/ (zitiert auf S. 60).

[Rob15a] Robert Bosch GmbH, CI/OSD. „Mitgeltendes Dokument C zur RB/GF 181 'Archivierungssysteme und Lösungen'". Bosch-internes Dokument. 2015 (zitiert auf S. 27, 28).

[Rob15b] Robert Bosch GmbH, C/QM. „Mitgeltendes Dokument A zur RB/GF 181 'Aufbewahrungsfristen für Geschäftsunterlagen'". Bosch-internes Dokument. 2015 (zitiert auf S. 30).

[RT16] RethinkDB. *RethinkDB*. 2016. URL: https://www.rethinkdb.com/ (zitiert auf S. 80).

[Rup16] C. Rupp. *upscaledb*. 2016. URL: https://upscaledb.com/ (zitiert auf S. 81).

[SAK07] B. Schwerdtfeger, T. Alt, G. Klinker. „Augmented Reality/ Mobile Vision zur aktiven Fehlervermeidung". In: *Neue Wege in der Automobillogistik* (2007) (zitiert auf S. 21).

[San16a] S. Sanfilippo. *Data types*. 2016. URL: http://redis.io/topics/data-types (zitiert auf S. 51, 52).

[San16b] S. Sanfilippo. *Partitioning: how to split data among multiple Redis instances*. 2016. URL: http://redis.io/topics/partitioning (zitiert auf S. 51).

[San16c] S. Sanfilippo. *redis*. 2016. URL: http://redis.io/ (zitiert auf S. 51, 81).

[San16d] S. Sanfilippo. *Redis Security*. 2016. URL: http://redis.io/topics/security (zitiert auf S. 51).

[San16e] S. Sanfilippo. *Replication*. 2016. URL: http://redis.io/topics/replication (zitiert auf S. 52).

[Sar14] Sarathi, Partha. *A deep dive into NoSQL: A complete list of NoSQL databases*. Juli 2014. URL: http://bigdata-madesimple.com/a-deep-dive-into-nosql-a-complete-list-of-nosql-databases/ (zitiert auf S. 79).

[Scy16] ScyllaDB. *Scylla*. 2016. URL: http://www.scylladb.com/ (zitiert auf S. 80).

[SE05] P. Stritzke, U. Eissing. „Konvergenz der Systeme im Gesundheitswesen: Paradigma für die Integration Radiologischer Bilder und Befunde in die Gesundheitsakte“. In: *Telemedizinführer Deutschland* (2005), S. 302–310 (zitiert auf S. 16).

[SHI16] SERgroup Holding International GmbH. *Die Archivierungssoftware Doxis4 für die perfekte elektronische Archivierung*. 2016. URL: http://www.ser.de/produkte-loesungen/elektronische-archivierung.html (zitiert auf S. 43, 44, 60, 61).

[SI16a] solid IT gmbh. *Berechnungssmethode der Wertungen im DB-Engines Ranking*. 2016. URL: http://db-engines.com/de/ranking_definition (zitiert auf S. 41).

[SI16b] solid IT gmbh. *DB-Engines*. 2016. URL: http://db-engines.com/ (zitiert auf S. 40, 42).

[SI16c] solid IT gmbh. *DB-Engines Ranking - Trend der Document Stores Popularität*. 2016. URL: http://db-engines.com/de/ranking_trend/document+store (zitiert auf S. 41).

[SI16d] solid IT gmbh. *DB-Engines Ranking von Graph DBMS*. 2016. URL: http://db-engines.com/de/ranking/graph+dbms (zitiert auf S. 41).

[SI16e] solid IT gmbh. *DB-Engines Ranking von Graph DBMS*. 2016. URL: http://db-engines.com/de/ranking/relational+dbms (zitiert auf S. 41).

[SK16] Synergex, Inc. *KitaroDB*. 2016. URL: http://www.kitarodb.com/ (zitiert auf S. 80).

[SMS16] Splice Machine, Inc. *First RDBMS Powered by Hadoop and Spark*. 2016. URL: http://www.splicemachine.com/ (zitiert auf S. 80).

[SSS16] STS Soft SC. *STSSOFT*. 2016. URL: http://stssoft.com/ (zitiert auf S. 81).

[SSu16] Symisc Systems S.U.A.R.L. *An Embeddable NoSQL Database Engine*. 2016. URL: https://unqlite.org/ (zitiert auf S. 80).

[Ste14] M. Stephan. „Kollaboratives, synchrones und konsistentes Engineering von Audiodaten“. Magisterarb. Feb. 2014 (zitiert auf S. 21).

[STS16] Sparsity Technologies. *Sparsity Technologies - Powering Extreme Data*. 2016. URL: http://www.sparsity-technologies.com/ (zitiert auf S. 81).

[TDc16] The Druid community. *Druid is a high-performance, column-oriented, distributed data store*. 2016. URL: http://druid.io/ (zitiert auf S. 80).

[TFB16a] ThinkParQ & Fraunhofer ITWM. *BeeGFS The Parallel Cluster File System*. 2016. URL: http://www.beegfs.com/ (zitiert auf S. 42, 58).

[TFB16b] ThinkParQ & Fraunhofer ITWM. *BeeGFS Wiki : AboutMirroring*. 2016. URL: http://www.beegfs.com/wiki/AboutMirroring (zitiert auf S. 59).

[TFB16c] ThinkParQ & Fraunhofer ITWM. *BeeGFS Wiki : AdmonLogin*. 2016. URL: http://www.beegfs.com/wiki/AdmonLogin (zitiert auf S. 58).

[TFB16d] ThinkParQ & Fraunhofer ITWM. *BeeGFS Wiki : AdmonMenu*. 2016. URL: http://www.beegfs.com/wiki/AdmonMenu (zitiert auf S. 58).

[TFB16e] ThinkParQ & Fraunhofer ITWM. *BeeGFS Wiki : FAQ*. 2016. URL: http://www.beegfs.com/wiki/FAQ (zitiert auf S. 58, 59).

[TFB16f] ThinkParQ & Fraunhofer ITWM. *BeeGFS Wiki : SystemArchitecture*. 2016. URL: http://www.beegfs.com/wiki/SystemArchitecture (zitiert auf S. 58).

[TMS16] Tec Media Services GmbH. *Suchzeiten und Verwaltungsaufwand senken mit TMS Archiv - der Cloud-basierten Archivierungslösung*. 2016. URL: http://www.tec-media-services.de/tms-archiv.html (zitiert auf S. 43).

[TP16] ThinkParQ GmbH. *BeeGFS Flyer*. 2016 (zitiert auf S. 58).

[TSA16a] FAL Labs. *Tokyo Cabinet: a modern implementation of DBM*. 2016. URL: http://fallabs.com/tokyocabinet/ (zitiert auf S. 81).

[TSA16b] TIBCO Software Inc. *TIBCO ACTIVESPACES ENTERPRISE EDITION*. 2016. URL: http://www.tibco.com/products/automation/in-memory-computing/in-memory-data-grid/activespaces-enterprise-edition (zitiert auf S. 81).

[VOI09] VOI - Verband Organisations- und Informationssysteme e. V. *Merksätze des VOI zur revisionssicheren elektronischen Archivierung*. 2009 (zitiert auf S. 15, 27–29).

[Wag14] S. Wagner. *Einführung in die Softwaretechnik. Vorlesungsfolien*. 2014 (zitiert auf S. 34).

[Whi09a] T. White. *Hadoop The Definitive Guide*. Hrsg. von M. Loukides. OReilly Media, Juni 2009, S. 42 (zitiert auf S. 57).

[Whi09b] T. White. *Hadoop The Definitive Guide*. Hrsg. von M. Loukides. OReilly Media, Juni 2009, S. 71–73 (zitiert auf S. 57).

[Whi09c] T. White. *Hadoop The Definitive Guide*. Hrsg. von M. Loukides. OReilly Media, Juni 2009, S. 190–191 (zitiert auf S. 57).

[WPG15] Dr. Wieselhuber & Partner GmbH, Fraunhofer-Institut für Produktionstechnik und Automatisierung. *Geschäftsmodell-Innovation durch Industrie 4.0*. Dr. Wieselhuber & Partner GmbH, März 2015, S. 14 (zitiert auf S. 13).

[WTW13] WhiteDB team. *Pure speed*. 2013. URL: http://whitedb.org/ (zitiert auf S. 81).

[ZIB16] Zuse Institute Berlin. *What is Scalaris?* 2016. URL: http://scalaris.zib.de/ (zitiert auf S. 81).

Alle URLs wurden zuletzt am 05. 09. 2016 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift