

BACHELOR THESIS
Zulkhizhat Ismailova

Bewertung von SQL- und NoSQL-Datenbanksystemen: Eine Nutzwertanalyse zur Entscheidungsfindung

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

Zulkhizhat Ismailova

Bewertung von SQL- und NoSQL-Datenbanksystemen: Eine Nutzwertanalyse zur Entscheidungsfindung

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Wirtschaftsinformatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Stefan Sarstedt
Zweitgutachter: Prof. Dr. Ulrike Steffens

Eingereicht am: 03. März 2025

Zulkhizhat Ismailova

Thema der Arbeit

Bewertung von SQL- und NoSQL-Datenbanksystemen: Eine Nutzwertanalyse zur Entscheidungsfindung

Stichworte

SQL, NoSQL, Big Data, Nutzwertanalyse

Kurzzusammenfassung

Daten werden mit jedem Tag wichtiger und gewinnen vor allem in der Wirtschaft an Bedeutung. Aus diesem Grund ist es entscheidend, wie die Daten gespeichert werden. Derzeit gehören SQL- und NoSQL-Datenbanken zu den am häufigsten eingesetzten Datenbanksystemen. Die Frage ist jedoch wie entscheidet man welches Datenbankmanagementsystem am besten zu dem Anwendungsfall und den daraus folgenden Daten passt? Die vorliegende Arbeit untersucht welche Kriterien bei der Entscheidungsfindung möglich sind und welche tatsächlich auch eine Rolle in der realen Welt spielen. Zudem werden Performance Tests durchgeführt, um die Leistungsstärke der jeweiligen Datenbanksysteme zu bewerten. Alle gemachten Erkenntnisse werden in einer Nutzwertanalyse zusammengefasst und ausgewertet. Das Ergebnis ist hiermit eine Unterstützung zu der Wahl zwischen SQL- und NoSQL für zwei Anwendungsfälle - Banking Daten sowie Daten aus E-Commerce.

Zulkhizhat Ismailova

Title of Thesis

Evaluation of SQL and NoSQL Database Systems: A Utility Value Analysis for Decision-Making

Keywords

SQL, NoSQL, Big Data, Utility Value Analysis

Abstract

Data is becoming increasingly important every day and is gaining particular significance in the business world. For this reason, it is crucial to determine how data is stored. Currently, SQL and NoSQL databases are among the most commonly used database systems. However, the question arises: how does one decide which database management system is best suited for a specific use case and its associated data requirements? This study explores the criteria that can influence decision-making and identifies those that are truly relevant in real-world scenarios. Additionally, performance tests are conducted to evaluate the efficiency of each database system. All findings are summarized and analyzed in a utility value analysis. The result provides guidance for choosing between SQL and NoSQL databases for two specific use cases: banking data and e-commerce data.

Inhaltsverzeichnis

Abbildungsverzeichnis	viii
Tabellenverzeichnis	ix
1 Einleitung	1
1.1 Problemstellung	2
1.2 Zielsetzung	2
1.3 Vorgehensweise und Struktur der Arbeit	3
2 Theoretische Grundlagen	4
2.1 Daten und Datenbanktechnologien	4
2.1.1 SQL Datenbanksysteme	4
2.1.2 NoSQL Datenbanksysteme	5
2.1.3 Big Data	6
2.2 Transaktionsprinzipien	6
2.2.1 ACID	8
2.2.2 BASE	9
2.2.3 CAP-Theorem	9
2.2.4 ACID vs. BASE	10
2.3 Informationssicherheit	11
2.3.1 CIA-Triade	11
3 Vergleichskriterien	13
3.1 Kriterien	13
3.1.1 Datenstruktur und Komplexität	14
3.1.2 Konsistenz, Verfügbarkeit und Partitionstoleranz	14
3.1.3 Leistung und Skalierbarkeit	15
3.1.4 Datenvolumen und -vielfalt	15
3.1.5 Sicherheitsfunktionen	16

3.1.6	Datenwiederherstellung und Backup	16
3.1.7	Datenverteilung und Lokalisierung	16
3.1.8	Echtzeitverarbeitung	17
3.1.9	Kosten, Lizenzierung und Ökosystem	17
3.2	Zusammenfassung	18
4	Anforderungsanalyse	21
4.1	Nutzergruppen	21
4.1.1	Anwender	22
4.1.2	IT-Operations	23
4.2	Methodisches Vorgehen	25
4.2.1	Interview vs. Workshop	25
4.3	Ergebnisse der Interviews	26
4.3.1	Anwender - Person A	26
4.3.2	IT-Operations - Person B	28
4.3.3	IT-Operations - Person C	29
4.3.4	Ergebnisse und Theorie	30
4.4	User Storys	31
4.4.1	Anwender	31
4.4.2	IT-Operations	32
5	Vergleichende Analyse	33
5.1	Funktionaler Vergleich	34
5.1.1	Datenstruktur	34
5.1.2	Datenintegrität & Transaktionen	35
5.1.3	Sicherheit	36
5.2	Nicht funktionaler Vergleich	39
5.2.1	Leistung	39
5.2.2	Datencharakteristika	40
5.2.3	Datensicherung	41
5.2.4	Datenverteilung	41
5.2.5	Verarbeitungsanforderungen	41
5.2.6	Wirtschaftliche Aspekte	42
5.3	Performance-Analyse	45
5.3.1	Testumgebung	45
5.3.2	Testsznarien	45

5.3.3	Test Set-up	46
5.3.4	Ergebnisse	47
5.3.5	Fazit	56
6	Nutzwertanalyse	57
6.1	Schritt 1: Kriterien	57
6.2	Schritt 2: Gewichtung	58
6.3	Schritt 3: Bewertung	59
6.4	Banking Nutzwertanalyse	59
6.5	E-Commerce Nutzwertanalyse	60
7	Fazit und Ausblick	61
7.1	Zusammenfassung	61
7.2	Fazit	62
7.3	Ausblick	62
	Literaturverzeichnis	63
A	Anhang	66
A.1	Performance-Test Ergebnisse: Banking	66
A.2	Performance-Test Ergebnisse: E-Commerce	67
A.3	CD Anhang	67
	Selbstständigkeitserklärung	69

Abbildungsverzeichnis

2.1	Relationale Tabellenstruktur (Kleuker, 2023)	5
2.2	Drei unterschiedliche NoSQL-Datenbanken (Kaufmann und Meier, 2023)	5
2.3	Vielfalt der Quellen bei Big Data (Kaufmann und Meier, 2023, S. 12)	6
2.4	Mögliche Optionen des CAP-Theorems (Kaufmann und Meier, 2023)	10
2.5	Wichtigste Unterschiede zwischen ACID und BASE (Kaufmann und Meier, 2023)	11
2.6	Maßnahmen für die Ziele der CIA-Triade (Kaufmann und Meier, 2023)	12
3.1	Datenbanken in Datenbanksystemen (Kleuker, 2023)	14
4.1	Nutzergruppenaufteilung	22
5.1	Durchsatz bei Simple & Complex Read Tests (Banking Daten)	48
5.2	Latenz bei Simple & Complex Read Tests (Banking Daten)	48
5.3	Durchsatz bei Simple & Batch Insert Tests (Banking Daten)	49
5.4	Durchsatz bei Simple & Complex Update Tests (Banking Daten)	50
5.5	Durchsatz bei Simple & Complex Update Tests (Banking Daten)	50
5.6	Durchsatz bei Simple & Batch Delete Tests (Banking Daten)	51
5.7	Latenz bei Simple & Batch Delete Tests (Banking Daten)	52
5.8	Durchsatz bei Simple & Complex Read Tests (E-Commerce Daten)	53
5.9	Durchsatz bei Simple & Batch Insert Tests (E-Commerce Daten)	53
5.10	Durchsatz bei Simple & Complex Update Tests (E-Commerce Daten)	54
5.11	Latenz bei Simple & Complex Update Tests (E-Commerce Daten)	54
5.12	Durchsatz bei Simple & Batch Delete Tests (E-Commerce Daten)	55
5.13	Latenz bei Simple & Batch Delete Tests (E-Commerce Daten)	55
6.1	Nutzwertanalyse Banking	59
6.2	Nutzwertanalyse E-Commerce	60

Tabellenverzeichnis

3.1	Vergleichskriterien Teil 1	19
3.2	Vergleichskriterien Teil 2	20
5.1	Funktionale Vergleichskriterien	38
5.2	Nicht funktionale Vergleichskriterien	44
5.3	Latenz bei Simple & Batch Insert Tests (Banking Daten)	49
A.1	Performance-Test Ergebnisse: Banking	66
A.2	Performance-Test Ergebnisse: E-Commerce	67

1 Einleitung

In den vergangenen Jahren hat die Bedeutung von Daten in der Geschäftswelt signifikant zugenommen. Besonders durch die rasante Entwicklung von Technologien wie Künstlicher Intelligenz erkennen Unternehmen zunehmend, wie entscheidend Daten und deren korrekte Speicherung für den geschäftlichen Erfolg sind. Die Auswahl der geeigneten Datenbanktechnologie spielt dabei eine zentrale Rolle. Im Fokus stehen SQL (Structured Query Language) und NoSQL (Not only SQL) Datenbanken, die jeweils unterschiedliche Stärken und Schwächen aufweisen. Die aktuelle IT-Landschaft ist geprägt von traditionellen relationalen SQL-Datenbanksystemen und den flexiblen NoSQL-Systemen.

SQL-Datenbanken, wie MySQL und Microsoft SQL Server, gelten seit Jahrzehnten als Standard für die Speicherung und Abfrage von strukturierten Daten. Sie zeichnen sich durch eine starke Kompetenz in der Relationen Algebra aus und folgen dem ACID-Prinzip, das für Atomarität, Konsistenz, Isolation und Dauerhaftigkeit steht. Diese Systeme bieten eine klare Struktur und ermöglichen komplexe Abfragen.

Im Gegensatz dazu bieten NoSQL-Datenbanken eine Vielzahl von Datenmodellen, darunter dokumentenorientierte (wie MongoDB), Key-Value-Stores (wie Redis), spaltenorientierte (wie Cassandra) und graphenbasierte Modelle (wie Neo4j). Sie sind bekannt für ihre effiziente Handhabung unstrukturierter Daten, schnelle Verarbeitung und hohe Skalierbarkeit.

Angesichts der spezifischen Stärken und Schwächen jedes Datenbanktyps gibt es zahlreiche Kriterien für die Auswahl einer geeigneten Datenbanktechnologie. Diese Entscheidung hängt stark vom jeweiligen Anwendungsfall ab, was die Komplexität der Auswahl weiter erhöht.

1.1 Problemstellung

Die Wahl eines geeigneten Datenbanksystems (DBS) ist von kritischer Bedeutung für viele Unternehmen, unabhängig von der Größe und Branche. Diese Entscheidung hat weitreichende Auswirkungen auf viele Geschäftsprozesse, zum Beispiel durch den Einfluss der Leistung des Datenbanksystems. So kann ein Online-Shop Kunden verlieren, wenn sie zu lange darauf warten müssen, bis die Webseite lädt, weil das Datenbanksystem überlastet ist.

Die Wahl zwischen SQL und NoSQL Datenbanksystemen ist oft eine komplexe Aufgabe, da verschiedene Anwendungsfälle und Nutzer unterschiedliche Anforderungen an die Datenverarbeitung stellen. Zudem gibt es angesichts der spezifischen Stärken und Schwächen jedes Datenbanksystems eine Vielzahl von Kriterien, auf die bei der Entscheidungsfindung einbezogen werden sollten.

Da es bisher an standardisierten und objektiven Methoden fehlt, um SQL und NoSQL Datenbanken umfassend zu vergleichen, stehen Entscheidungsträger oft vor großen Herausforderungen. Es entstehen Entscheidungsunsicherheiten, erhöhte Verwaltungsanforderungen und potenziell ineffiziente Ressourcennutzung in Unternehmen, die die Wahl des optimalen Datenbanksystems erschweren. Um die Entscheidung zu stützen, wird in dieser Arbeit mit der in der Betriebswirtschaftslehre oft eingesetzten Methode der Nutzwertanalyse gearbeitet.

1.2 Zielsetzung

Im Rahmen dieser Arbeit soll eine Nutzwertanalyse konzipiert werden, die die wichtigsten Kriterien für zwei spezifische Anwendungsfälle betrachtet. Diese Kriterien werden in zwei Etappen gesammelt: zuerst in einer theoretischen Recherche und dann durch Interviews mit Datenbanknutzern, die die Wichtigkeit der Kriterien bestätigen oder ablehnen. Somit soll die Nutzwertanalyse Bewertungskriterien enthalten, die tatsächlich relevant sind. Die Auswertung der Nutzwertanalyse soll zur Entscheidungsfindung beitragen. Es soll ein klares Ergebnis zu sehen sein, das kaum bis gar keinen Raum für Zweifel lässt.

1.3 Vorgehensweise und Struktur der Arbeit

Wie bereits oben erwähnt, werden zunächst Vergleichskriterien durch theoretische Recherche erarbeitet. Diese werden in Form von Fragen in die Anforderungserhebung integriert, welche mit Interviews durchgeführt wird. Somit sollte klar werden, welche Kriterien für die Nutzer tatsächlich von Bedeutung sind. Auf dieser Grundlage erfolgt ein Vergleich der beiden ausgewählten SQL- und NoSQL-Datenbanken. Zusätzlich werden beide Datenbanken einer Performance-Analyse unterzogen, bei der die CRUD-Operationen hinsichtlich Latenz und Durchsatz getestet werden.

Abschließend wird eine Nutzwertanalyse erstellt, wobei die Bewertung und Gewichtung der Kriterien unter anderem auf den Ergebnissen des Vergleichs basieren.

2 Theoretische Grundlagen

2.1 Daten und Datenbanktechnologien

Daten – im Jahre 2010 waren es nur zwei Zettabytes, 14 Jahre später 149 und 2028 sollen es 349 Zettabytes werden (Statista, 2024). Daten sind heute überall und begleiten uns in jedem Moment unseres Lebens, sei es beim Online-Shopping, E-Mails schreiben oder bei einem Arztbesuch (vgl. Wiese, 2015). Genauso schnell wie sich das Datenvolumen erhöht, so entwickelt sich auch die Welt der Datenbanktechnologien. Das weltweit erste integrierte Datenbanksystem sah die Welt im Jahre 1960 und schon 10 Jahre später wurde die Idee der ersten relationalen Datenbank konzipiert. Ein Durchbruch für die Zeit – effizient und praktisch. Seit nun 50 Jahren sind SQL-Datenbanken der Favorit unter vielen Anwendungsentwicklern und Unternehmen. Doch mit der Zeit werden Datensätze unstrukturierter und Daten aus dem Internet präsenter (vgl. Foote, 2023). Somit entstehen weitere Arten von Datenbanksystemen, die dafür gedacht sind, besser mit den Daten von morgen umzugehen. So wurde 2009 erstmals der Begriff „NoSQL“ eingeführt, hierzu gehören beispielsweise dokumentenorientierte, Key-Value- sowie Graphen-Datenbanken.

2.1.1 SQL Datenbanksysteme

In diesem Abschnitt wird ein kurzer Steckbrief zu SQL-Datenbanksystemen gegeben, ohne dabei tiefgehend auf die Punkte einzugehen.

- Relationales Datenmodell basierend auf der relationalen Algebra
- Datenspeicherung: Tabellen
- Daten können miteinander verknüpft sein (wie in Abbildung 2.1)
- Datenbanksprache: SQL
- Häufig zentralisiert

Artikel		
Artikelnummer	Bezeichnung	Anzahl
12345	Schnuller	42
12346	Latzhose	25
12347	Teddy	6

Artikeleinkauf		Firma	
Artikelnummer	FirmaNr	FirmaNr	Firma
12345	1	1	Babys Unlimited
12346	2	2	Cool Clothes AG
12347	1		

Abbildung 2.1: Relationale Tabellenstruktur (Kleuker, 2023)

2.1.2 NoSQL Datenbanksysteme

In diesem Abschnitt der Arbeit wird wie oben ein Steckbrief zu NoSQL-Datenbanksystemen gegeben.

- Datenbanksprache: nicht SQL (MQL, Cypher etc.)
- Datenspeicherung: Dokumenten, Key-Value Stores etc. (siehe Abbildung 2.2)
- Erfüllt alle Anforderungen an eine Big Data Anwendung
- Verteilte Replikate und parallele Ausführung (vgl. Fasel, 2016, S. 12)

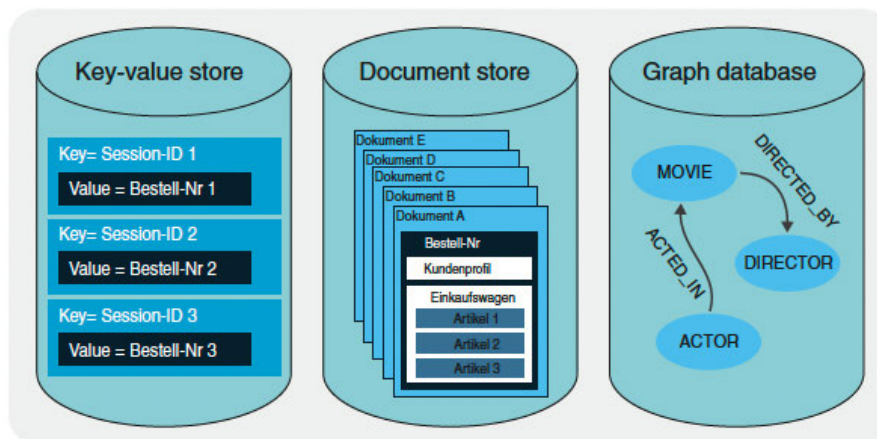


Abbildung 2.2: Drei unterschiedliche NoSQL-Datenbanken (Kaufmann und Meier, 2023)

2.1.3 Big Data

„The future of business is big data.“ (vgl. Ghavami, 2021)

Big Data ist ein Begriff, der bereits seit Jahren, vor allem im Kontext von Datenbanken, Gegenstand von Diskussionen ist. Erfüllt eine Datenmenge die folgenden drei V-Eigenschaften, kann man diese zu Big Data zählen (vgl. Ghavami, 2021).

- Volume: große Datenmenge
- Velocity: Echtzeitverarbeitung
- Variety: unterschiedliche Daten (bzw. Datenformate)

Handelt es sich also um eine große unstrukturierte Datenmenge, welche schnell verarbeitet werden muss, so handelt es sich um Big Data. In Abbildung 2.3 ist zu sehen, dass die Vielfalt der Daten sehr breit aufgestellt ist. So kann man hier z. B. für soziale Netzwerke typische Daten wiederfinden, aber auch medizinische Befunde wie „Röntgenbilder“.

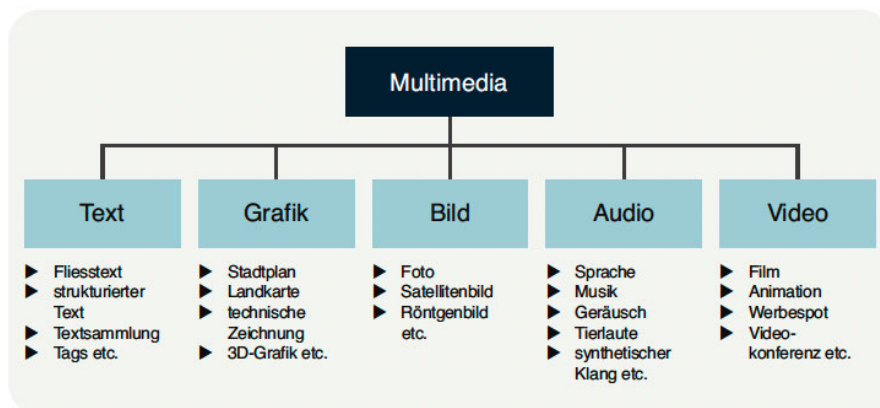


Abbildung 2.3: Vielfalt der Quellen bei Big Data (Kaufmann und Meier, 2023, S. 12)

2.2 Transaktionsprinzipien

Unabhängig davon, wo und wie eine Datenbank eingesetzt wird, wird erwartet, dass nach jeder Änderung, die durch den Menschen, eine Anwendung oder Ausfall verursacht wurde, die Daten korrekt bleiben. Wie genau das Datenbanksystem die Konsistenz der Daten

beibehält, hängt von dem Anwendungsfall sowie den funktionalen und nicht funktionalen Eigenschaften des DBS ab. Zwei Konzepte können aber hier zur Hilfe gezogen werden, um an die Anforderung nach Konsistenz strukturierter heranzugehen.

„Eine Transaktion ist eine Konsistenz erhaltende Operation auf einer Datenbank.“ (Ghavami, 2021, S. 17).

2.2.1 ACID

Die Verarbeitung von Daten erfolgt üblicherweise durch mehrere Operationen, die in einer Transaktion zusammengefasst sind. Viele Geschäftsbereiche und Anwendungsfälle erfordern die Konsistenz der Daten. Um die Konsistenz sicherzustellen, kann unter anderem ein klassisches Transaktionsprinzip wie ACID eingesetzt werden. Das Befolgen von ACID soll die Konsistenz einer Datenbankoperation, auch Transaktion genannt, gewährleisten, indem folgende Anforderungen erfüllt werden (vgl. Rahm u. a., 2015).

Atomicity (Atomarität): Die Transaktion muss entweder vollständig und zu Ende ausgeführt werden oder sie ändert nichts an dem Datenbankzustand (Meier, 2017). Ist eine Transaktion fehlerhaft, so wird diese vollständig zurückgesetzt. Das Datenbanksystem sorgt hiermit dafür, dass der Programmierer von einer fehlerfreien Umgebung ausgehen kann und dadurch die Realisierung der Anwendung vereinfacht wird (Rahm u. a., 2015).

Consistency (Konsistenz): Jede Transaktion transformiert die Datenbank von einem konsistenten Zustand in einen konsistenten Zustand (Meier, 2017). Endet eine Transaktion, so müssen alle Konsistenzbedingungen erfüllt sein, was die Widerspruchsfreiheit der Daten garantiert (Kaufmann und Meier, 2023).

Isolation (Isolation): Datenbanken werden in der Regel von mehreren Benutzern genutzt. Deshalb finden oft mehrere Transaktionen parallel statt. Damit es zu keinen Fehlern in der Bearbeitung der Daten kommt, müssen Transaktionen seriell, nach einander ausgeführt werden. Es folgt also dasselbe Ergebnis wie, wenn es sich um einen Einbenutzerbetrieb handeln würde, was ebenso der Vereinfachung in der Programmierung beiträgt (Rahm u. a., 2015).

Durability (Dauerhaftigkeit): Nach jeder durchgeführten Transaktion wird der Zustand der Datenbank verändert. Dieser Zustand bleibt bestehen, bis eine andere Transaktion ausgeführt und der Zustand somit aktualisiert wird. Sollte es zum Beispiel zu Fehlern im Programm oder Systemabstürzen gekommen sein, so bleibt der Zustand der zuletzt korrekt durchgeführten Transaktion bewahrt (Kaufmann und Meier, 2023).

2.2.2 BASE

Oft wird, vor allem bei Webanwendungen, eine hohe Verfügbarkeit angestrebt. Das heißt, dass bei Ausfall eines Rechnerknotens mit dem replizierten Rechnerknoten weitergearbeitet werden kann. Dabei muss der Anwender keine Einschränkungen in der Arbeit der Anwendung wahrnehmen (Kaufmann und Meier, 2023). In vielen verteilten Systemen ist es jedoch nicht einfach, alle Forderungen des ACID-Prinzips zu erreichen. Vor allem Atomarität, Konsistenz und Isolation sind nur mit Verfahren erreichbar, die schlecht skalieren oder ausfallempfindlich sind (Rahm u. a., 2015). In diesen Fällen wird einer Konsistenzforderung gefolgt, die es zulässt, dass replizierte Knoten vorübergehend unterschiedliche Datenversionen haben können und „irgendwann“ (engl. „eventually“) synchronisiert werden (Kaufmann und Meier, 2023). Das BASE Konzept wird wie folgt definiert (vgl. Rahm u. a., 2015, S. 354):

Basically Available: Die Datenbank muss immer erreichbar sein. So müssen Daten eines Onlinehandels zum Beispiel immer erreichbar sein, damit Kunden auf die Webanwendung durchgehend Zugriff haben.

Soft State: Da der Zustand einer globalen Konsistenz in vielen Fällen nicht immer notwendig ist, können sich Datenbanksysteme auch im „Soft State“ befinden. Dies bedeutet, dass kurzzeitige Inkonsistenzen bei replizierten Datenbeständen und Antworten, die nicht zu hundert Prozent aktuell sind, erlaubt werden.

Eventually Consistent: Da von den Clients „irgendwann“ ein konsistenter Zustand erreicht wird, ist eine strikte Konsistenz nicht immer notwendig. Das heißt, dass einzelne Clients zwischenzeitlich lediglich Zugriff auf veraltete Daten haben.

2.2.3 CAP-Theorem

Laut Eric Brewer der Universität Berkeley können Konsistenz, Verfügbarkeit und Ausfalltoleranz in einem massiv verteilten Rechnersystem nicht gleichzeitig gelten. Aus dieser Idee entstand das CAP-Theorem, das aussagt, dass zur selben Zeit nur zwei dieser drei Eigenschaften ein System beschreiben können (vgl. Kaufmann und Meier, 2023):

Consistency (Konsistenz): Werden nach einer Transaktion Daten auf einem der replizierten Knoten verändert, so erhalten alle nachfolgenden lesenden Transaktionen den aktuellen Zustand, unabhängig davon, über welchen Knoten zugegriffen wird.

Availability (Verfügbarkeit): Verfügbarkeit bedeutet, dass die Anwendung stets ohne Unterbrechung läuft und die Antwortzeiten im Rahmen bleiben.

Partition Tolerance (Verfügbarkeit): Wenn ein Server oder die Verbindung zwischen Servern in einem replizierten Netzwerk ausfällt, bleibt das Gesamtsystem weiterhin funktionsfähig. Neue Knoten können jederzeit ohne Unterbrechung des Betriebs hinzugefügt werden.

Das heißt, es können also, wie in Abbildung 2.4 zu sehen, entweder Konsistenz und Verfügbarkeit (CA) oder Konsistenz und Ausfalltoleranz (CP) oder Verfügbarkeit und Ausfalltoleranz (AP) sichergestellt werden. Wenn man aber noch bedenkt, dass das CAP-Theorem für verteilte Systeme konzipiert wurde, dann wäre die Kombination „CA“ für verteilte Datenbanksysteme irrelevant.

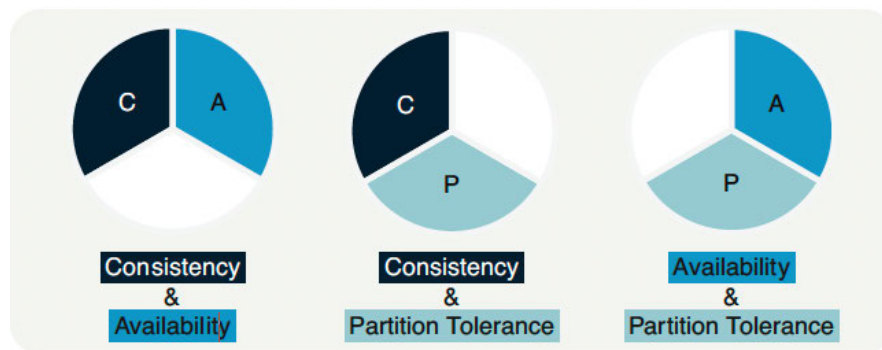


Abbildung 2.4: Mögliche Optionen des CAP-Theorems (Kaufmann und Meier, 2023)

So sind beispielsweise die Buchungssysteme internationaler Hotelketten darauf ausgelegt, dass die Verfügbarkeit und Ausfalltoleranz garantiert werden müssen. Und aufgrund des CAP-Theorems fällt die Konsistenz raus, da in diesem Fall dann lediglich das „AP“ gewährleistet werden kann. Das heißt dann aber auch, dass Buchungen nach dem BASE-Prinzip erfolgen (vgl. Kaufmann und Meier, 2023, S. 164).

2.2.4 ACID vs. BASE

Viele der SQL und NoSQL Datenbanksysteme sind ACID orientiert. Jedoch gibt es auch einige NoSQL-Datenbanken, die eher auf den BASE-Ansatz ausweichen. Der Grund dafür ist, dass Änderungen, die an einem Knoten vorgenommen werden, nicht unmittelbar auf allen anderen Knoten wirksam werden. Daher kann die Konsistenz in solchen Datenbanksystemen nur mit Verzögerung sichergestellt werden. Dementsprechend ist ein

Knoten dann meistens verfügbar und kann dabei eventuell nicht immer konsistent nachgeführt werden und befindet sich dadurch im „weichen“ Zustand (Kaufmann und Meier, 2023). Die in Abb. 2.5 aufgeführten Unterschiede folgen aus der Art und Weise, auf die Konsistenz gewährleistet wird.

ACID	BASE
Konsistenz hat oberste Priorität (strong consistency)	Konsistenz wird verzögert etabliert (weak consistency)
meistens pessimistische Synchronisationsverfahren mit Sperrprotokollen	meistens optimistische Synchronisationsverfahren mit Differenzierungsoptionen
Verfügbarkeit bei überschaubaren Datenmengen gewährleistet	hohe Verfügbarkeit resp. Ausfalltoleranz bei massiv verteilter Datenhaltung
einige Integritätsregeln sind im Datenbankschema gewährleistet (z.B. referenzielle Integrität)	einige Integritätsregeln sind im Datenbankschema gewährleistet (z.B. referenzielle Integrität)

Abbildung 2.5: Wichtigste Unterschiede zwischen ACID und BASE (Kaufmann und Meier, 2023)

2.3 Informationssicherheit

2.3.1 CIA-Triade

Da eine Datenbank oft Daten enthält, die nicht in die falschen Hände geraten oder böswillig verändert werden sollten, spielt die Informationssicherheit eine essenzielle Rolle. Um die Datensicherheit zu gewährleisten, befolgt man die CIA-Triade (Kaufmann und Meier, 2023):

Confidentiality: Schutz der Privatsphäre durch Verhinderung unbefugten Zugriffs auf Daten.

Integrity: Gewährleistung der Zuverlässigkeit der Informationen.

Availability: Sicherstellung, dass das Informationssystem funktionsfähig bleibt und jederzeit zugänglich ist.

Um die CIA-Ziele zu erreichen, müssen neben der physischen Gewährleistung der Datensicherheit folgende Maßnahmen umgesetzt werden:

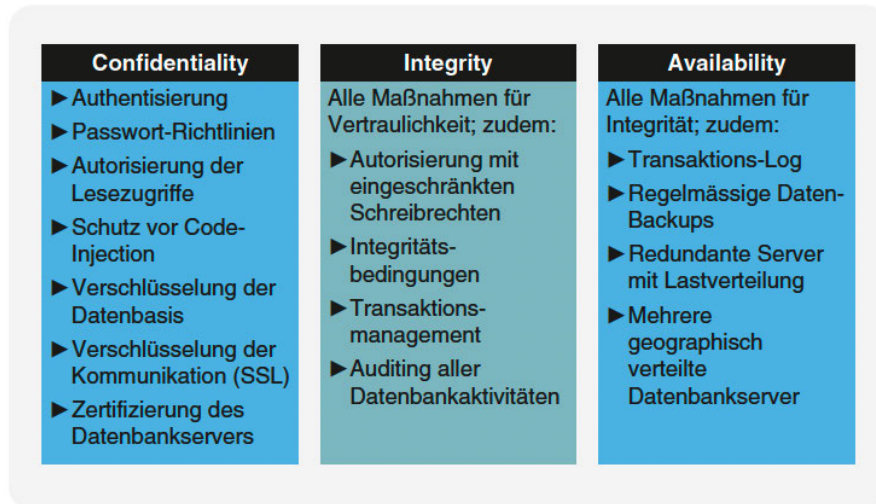


Abbildung 2.6: Maßnahmen für die Ziele der CIA-Triade (Kaufmann und Meier, 2023)

3 Vergleichskriterien

In diesem Kapitel wird eine Liste von möglichen Vergleichskriterien vorgestellt, die auf einer umfassenden Recherche basiert. Diese Kriterien werden in den nachfolgenden Kapiteln, insbesondere in der Nutzwertanalyse, eine zentrale Rolle spielen. Die endgültigen Bewertungskriterien könnten sich jedoch leicht von den hier aufgeführten unterscheiden, da es sich um eine erste Version handelt. Im weiteren Verlauf der Arbeit werden diese Kriterien überarbeitet und angepasst, um sie zu optimieren und zu präzisieren.

3.1 Kriterien

Die Vergleichskriterien, die zur Bewertung der verschiedenen Datenbankarten verwendet werden, entsprechen im Wesentlichen den Anforderungen, die an eine Datenbank gestellt werden. So sind die zentralen Anforderungen an eine Datenbank, die aus einem Softwaresystem eine Datenbank machen, laut (Kleuker, 2023, S. 2-4) folgende:

1. Persistenz, das heißt Daten sollen auf der Datenbank gespeichert werden können und sollten wiederverwendbar sein.
2. Es sollte das Anlegen von Datenschemata möglich sein, damit die Daten formal beschrieben werden können.
3. Dem Nutzer soll ermöglicht werden, Daten einzufügen, zu ändern und zu löschen.
4. Es sollte gewährleistet sein, dass die in der Datenbank gespeicherten Daten jederzeit abgerufen werden können.

Wie in Abbildung 3.1 dargestellt ist, wird eine Datenbank dann aber noch in ein Datenbankmanagementsystem (DBMS) eingebettet, was zu zusätzlichen Anforderungen führt. Dadurch entstehen Funktionalitäten, die dem Nutzer eine uneingeschränkte Nutzung der Datenbank ermöglichen (vgl. Kleuker, 2023). Genau diese Funktionalitäten sind es, die

den Unterschied zwischen Datenbank-Managementsystemen ausmachen und auf denen der Fokus im folgenden Abschnitt liegt.

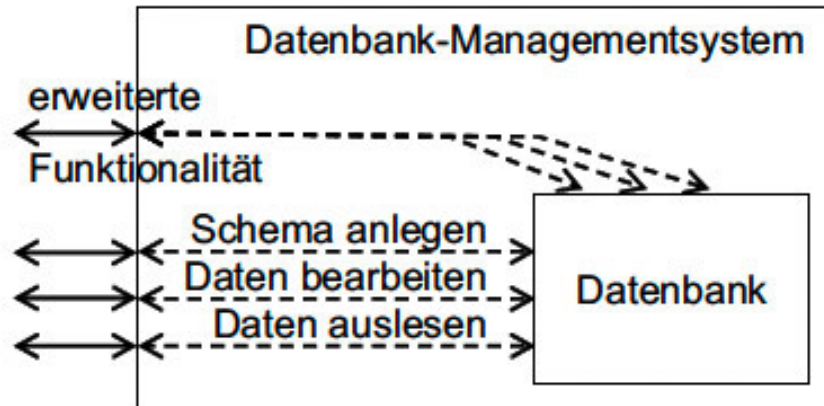


Abbildung 3.1: Datenbanken in Datenbanksystemen (Kleuker, 2023)

3.1.1 Datenstruktur und Komplexität

Beim Auswahlprozess einer Datenbank sollte die Art der zu speichernden Daten im Vordergrund stehen. Daten können sich in ihrer Struktur stark unterscheiden und als strukturiert, semistrukturiert oder unstrukturiert klassifiziert werden. Strukturierte Daten erfordern in der Regel eine klare und feste Struktur, wie sie beispielsweise in Tabellenform realisiert wird, was typischerweise durch SQL-Datenbanken unterstützt wird (vgl. Meier, 2010). Wenn jedoch eine gewisse Schemaflexibilität im Datenmodell erforderlich ist, können NoSQL-Datenbanken eine geeignete Alternative darstellen, da sie zum Beispiel die agile Softwareentwicklung fördern können (vgl. Klettke u. a., 2016).

3.1.2 Konsistenz, Verfügbarkeit und Partitionstoleranz

Damit mehrere Nutzer zur gleichen Zeit an einer Datenbank arbeiten können, muss für die Transaktionsverwaltung gesorgt werden:

„Transaktionsverwaltungen garantieren, dass konsistente Datenbankzustände immer in konsistente Datenbankzustände überführt werden. Ein Transaktionssystem arbeitet nach dem Alles-oder-nichts-Prinzip. Es wird damit ausgeschlossen, dass Transaktionen nur

teilweise Änderungen auf der Datenbank ausführen. Entweder werden alle gewünschten Änderungen ausgeführt oder keine Wirkung auf der Datenbank erzeugt.“ (Kaufmann und Meier, 2023, S. 150)

Hier kommt das ACID-Prinzip, das viele relationale Datenbanken einsetzen, ins Spiel. Eine Datenbank, die ACID vollständig einsetzt, unterstützt laut dem CAP-Theorem die Konsistenz (C) und muss deshalb zwischen Verfügbarkeit (A) und Partitionstoleranz (P) abwägen. Dies ist aber nicht immer nötig, wie bei größeren Webanwendungen, die eher auf Verfügbarkeit und Ausfalltoleranz angewiesen sind (vgl. Kaufmann und Meier, 2023). Aus diesem Grund muss im Idealfall noch vor der Auswahl der Datenbank entschieden werden, wo die Prioritäten liegen. So kann man statt ACID auf BASE setzen, was eine etwas weichere Konsistenzforderung erlaubt, wenn es in Ordnung ist, dass replizierte Knoten mit einer Verzögerung synchronisiert werden.

3.1.3 Leistung und Skalierbarkeit

Lese- und Schreibfunktionen machen aus einem System eine Datenbank (vgl. Kleuker, 2023), weshalb es wichtig ist, diese Anforderung noch einmal genauer zu berücksichtigen. Die Lese- und Schreibfunktionen einer Datenbank können je nach Anwendungsfall unterschiedliche Wichtigkeit bedeuten. Handelt es sich um eine IoT-Anwendung, die mit Zeitreihendaten aus Sensoren arbeitet, sollten starke Schreib- und Leseleistungen erwartet werden. Dies bedeutet wiederum, dass die Anwendung auf eine optimierte Skalierbarkeit der Datenbank angewiesen ist.

„Herkömmliche Datenbanken sind für diese Skalierbarkeit nicht ausgelegt. ZDBS bieten höchsten Schreibdurchsatz, schnellere Abfragen bei Skalierung und bessere Datenkomprimierung.“ (Kaufmann und Meier, 2023)

3.1.4 Datenvolumen und -vielfalt

In der Regel kann man noch vor dem Aufsetzen der Datenbank einschätzen, wie groß das Datenvolumen sein wird und ob Veränderungen in der Art der Daten zu erwarten sind. So können Daten aus einem ERP-System eines mittelständischen Unternehmens ganz andere Anforderungen mit sich bringen als Daten aus einer IoT-Anwendung. Hier ist wieder die Schreib- und Lesegeschwindigkeit sowie Skalierbarkeit von entscheidender Bedeutung. Zudem sollte man auch Daten, die als „Big Data“ klassifiziert werden, bewusst

unterscheiden, da diese der Definition nach so vielfältig und umfangreich sind, dass eine hohe Verarbeitungsgeschwindigkeit und Echtzeitverarbeitung vorausgesetzt sind. Dieses Kriterium gilt als Erweiterung des vorherigen Kriteriums „Leistung und Skalierbarkeit“, da es sich hier um dieselben Funktionen und Eigenschaften handelt.

3.1.5 Sicherheitsfunktionen

„Sobald ein Datenbanksystem in Betrieb genommen wird, ist es Risiken ausgesetzt. Zum Beispiel können Daten verloren gehen, ungewollt oder böswillig verändert oder preisgegeben werden.“ (Kaufmann und Meier, 2023)

Daher ist es wichtig, den Sicherheitsaspekt zu berücksichtigen. Dabei stellt sich die Frage, wie die Nutzer in ihrer Fähigkeit eingeschränkt werden können, die Daten zu sehen und zu bearbeiten. Zudem kann in dem Vergleich die CIA-Triade in Betracht gezogen werden.

3.1.6 Datenwiederherstellung und Backup

Beim Einsatz einer Datenbank ist mit möglichen Fehlern in verschiedenen Bereichen zu rechnen, darunter die Anwendung, die Datenbanksoftware oder die Hardware. Ohne angemessene Mechanismen zur Wiederherstellung könnten Daten verloren gehen oder inkonsistent werden (vgl. Kaufmann und Meier, 2023). Da SQL und NoSQL Datenbanksysteme unterschiedliche Ansätze nutzen, ergibt es Sinn, auch dieses Kriterium aufzunehmen. Die wesentlichen Unterschiede lassen sich wieder auf die Art der Daten sowie das CAP-Theorem zurückführen.

3.1.7 Datenverteilung und Lokalisierung

Bevor die Entscheidung fällt, welche Datenbank in Betrieb genommen wird, sollte genau analysiert werden, wo die Daten benötigt werden, welche Anforderungen an die Geschwindigkeit und Verfügbarkeit bestehen und welche rechtlichen und Compliance-Bedingungen zu beachten sind.

Für hochskalierbare Big Data Anwendungen mit einer großen Datenbank und hohem Durchsatz von Schreib- und Leseanfragen kommen DBMS infrage, die nach dem Sharding Prinzip arbeiten (vgl. Rahm u. a., 2015). Sharding ist auch bei geografisch verteilten

Knoten von Vorteil und sorgt für bessere Performance. Zudem sind die gesetzlichen Vorgaben von Land zu Land unterschiedlich. So kann gefordert werden, dass bestimmte Daten innerhalb der Landesgrenze gespeichert werden müssen. Wird auf die Konsistenz gesetzt und kann die Skalierbarkeit benachteiligt werden, wird eine zentralisierte Speicherung bevorzugt.

3.1.8 Echtzeitverarbeitung

Es ist wichtig zu unterscheiden, ob je nach Anwendungsfall Echtzeitverarbeitungsfähigkeiten erforderlich sind und das einzusetzende DBMS kontinuierliche Datenströme verarbeiten kann. Echtzeitverarbeitung der Daten ist vor allem bei IoT-Anwendungen von großer Bedeutung und kann einen entscheidenden Faktor darstellen. Denn *„die Speicherung und Analyse von Echtzeit-Sensordaten in Zeitreihen-Datenbanken ermöglicht schnellere und präzisere Anpassungen an Infrastrukturänderungen, Energieverbrauch, Gerätewartung oder andere wichtige Entscheidungen, die sich auf ein Unternehmen auswirken.“* (Kaufmann und Meier, 2023).

Echtzeitverarbeitung ist jedoch nicht nur bei IoT-Daten relevant. Sie kann auch bei der Analyse von Kundendaten oder der Überwachung von Schlüsselmetriken in Business-Intelligence-Anwendungen eingesetzt werden.

3.1.9 Kosten, Lizenzierung und Ökosystem

Lizenzkosten, Hardware- und Betriebskosten sowie die Kosten für die weitere Wartung der Datenbank sind die Fragen, die häufig das höhere Management am meisten interessieren. Hier hat man die Wahl zwischen kommerziellen und Open-Source-Datenbanken, der vertikalen und horizontalen Skalierung sowie der unterschiedlichen Verwaltungs- und Konfigurationskostenmodelle. Dasselbe gilt auch in Bezug auf die Lizenzierung.

Das Thema Ökosystem und Community-Support ist wiederum für die direkten Nutzer der Datenbanken von Bedeutung. Hier stellt sich die Frage, zu welchen Tools das DBMS eine einfache Integration bietet und welche dieser Tools in dem jeweiligen Anwendungsfall im Einsatz sind. Ist eine Datenbank zudem schon länger auf dem Markt, so gibt es eine etablierte Support-Community, die sich im Laufe der Jahre aufgebaut hat.

3.2 Zusammenfassung

Im folgenden Abschnitt sind die aufgeführten Kriterien in einer Tabelle zusammengefasst. Die Tabelle beinhaltet zudem Fragen, die gestellt werden sollten, um zu verstehen, welche Anforderungen der Anwendungsfall mit sich bringt. Und ebenso Fragen zu der Datenbank, diese werden in einem der folgenden Kapitel im Vergleich der Datenbanken wieder aufgegriffen.

3 Vergleichskriterien

Nr.	Kategorie	Kriterium	Frage zum Anwendungsfall	Frage zur Datenbank
1	Datenstruktur	Strukturierung der Daten	Wie sind die zu speichernden Daten strukturiert? Wird eine feste Struktur oder Schemaflexibilität benötigt?	Unterstützt die Datenbank flexible Schemata oder erfordert sie ein festes Schema?
2.1	Datenintegrität	Konsistenz	Wie wichtig ist die sofortige Konsistenz der Daten?	Bietet die Datenbank starke Konsistenzgarantien oder eventual consistency?
2.2		Verfügbarkeit	Welche Priorität hat die ständige Verfügbarkeit des Systems?	Wie gewährleistet die Datenbank hohe Verfügbarkeit?
2.3		Partitions-toleranz	Wie wichtig ist die Fähigkeit, Netzwerkpartitionen zu tolerieren?	Wie geht die Datenbank mit Netzwerkpartitionen um?
3.1	Leistung	Lese- und Schreibgeschwindigkeit	Welche Anforderungen bestehen an die Lese- und Schreibgeschwindigkeit?	Wie schnell kann die Datenbank Lese- und Schreiboperationen durchführen?
3.2		Skalierbarkeit	Wie wichtig ist die Fähigkeit, mit wachsendem Datenvolumen zu skalieren?	Wie wird die Datenbank in der Regel skaliert?
4.1	Datencharakteristika	Datenvolumen	Mit welchem Datenvolumen wird im Anwendungsfall gerechnet?	Wie viel Daten kann die Datenbank effizient verwalten?
4.2		Datenvielfalt	Wie vielfältig sind die Daten? Fallen sie in die Kategorie "Big Data"?	Kann die Datenbank verschiedene Datentypen und -formate effizient speichern und abfragen?

Tabelle 3.1: Vergleichskriterien Teil 1

Nr.	Kategorie	Kriterium	Frage zum Anwendungsfall	Frage zur Datenbank
5	Sicherheit	Zugriffskontrollen & CIA-Triade	Welche Anforderungen bestehen an die Einschränkung von Nutzerrechten?	Welche Möglichkeiten zur Zugriffskontrolle und Authentifizierung bietet die Datenbank?
6	Datensicherung	Backup und Wiederherstellung	Welche Anforderungen bestehen an Backup-Mechanismen und Datenwiederherstellung?	Welche Backup- und Wiederherstellungsfunktionen bietet die Datenbank?
7	Datenverteilung	Geografische Verteilung	Wo werden die Daten benötigt und gespeichert; gibt es rechtliche Einschränkungen?	Unterstützt die Datenbank geografisch verteilte Datenspeicherung und -replikation?
8	Verarbeitungsanforderungen	Echtzeitverarbeitung	Werden Echtzeit-Datenverarbeitungsfähigkeiten benötigt?	Wie gut unterstützt die Datenbank Echtzeitverarbeitung und -analysen?
9.1	Wirtschaftliche Aspekte	Kosten	Welche Lizenz-, Hardware- und Betriebskosten sind tragbar?	Wie sieht das Kostenmodell der Datenbank aus (Lizenzierung, Support, Hardware-Anforderungen)?
9.2		Lizenzierung	Wird eine kommerzielle oder Open-Source-Lösung bevorzugt?	Unter welcher Lizenz wird die Datenbank angeboten?
9.3		Ökosystem und Support	Welche Integrationen und welches Community-Support werden benötigt?	Wie umfangreich ist das Ökosystem der Datenbank und welche Supportoptionen gibt es?

Tabelle 3.2: Vergleichskriterien Teil 2

4 Anforderungsanalyse

Datenbanksysteme bieten eine Vielzahl an Funktionalitäten und Möglichkeiten. Dies führt zu einer Vielzahl von möglichen Vergleichskriterien, die in dem vorherigen Kapitel dargestellt wurden. Diese Kriterien, die auf Basis einer Literaturrecherche gesammelt und erläutert wurden, sind jedoch sehr theoretisch geprägt. Daher stellt sich die Frage, ob diese Kriterien in der realen Welt tatsächlich als Anforderungen an Datenbanksysteme wiederzufinden sind. Um diese Frage zu untersuchen, wird in diesem Kapitel eine Anforderungserhebung und -analyse durchgeführt. Diese beinhaltet eine Befragung von zwei Nutzergruppen, die im Folgenden beschrieben werden. Diese umfasst eine Befragung zweier Nutzergruppen, die im Folgenden beschrieben werden. Basierend auf der Liste der Vergleichskriterien und den Ergebnissen der Interviews werden zudem User Stories erstellt, die später bei der Erstellung der Nutzwertanalyse genutzt werden. Das Ziel der Anforderungsanalyse besteht darin, die Anforderungen zu ermitteln, die Nutzer in der realen Welt an Datenbanksysteme stellen. Zudem sollen die gesammelten Vergleichskriterien nach ihrer Wichtigkeit priorisiert werden, um sie als Bewertungskriterien im Rahmen der Nutzwertanalyse zu verwenden.

4.1 Nutzergruppen

Zunächst werden die Nutzergruppen definiert, indem sie anhand ihres Aufgabenbereichs in Anwender und IT-Operations unterteilt werden. Dies vereinfacht die Abgrenzung der Interviewfragen. Die Gruppe IT-Operations wird weiter in spezifische Rollen und Aufgaben untergliedert, was zu vier Untergruppen führt (siehe Abb. 4.1). Die Nutzergruppe der Anwender bleibt als eine einheitliche Gruppe bestehen, da die spezifischen Aufgaben, die der Anwender bei der Arbeit mit der Datenbank übernimmt, keine große Rolle spielen. Im weiteren Verlauf der Arbeit wird der Anwender als „Person A“, die IT-Operationsmitarbeiter als „Person B“ und „Person C“ bezeichnet.

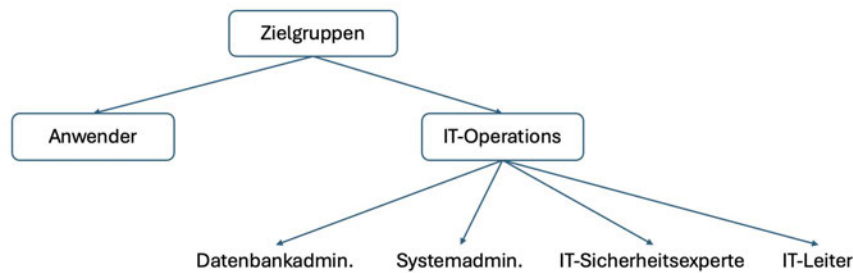


Abbildung 4.1: Nutzergruppenaufteilung

4.1.1 Anwender

Die Gruppe der Anwender umfasst Personen, die täglich auf der Benutzeroberfläche mit Datenbanksystemen arbeiten und diese für ihre Aufgaben und Anwendungsfälle nutzen. Hierzu gehören Data Analysten und Business-Intelligence-Experten, Data Engineers, Data Scientists und Softwareentwickler. Interviewt wurde ein Business-Intelligence-Experte, der gleichzeitig auch die Rolle eines Data Engineers übernimmt und somit für alles rund um Daten, deren Verarbeitung und Darstellung verantwortlich ist. Zu seinen Aufgaben zählen folgende:

- Identifikation und Bereinigung von Inkonsistenzen, Duplikaten und fehlenden Werten in Datenquellen (in diesem Fall hauptsächlich dem ERP-System Navision)
- Entwicklung und Optimierung von Datenmodellen, die sowohl Analysezwecken als auch Performance-Anforderungen entsprechen
- Aufbau und Wartung von Prozessen zur Extraktion, Transformation und Laden (ETL) von Daten in das Data Warehouse
- Automatisierung von Workflows durch Implementierung von Datenpipelines für wiederkehrende Aufgaben
- Entwicklung von Berichten zur Visualisierung und Analyse von Daten mithilfe von Microsoft Power BI, um Geschäftserkenntnisse verständlich darzustellen.
- Entwicklung komplexer SQL-Abfragen für Analysen und Aggregationen
- Leistungsoptimierung der Datenbank durch Indexing, Caching etc.

- Sicherstellung hoher Schreibgeschwindigkeiten und Anpassungsfähigkeiten des Datenbanksystems an wachsende Datenvolumen
- Zusammenarbeit mit Stakeholdern zur Ermittlung ihrer Anforderungen sowie Schulung und Support von Nutzern bei der Anwendung und Erstellung von Berichten

4.1.2 IT-Operations

Zu der Nutzergruppe der IT-Operations gehören im Rahmen dieser Arbeit die Personen, deren Aufgaben sich rund um die Entwicklung bzw. Konzipierung, Einsatz und Wartung von Datenbanksystemen richten. Hier gehören die folgenden User hinzu (vgl. Kaufmann und Meier, 2023, S. 77-78):

Datenbankadministratoren (DBA) sind für die Installation und Konfiguration von Datenbanksystemen zuständig. Sie kümmern sich auch oft um das Performance-Tuning und Optimierung der Datenbanksysteme. Zu den wichtigen Aufgaben gehört vor allem der Bereich Backup und Recovery, aber auch Sicherheitsmanagement und Zugriffskontrollen. DBA's sind zudem auch für die Kapazitätsplanung und Skalierung zuständig.

Systemadministratoren sind für die Verwaltung der IT-Infrastruktur verantwortlich, zu der auch die Systeme gehören, auf denen Datenbank-Systeme betrieben werden. Ihre Aufgaben umfassen die Verwaltung von Betriebssystemen und Servern sowie das Netzwerkmanagement und die Netzwerksicherheit. Zudem sind sie für die Allokation und Optimierung von Ressourcen zuständig.

IT-Sicherheitsexperten sind verantwortlich für den Schutz von Datenbanksystemen vor Sicherheitsbedrohungen und die Einhaltung von Compliance-Anforderungen. Typische Aufgaben für sie sind die Implementierung von Sicherheitsrichtlinien und -kontrollen, Überwachung von Sicherheitsvorfällen und die Sicherstellung der Einhaltung von Datenschutzbestimmungen.

IT-Leiter sind für die IT-Strategie und IT-Operations eines Unternehmens verantwortlich. Im Kontext der Datenbanken gehören zu ihren Aufgaben die Überwachung und Steuerung des IT-Budgets, Entscheidungsfindung bei großen IT-Investitionen und -Projekten sowie das Risikomanagement und Compliance-Sicherstellung im IT-Bereich

und Förderung von Innovation und digitaler Transformation.

Die ursprüngliche Idee bestand darin, dass jede Untergruppe der Nutzergruppe Fragen erhält, die auf ihren spezifischen Aufgabenbereich zugeschnitten sind. In der Praxis sah es jedoch anders aus. Die Interviewten für diese Nutzergruppe stammten aus derselben Firma. Schon während der Vorbereitung der Interviews wurde in einem Gespräch mit dem IT-Leiter klar, dass im Unternehmen keine klaren Rollentrennungen bestehen. Diese Erkenntnis wurde durch die Aufgabenbeschreibungen der Interviewten selbst bestätigt. So hat etwa einer der Gesprächspartner eine leitende Rolle im ERP-System-Team Navision und arbeitet intensiv mit der Datenbankadministration. Um dies genauer zu erläutern, wird im folgenden Abschnitt die Beschreibung der Aufgaben der Befragten dargestellt, basierend auf ihren eigenen Aussagen.

Person B ist seit über 20 Jahren im Unternehmen tätig und verantwortlich für die Entwicklung und Wartung des ERP-Systems Microsoft Navision. Seit 2019 leitet er das Entwicklerteam, was zusätzliche Aufgaben wie die Organisation der Teamarbeit mit sich bringt. Im Folgenden sind seine Aufgaben aufgeführt:

- **Teamleitung:** Kommunikation mit Fachabteilungen zur Aufnahme von Anforderungen und Wünschen sowie die Beratung zu neuen Funktionalitäten.
- **Entwicklung in Navision:** Programmierung und Fehlerbehebung innerhalb von Navision auf Basis einer SQL-Datenbank zur Unterstützung der Arbeitsprozesse in Bereichen wie Buchhaltung, Service- und Verkaufsaufträgen.
- **Optimierung der Datenbanknutzung:** Berücksichtigung von Performance-Aspekten bei der Programmierung (z. B. Schlüsselverwendung, Vermeidung von Verzögerungen durch Sperrungen). Dies auch mithilfe von SQL-Tools wie „SQL View“ zur Optimierung von Datenabfragen, insbesondere bei großen Datensätzen. Hierzu gehört auch die Sicherstellung, dass Anwendungen in Navision auf der Datenbank korrekt und effizient laufen.

Person C ist ein erfahrener IT-Infrastrukturberater mit 26 Jahren Erfahrung in der IT. Er ist seit über fünf Jahren als externer Berater für das Unternehmen tätig. Seine Spezialisierung liegt in IT-Infrastrukturlösungen innerhalb der Microsoft-Umgebung sowie in der Entwicklung von Lösungsansätzen für Cloud-Migrationen und hybride Szenarien. Person C hat tiefere Kenntnisse im Aufbau und Betrieb von SQL-Server-Infrastrukturen, jedoch keinen Fokus auf NoSQL. Die Aufgabenbereiche von Person C umfassen:

- **Kundenberatung:** regelmäßiger Kontakt mit IT-Leitern über bestehende und zukünftige Projekte.
- **Projektmanagement:** Leitung und Steuerung von IT-Projekten, insbesondere in Microsoft-Umgebungen. Hierzu gehört die Planung, Koordination und Überwachung von Teilprojekten in Zusammenarbeit mit Kunden und internen Teams.
- **Technische Umsetzung der Projekte:** Aufbau und Konfiguration von hochverfügbaren IT-Systemen mit Schwerpunkt auf Microsoft-Technologien in Bereichen wie Active Directory, Exchange (Online/ On-premises) und SQL Server.
- **Infrastrukturmanagement:** Wartung und Optimierung von gewachsenen IT-Strukturen. Sowie die Sicherstellung der Verfügbarkeit und Skalierbarkeit von Systemen in hybriden Umgebungen.

Person C kann aufgrund ihrer umfassenden Aufgaben und Erfahrungen als Vertreter der IT-Leitung angesehen werden und übernimmt teilweise auch Aufgaben, die einem Systemadministrator ähneln. Die Erfahrungen, die C in verschiedenen Unternehmen gesammelt hat, sind ebenfalls von großem Nutzen.

4.2 Methodisches Vorgehen

Eine Anforderungsanalyse kann mittels vieler unterschiedlicher Methodiken durchgeführt werden. Eine der möglichen Herangehensweisen sind Befragungen, die ebenso unterschiedlicher Art sein können. Für die vorliegende Analyse wurde zwischen einem Interview und einem Workshop entschieden. Diese beiden Ansätze ermöglichen durch den persönlichen Austausch mit den Befragten mehr Möglichkeiten zur Klärung und Vertiefung von Fragen (vgl. Herrmann, 2022).

4.2.1 Interview vs. Workshop

Der größte Vorteil eines Workshops ist, dass mehrere Stakeholder bzw. Vertreter der Nutzergruppen in einem Raum sitzen und über die Anforderungen diskutieren können. Somit könnten Ergänzungen und Widersprüche direkt geklärt werden (vgl. Herrmann, 2022). Vor allem in dem Kontext der Unternehmensorganisation wäre dies von Vorteil. Wie oben beschrieben, hat sich im Laufe der Anforderungserhebung erwiesen, dass die

Rollenverteilung nicht so strukturiert ist wie erwartet.

Das Problem war jedoch die Organisation des Workshops. Nach einigen Versuchen, einen Termin zu finden, der allen Beteiligten passt, wurde klar, dass dies in kürzester Zeit nicht möglich ist. Somit wurde die Entscheidung getroffen, Interviews im Eins-zu-eins-Format durchzuführen. In einem Interview können Fragen an den Stakeholder individuell angepasst werden. Vor jedem Interview hat deshalb die zu befragende Person einen Fragenkatalog mit denen für ihre Expertise passenden Fragen zugeschickt bekommen. Der Fragenkatalog wurde hierbei anhand der in dem vorherigen Kapitel erstellten Kriterienliste konzipiert, dabei wurden alle Fragen offen formuliert. Für das Interview selbst wurden einige Fragen hervorgehoben und basierend auf den Antworten weitere aus dem Katalog gestellt. Da eines der Vorteile eines Interviews die Möglichkeit der Rückfragen ist (vgl. Herrmann, 2022), wurden auch viele Fragen dieser Art gestellt. Jedes Interview wurde zeitlich auf jeweils eine Stunde begrenzt.

4.3 Ergebnisse der Interviews

Die Ergebnisse der Interviews werden in der Anforderungsanalyse zur Erstellung der User Storys beitragen. Hierfür werden im Folgenden die wichtigsten und relevanten Aussagen der zwei Nutzergruppen zusammengefasst.

4.3.1 Anwender - Person A

Erfahrungen mit SQL und NoSQL

- Der Anwender (im Weiteren „A“) arbeitet täglich mit einer Datenbank, dem Microsoft SQL Server, für Datenanalysen und Prozessoptimierungen. SQL sieht er dabei als besser geeignet für seine Arbeit, da die strukturierte Natur und vordefinierte Schemata die Datenqualität sichern und Fehler minimieren.
- NoSQL wird als weniger relevant für die Aufgaben von A eingeschätzt, da die vorhandenen Datenmengen überschaubar und zur strukturierten Verarbeitung gut geeignet sind.

- NoSQL sieht A als vorteilhaft für große, geografisch verteilte oder dynamische Daten (z. B. in IoT-Anwendungen), jedoch nicht im aktuellen Unternehmenskontext.
- Vordefinierte Schemata helfen, klare Strukturen zu schaffen und Fehler durch falsche Datentypen zu vermeiden. Und da die Unternehmensprozesse aktuell sehr strukturiert sind, werden dynamische Schemata oder unvorhersehbare Datentypen als unwahrscheinlich angesehen.
- SQL-Datenbanken sind eng mit BI-Tools wie Power BI integriert, was ihm die Arbeit als Business Intelligence Experte erleichtert.

Performance und Indexierung

- Die Einführung von Indexierung hat die Performance in SQL signifikant verbessert. In einigen Fällen wurden Abfragen bis zu 100-fach bis bei manchen komplexeren Abfragen sogar 500-fach schneller.
- Besonders bei Power BI Direct Queries sorgte die Indexierung für deutliche Geschwindigkeitsgewinne, indem Timeouts verhindert wurden.
- A vermutet, dass Indexierung bei NoSQL-Datenbanken weniger relevant ist, da die Datenbankkonzepte unterschiedlich sind.

Datenverfügbarkeit und Echtzeitverarbeitung

- Die konstante Verfügbarkeit und Zuverlässigkeit der Datenbank sind für A ein entscheidender Aspekt. Insbesondere, weil es sich erwiesen hat, dass es dem Einzelnutzer einfacher ist, an Daten in einem BI-Tool zu kommen als über ein ERP-System, deshalb ist A der Meinung, dass eine BI-Software für ein Unternehmen teilweise sogar wichtiger ist als das ERP-System. Das begründet A damit, dass das Erheben der Daten lediglich das Werkzeug zu dem, was man am Ende mit den Daten machen würde, darstellt. Deswegen hat die Verfügbarkeit der Datenbank, auf die das BI-Tool zugreift, eine enorme Wichtigkeit, sonst kann es den Endnutzer verunsichern.
- Die Echtzeitverarbeitung der Daten ist aktuell im Unternehmen nicht erforderlich, könnte jedoch in Zukunft bei IoT-Anwendungen relevant werden.

4.3.2 IT-Operations - Person B

Datenbanken in Navision

- Navision baut auf einer SQL-Datenbank auf, Geschäftsprozesse wie Verkaufsaufträge, Buchhaltungsvorgänge und andere operative Abläufe werden hiermit ermöglicht.
- NoSQL-Datenbanken hätten in Navision keinen großen Nutzen, da das ERP-System eine relationale Struktur benötigt, da die Prozesse und Strukturen auf relationale Datenbanken zugeschnitten sind.

Performance-Optimierung

- Die Performance der SQL-Datenbank wird von der Programmierung im ERP-System direkt beeinflusst. Dabei spielen Faktoren wie die Verwendung von Schlüsseln und die Reihenfolge der Datenzugriffe eine Rolle.
- Eine fehlerhafte Programmierung in Navision kann zu Verzögerungen oder gar Blockierungen von Nutzern in der Datenbank führen.
- Es werden auch Tools eingesetzt, um Indizes zu optimieren, wodurch Lesezugriffe schneller werden.

Backup, Entscheidungsfindung und Kosten

- Navision bietet integrierte Funktionen für die Datensicherung, die etwa das Sichern der gesamten Datenbank, einzelner Mandanten oder nur der Programmierungen gewährleisten.
- Die Auswahl von Navision als ERP-System basierte weniger auf der zugrundeliegenden Datenbank, sondern auf den Anforderungen an das ERP-System und den Lizenzkosten.
- Die Lizenzmodelle für SQL und ERP-Systeme beinhalten einmalige Kosten sowie Wartungsgebühren, die in die Entscheidungsfindung einfließen.

4.3.3 IT-Operations - Person C

Entscheidungsfindung bei Datenbanksystemen

- Bei der Auswahl eines Datenbanksystems wird häufig auf die Anforderungen der eingesetzten Applikation und weniger auf die dahinterliegende Datenbank geschaut.
- Budget, Supportmöglichkeiten und langfristige Wartung spielen eine zentrale Rolle.
- Hochverfügbarkeitslösungen wie Failover-Clustering und Always-on werden nach Bedürfnissen des Kunden eingesetzt.
- Monitoring-Daten erleichtern es dem Kunden bzw. dem Entscheider zu erklären, an welcher Stelle und aus welchem Grund zusätzliche Ressourcen notwendig sind, insbesondere bei steigenden Anforderungen durch neue Applikationen.
- Häufig geht es dem Kunden überwiegend darum, Lösungen zu finden, die möglichst viele Anforderungen abdecken, jedoch innerhalb des finanziellen Rahmens liegen.
- Sicherheit ist ein zentrales Thema, insbesondere in Bezug auf Datenverschlüsselung und Backups.

Cloud-Lösungen und hybride Ansätze

- Hybride Ansätze, in denen bestimmte Dienste in die Cloud ausgelagert werden, sind aktuell am weitesten verbreitet.
- Die Entscheidung wird oft pragmatisch zwischen Cloud-Only, hybriden und On-Premises-Lösungen basierend auf dem Budget, Flexibilität und bestehender Applikationen getroffen.
- Flexibilität ist eines der Vorteile einer cloudbasierten Lösung, was jedoch kostenintensiver ist und ein gewisses Maß an Know-how und Vertrauen an die Anbieter erfordert.

4.3.4 Ergebnisse und Theorie

Wenn die Ergebnisse der Interviews mit der Liste der Vergleichskriterien aus dem vorherigen Kapitel verglichen werden, zeigen sich zahlreiche Übereinstimmungen. So betonte Person A, dass die konstante Verfügbarkeit und Zuverlässigkeit der Datenbank für sie entscheidende Aspekte sind, eine Meinung, die auch Person C teilt. Zusätzlich hob Person A die Wichtigkeit der einfachen Integration von BI-Tools hervor, was teilweise das Kriterium „Ökosystem und Support“ abdeckt. Person C unterstrich ebenfalls die zentrale Rolle der Supportmöglichkeiten, was ebenfalls in dieses Kriterium fällt.

Person B wies auf ein wichtiges, bisher nicht berücksichtigtes Kriterium hin – die Art der Datenbank, die eine bereits ausgewählte Anwendung erfordert. Oftmals wird nach der Auswahl eines ERP-Systems anhand bestimmter Anforderungen nicht mehr auf die im Hintergrund verwendete Datenbank geachtet. Die Programmierung im ERP-System hat jedoch einen erheblichen Einfluss auf die Performance der Datenbank.

Zudem wird die Datensicherung durch das ERP-System beeinflusst, da die Daten in dem System durch integrierte Funktionen gesichert werden können. Dies wirft die Frage auf, wie relevant die Datensicherung im Datenbankmanagementsystem (DBMS) ist, wenn Anwendungen selbst für die Datensicherung sorgen.

A hat eine wichtige Bedeutung dem Zusammenhang zwischen Indexierung und der Performance gegeben. Laut seiner Erfahrung wurde die Datenbankleistung erheblich effizienter durch den Einsatz von Indexierung. Da die Indexierung in verschiedenen DBMS unterschiedlich eingesetzt wird und unterschiedliche Bedeutung hat, sollte dieses Kriterium die Liste ergänzen.

Offenbar gehören die Kosten zu den zentralen Kriterien, die durch die Beschaffung des DBMS, die Wartungskosten und die Lizenzgebühren für die Nutzer entstehen. Sowohl Person B als auch Person C haben diesen Faktor erwähnt. Da beide in die Entscheidungsfindung involviert sind, kann davon ausgegangen werden, dass dieses Kriterium eine zentrale Rolle spielen könnte. Person C wies auf einen weiteren wichtigen Aspekt hin, der primär mit den Kosten zusammenhängt – die Entscheidung zwischen Cloud-, On-Premise- oder Hybrid-Lösungen. Obwohl dieser Punkt in der ursprünglichen Kriterienliste nicht berücksichtigt wurde, nimmt er in der modernen Datenwelt an Bedeutung zu.

4.4 User Storys

Als abschließender Schritt der Anforderungsanalyse werden User Stories entwickelt, jedoch nicht speziell für die Nutzwertanalyse. Diese soll zwar den Entscheidungsträgern bei der Entscheidungsfindung helfen, es wäre jedoch sinnlos, die Anforderungen in Form von User Stories darzustellen. Die relevanten Anforderungen, die zur Analyse beitragen, sind diejenigen der oben genannten Datenbanknutzer. Dadurch wird die Gewichtung der Kriterien sowie die Bewertung der Datenbanken erleichtert. Diese User Stories werden in den weiteren Kapiteln, beim Vergleich der DBMS und bei der Erstellung der Nutzwertanalyse, genutzt. Die unten aufgeführten User Stories stellen eine Kombination aus der Kriterienliste und den Ergebnissen der Interviews dar. Sie geben Antworten auf die in (Tresp, 2022) aufgeführten Leitfragen.

Wer ist der Stakeholder?

Was möchte der Stakeholder?

Wieso möchte dies der Stakeholder?

4.4.1 Anwender

Als Anwender möchte ich, ...

[A1] dass meine Datenbank hochverfügbar ist, um auch bei einem Serverausfall weiterhin Zugriff auf meine Daten zu haben.

[A2] dass meine Datenbank immer konsistente und zuverlässige Daten liefert, damit ich in den Daten sicher sein kann.

[A3] dass meine Datenbank mit BI-Tools wie Power BI einfach integrierbar ist, damit ich meine Daten effizient für Analysen und Berichte nutzen kann.

[A4] eine schnelle und zuverlässige Performance bei Lese- und Schreiboperationen, um meine Geschäftsprozesse ohne Verzögerung durchführen zu können.

[A5] Zugriffskontrollen für meine Datenbank definieren können, damit nur autorisierte Personen meine Daten sehen oder bearbeiten können.

[A6] dass meine Datenbank auch bei hoher Last performant bleibt, damit meine Anwendungen, die auf der Datenbank basieren, auch bei hoher Last performant bleiben.

[A7] eine Datenbank mit geringer Latenz für Lese- und Schreiboperationen, damit meine Anwendungen schnell reagieren.

4.4.2 IT-Operations

Als IT-Operationsspezialist möchte ich, ...

[O1] dass die Datenbank eine zuverlässige Backup- und Wiederherstellungsfunktion bietet, um Datenverluste zu vermeiden.

[O2] ich die Möglichkeit haben, die Datenbank flexibel zu skalieren, um steigende Anforderungen ohne Performance-Einbußen zu bewältigen.

[O3] Datenverschlüsselung und Zugriffskontrollen konfigurieren können, um die Sicherheit und den Datenschutz zu gewährleisten.

[O4] Backup- und Wiederherstellungsmechanismen nutzen, um Datenverluste im Falle eines Fehlers oder Cyberangriffs zu minimieren.

[O5] Lizenz- und Betriebskosten optimieren, um eine kosteneffiziente Datenbanklösung bereitzustellen.

[O6] sicherstellen, dass die Datenbank mit den bestehenden Unternehmensanwendungen kompatibel ist, um eine reibungslose Integration zu ermöglichen.

[O7] eine einfache Möglichkeit zur Optimierung von Indizes haben, um die Leseleistung der Datenbank zu verbessern und die Performance zu steigern.

5 Vergleichende Analyse

In diesem Kapitel werden SQL und NoSQL Datenbanksysteme anhand der im Kapitel „Vergleichskriterien“ genannten Aspekte miteinander verglichen. Zusätzlich wird der Aspekt der Indexierung zur Optimierung der Performance im Abschnitt „Leistung“ betrachtet. Um den Vergleich übersichtlicher zu gestalten, wird er in funktionale und nicht funktionale Kriterien unterteilt. Der funktionale Vergleich soll einen Einblick auf die Unterschiede zwischen den DBMS in Hinsicht auf ihre Fähigkeiten und Features geben. Der nicht funktionale Vergleich hingegen soll zeigen, wie die DBMS in Bezug auf die Qualität arbeiten (vgl. Grande, 2014).

Dieses Kapitel nimmt mehrfach Bezug auf die beiden Anwendungsszenarien, die im Folgenden genauer erläutert werden. Der Fokus der Performance-Analyse liegt dabei hauptsächlich auf den Szenarien, da die Tests mit E-Commerce- und Banking-Daten durchgeführt werden. In dem funktionalen und nicht funktionalen Vergleich wird auf die Anwendungsfälle an relevanten Stellen eingegangen.

Da es eine Vielzahl an NoSQL DBMS gibt, die bei einigen der Aspekte unterschiedliche Ansätze haben, wird der Fokus in dieser Arbeit auf MongoDB – einem dokumentenorientierten DBMS – gelegt. SQL-Datenbanken hingegen unterscheiden sich größtenteils ganz wenig in Bezug auf die meisten Vergleichskriterien, dennoch liegt der Fokus auf MySQL als einem Repräsentanten dieser Datenbankart.

Banking Daten Wie jedes Unternehmen haben auch Banken eine Menge von unterschiedlichsten Daten, die gespeichert und verarbeitet werden müssen. Kundenbezogene Daten, Kontodaten, Zahlungsdaten, Transaktionsdaten – an eine DBMS, die all dies speichern soll, werden große Ansprüche gestellt. Denn diese Daten sind nicht nur für die Bank selbst, sondern auch für Kunden, Aufsichtsbehörden und Finanzdienstleister von großer Bedeutung. Hierfür kann eine hohe Konsistenz, maximale Sicherheit, starke Performance, die Möglichkeit zur Speicherung von Daten in spezifischen Regionen sowie ein zuverlässiges Backup-System vorausgesetzt werden.

E-Commerce Online-Shopping gehört heute zum Alltag, und im Hintergrund jedes E-Commerce-Unternehmens stehen Daten, die im Idealfall die reibungslose Arbeit des Shops ermöglichen. Hier sind Kundendaten, Produkt- und Bestellinformationen sowie Daten zum Nutzerverhalten enthalten. Dies sind meistens Daten mit wenig Struktur, bei denen mit häufigen Veränderungen gerechnet werden muss. Hinzu kommen viele gleichzeitige Nutzer, die zu hohen Zugriffszahlen führen können, sowie die Anforderung einer hohen Verfügbarkeit, um Ausfälle zu vermeiden, damit Kunden stets zufrieden mit ihrem Einkaufserlebnis sind.

5.1 Funktionaler Vergleich

5.1.1 Datenstruktur

Das erste Vergleichskriterium sollte von Beginn an aus zwei Perspektiven betrachtet werden: welche Daten gespeichert werden und welche Möglichkeiten das DBMS bietet. Relationale Datenbanken wie MySQL speichern Informationen in Form von strukturierten Tabellen. Dabei muss das Datenschema eindeutig festgelegt sein und die Daten müssen strikt den im Schema definierten Datentypen entsprechen, was eine hohe Konsistenz gewährleistet (vgl. Kaufmann und Meier, 2023).

Sind die Daten semi- oder unstrukturiert, eignen sie sich besser für die Speicherung in NoSQL-DBMS. MongoDB ermöglicht durch seine Schemafreiheit die flexible Verwendung beliebiger Felder innerhalb einer Kollektion. Dadurch können neue Felder und Objekte problemlos hinzugefügt werden (vgl. Kaufmann und Meier, 2023).

Banking Daten sind stark strukturiert. Kunden- und Kontodaten sowie auch Transaktionen lassen sich am besten in Tabellen mit festen Datentypen speichern. Die Felder in solchen Tabellen sind in der Regel vorhersehbar und bleiben unverändert, was MySQL problemlos verarbeiten kann.

E-Commerce-Daten hingegen können sowohl strukturiert als auch unstrukturiert und semistrukturiert sein (vgl. Akter und Wamba, 2016). Während Kundendaten strukturiert gespeichert werden können und das Schema in diesem Fall kaum oder gar nicht verändert werden muss, könnte es sich bei Produktdaten und Kundenbewertungen anders verhalten. Produkte können unterschiedliche Attribute haben: Bücher haben beispielsweise einen Autor und Kleidung hat Größenangaben. Ein flexibles Schema erlaubt ein

unkompliziertes Hinzufügen neuer Eigenschaften. Auch für Kundenbewertungen ist die Schemaflexibilität von Vorteil, da Fotos, Videos und Freitexte in verschiedenen Dokumenten innerhalb einer Kollektion gespeichert werden können.

5.1.2 Datenintegrität & Transaktionen

In dem CAP-Theorem heißt es, dass in einem verteilten Datenbanksystem immer nur zwei der drei Eigenschaften – Konsistenz, Verfügbarkeit und Partitionstoleranz – gleichzeitig erfüllt werden können. Obwohl das CAP-Theorem ursprünglich für verteilte Datenbanksysteme konzipiert wurde und daher davon ausgegangen wird, dass Partitionstoleranz stets gegeben ist, wird in dieser Arbeit auch bei üblicherweise zentralisierten Datenbanksystemen auf das CAP-Theorem Bezug genommen. Das heißt, es wird auch die Kombination von Konsistenz und Verfügbarkeit zugelassen.

Es muss nun zunächst unterschieden werden, ob es sich in diesem Abschnitt um zentralisierte oder verteilte DBS handelt. MySQL wird in der Regel als zentralisierte bzw. als eine Single-Node-Datenbank eingesetzt, in diesem Fall wird MySQL keine Partitionstoleranz anbieten können, da diese nur in verteilten Systemen möglich ist. Daher erfüllt MySQL in der Standardform die Konsistenz sowie Verfügbarkeit gemäß dem CAP-Theorem.

MySQL kann aber auch als verteilte Datenbank konfiguriert werden, beispielsweise durch den Einsatz von Clustern oder Replikation. Die Entscheidung, ob das DBS zentralisiert oder verteilt betrieben wird, hängt von den Anforderungen ab. Im Kontext von Banking-Daten ist es besonders wichtig, sowohl die Konsistenz der Daten sicherzustellen als auch eine hohe Ausfalltoleranz zu gewährleisten (vgl. Kaufmann und Meier, 2023). Die Ausfalltoleranz kann MySQL nur in einer verteilten oder replizierten Konfiguration erreichen. Die Konsistenz wird bei relationalen Datenbanken wie MySQL üblicherweise durch verschiedene SQL-Mechanismen gewährleistet. Dies ist in MySQL unter anderem mit der Autocommit-Einstellung oder dem Kennzeichnen einer Operation als Transaktion mittels Schlüsselworten wie „START TRANSACTION“ oder „BEGIN“ möglich. Bei letzterem ist ein „ROLLBACK“ möglich, sofern kein manueller Commit durchgeführt wurde (vgl. Oracle, 2025a). Konsistenz ist oft unabhängig von dem Anwendungsfall eine entscheidende Anforderung. Wie in User-Story [A2] beschrieben, legen Anwender großen Wert auf Konsistenz, um sich auf die Zuverlässigkeit der Daten verlassen zu können. Dies gilt auch in anderen Bereichen des Finanzsektors, etwa an Börsenplätzen (vgl. Kaufmann und Meier, 2023).

Somit ist MySQL in seiner traditionellen Form als Single-Node-DBMS für den Anwendungsfall der Banking-Daten nicht geeignet. Da NoSQL-Datenbanksysteme häufig bereits in der Standardkonfiguration verteilt sind, erfüllen sie diese Anforderung oft besser. So bietet MongoDB die benötigte Partitionstoleranz für den Banking-Fall. Die Konsistenz in MongoDB ist jedoch nur in einer weicheren Form verfügbar, da es die Anforderungen von BASE erfüllt (vgl. Kaufmann und Meier, 2023). Für Bankdaten wird jedoch eine Konsistenz bevorzugt, die den ACID-Kriterien entspricht. Deswegen wäre MySQL mit starker Konsistenz und Replikation eine vorteilhafte Wahl.

Auch in Szenarien, in denen die Verfügbarkeit zusammen mit der Partitionstoleranz gefordert wird, werden bevorzugt NoSQL-Datenbanksysteme eingesetzt. Dies liegt daran, dass die Eigenschaften A und P des CAP-Theorems mit relationalen Datenbanksystemen nicht realisiert werden können (vgl. Kaufmann und Meier, 2023).

5.1.3 Sicherheit

In diesem Abschnitt liegt der Schwerpunkt auf den technischen Funktionen, mit denen ein DBMS die Sicherheitsanforderungen erfüllt, weshalb dieses Kriterium den funktionalen Anforderungen zugeordnet wird (vgl. Kaufmann und Meier, 2023). Eine zentrale Rolle spielt dabei die CIA-Triade, an der sich in diesem Abschnitt orientiert wird. Die Vertraulichkeit (Confidentiality) wird durch den Schutz der Privatsphäre mittels Authentifizierung gewährleistet. Die Bedeutung dieser Anforderung wird auch durch die User Stories [A5] und [O3] unterstrichen. Um diese Vorgabe zu erfüllen, werden sowohl in MySQL als auch in MongoDB Benutzerkonten für die Nutzer angelegt. In MySQL können Zugriffsrechte basierend auf der Rolle oder dem User als einzelne Person vergeben werden. Das bedeutet, dass Rechte für eine bestimmte Tabelle, einen Tabellenabschnitt, Zeilen oder Spalten entweder einem einzelnen Nutzer oder einer definierten Rolle zugewiesen werden können (vgl. Kaufmann und Meier, 2023). In MongoDB hingegen können Zugriffsrechte lediglich an „built-in“ oder „user-defined“ Rollen vergeben werden (vgl. MongoDB, 2024b). Daher muss jedem Nutzer zwingend eine Rollengruppe zugewiesen werden.

Die Integrität sichert MySQL mit deklarativen Integritätsregeln, die dafür sorgen, dass die Daten auch nach Änderungen korrekt und konsistent bleiben. Hierzu gehören: Primär- und Fremdschlüssel, Constraints wie „UNIQUE“ und der „CHECK“-Befehl sowie Regeln

für das Löschen (vgl. Kaufmann und Meier, 2023, S. 143). MongoDB zeichnet sich hingegen durch seine Schemafreiheit aus, was insbesondere bei der Verarbeitung von Big Data Variety vorteilhaft ist. Um dennoch bei Bedarf die Datenintegrität sicherzustellen, kann in MongoDB eine JSON-Schema-Validierung eingerichtet werden, bei der Eigenschaften von Dokumenten und Kollektionen definiert werden. Wie in MySQL können auch in MongoDB Primärschlüssel und UNIQUE Constraints für die Integrität sorgen. Fremdschlüssel sind bei MongoDB jedoch nicht verfügbar. *„Dies ist eine bewusste Entscheidung, da die Annahme der Vollständigkeit der Dokumente schnellere Abfragen und eine einfachere Partitionierung der Datenbank für sehr große Datensätze ermöglicht.“* (Kaufmann und Meier, 2023, S. 149).

Die Sicherheit der Daten sowie die Kontrolle darüber, welcher Mitarbeiter Zugriff auf welche Informationen hat, sind im Fall einer Bank von großer Bedeutung. Es handelt sich um sensible Daten, die konsistent bleiben müssen und bei denen Änderungen ausschließlich kontrolliert vorgenommen werden dürfen. In MySQL kann dies durch restriktive Löschregeln unterstützt werden, die festlegen, welche Daten gelöscht werden dürfen (vgl. Kaufmann und Meier, 2023).

Das „A“ in CIA – die Verfügbarkeit – wurde bereits in dem Abschnitt „Transaktionen“ in Bezug auf das CAP-Theorem erläutert.

In der Tabelle 5.1 werden die funktionalen Vergleichskriterien mit der Bewertung zusammengefasst.

Nr.	Kategorie	Kriterium	MySQL	MongoDB
1	Datenstruktur	Strukturierung der Daten	Strukturierte Daten; fest strukturierte Schemata	Semi- und nicht strukturierte Daten; Schemaflexibilität
2	Datenintegrität	CAP-Theorem	Zentralisiert: CA; Cluster: CP	CP oder AP
5	Sicherheit	Vertraulichkeit: Zugriffskontrollen	Rollen- und Privilegienbasierter Zugriff (RBAC & PBAC) auf Tabellen, Tabellenabschnitte, Zeilen und Spalten	Rollenbasierter Zugriff (RBAC) auf Datenbank, Collection und Dokument
		Integrität	Deklarative Integritätsregeln: Primärschlüssel, Fremdschlüssel, UNIQUE, CHECK etc.	JSON-Schema-Formating; Primärschlüssel, UNIQUE; keine Fremdschlüssel, aber vorteilhaft für Big Data
		Verfügbarkeit	siehe Datenintegrität	siehe Datenintegrität

Tabelle 5.1: Funktionale Vergleichskriterien

5.2 Nicht funktionaler Vergleich

5.2.1 Leistung

Die Leistung einer Datenbank wird stark von den häufigsten Operationen und der Art und Weise der Datenspeicherung beeinflusst. Die Einführung von Indexierung, wie im Interview von Person A erwähnt, kann die Geschwindigkeit von SQL-Abfragen bedeutend steigern. Unabhängig vom spezifischen Anwendungsfall wird von einer Datenbank oft eine hohe Performance erwartet, da die darauf basierenden Anwendungen auch unter hoher Last leistungsfähig bleiben müssen, wie in User Story [A6] erläutert wird.

In einem Artikel (Capris u. a., 2022) wurden MySQL und MongoDB mit dem Benchmarking Tool Yahoo! Cloud Serving Benchmark anhand unterschiedlicher Workloads in Hinsicht auf den Durchsatz und die Laufzeit verglichen. Die Ergebnisse zeigten, dass MongoDB bei einem Workload aus 50 % Leseoperationen und 50 % Schreiboperationen bei 10 Millionen Datensätzen um 55,77 % besseren Durchsatz zeigte. Das liegt laut der Autoren daran, dass MongoDB weder das Schema noch die Fremdschlüssel überprüfen muss. Allerdings ist dies nicht für jeden Anwendungsfall ideal, da beispielsweise bei Banking-Daten sichergestellt werden muss, dass jeder neue Datensatz den Vorgaben des Schemas entspricht, da dies die Konsistenz der Daten beeinflusst. In einer weiteren Studie (Yedilkhan u. a., 2023) wurde MongoDB mit PostgreSQL – einer SQL-Datenbank – und einer weiteren NoSQL-Datenbank hinsichtlich der Leistung mit einer steigenden Anzahl von Daten verglichen. Die Ergebnisse zeigten, dass MongoDB beim Hinzufügen von Daten deutlich besser abschneidet, vor allem im Vergleich zu dem SQL-Datenbanksystem. Bei der Suchoperation jedoch hat die Leistung der NoSQL-Datenbank rapide nachgelassen und die PostgreSQL zeigte ein um einiges besseres Ergebnis.

Im Abschnitt „Performance-Analyse“ wird genauer betrachtet, wie sich MongoDB und MySQL im Detail im Umgang mit Daten entsprechend den jeweiligen Anwendungsfällen im Kontext von CRUD-Operationen verhalten.

Ein entscheidender Faktor, der Einfluss auf die Performance eines DBMS hat, ist die Skalierbarkeit. Die Verteilung der Arbeitslast auf mehrere Knoten kann die Leistung bei einer großen Menge von Lese- und Schreiboperationen sowie deren Geschwindigkeit und Reaktionszeiten positiv beeinflussen. Um mit großen Datenmengen besser zurechtzukommen, sind NoSQL-Datenbanksysteme horizontal skalierbar (vgl. Mukherjee u. a., 2015). Die starke horizontale Skalierbarkeit bei MongoDB folgt zudem daraus, dass es

sich bei MongoDB um ein dokumentenbasiertes DBMS handelt (vgl. Fasel, 2016). Diese wird durch die Verwendung von Sharded Clustern zusätzlich unterstützt (vgl. MongoDB, 2024a). In dem Fall von MySQL kommt es wieder darauf an: ist MySQL zentralisiert, kann diese nur vertikal skaliert werden, da es sich lediglich um einen Server handelt. Dies kann von Nachteil werden, wenn der Server ausfällt und kein Reserveserver vorhanden ist – die Daten sind einfach nicht mehr zugreifbar (vgl. Rahm u. a., 2015). Ist die Rede von MySQL in Form von einem Cluster oder Replikation, so kann auch wie bei MongoDB eine horizontale Skalierbarkeit angewandt werden.

5.2.2 Datencharakteristika

Wenn es um Daten geht, sollte erneut ein Blick auf die Anwendungsfälle geworfen werden. E-Commerce-Daten sind vielfältig: unterschiedliche Produkte und deren Beschreibungen, Bewertungen von Kunden, die aus Freitext, Bildern, Videos und Noten bestehen können, sowie verschiedene Bestellinformationen. Zudem können es schnell viele Daten werden, wenn der Betreiber neue Artikel hinzufügt. Während Sale-Phasen oder Aktionen besuchen viele Kunden zur gleichen Zeit die Webseite, was dafür sorgt, dass das Datenvolumen rasant erhöht wird (vgl. Akter und Wamba, 2016). Tätigen Kunden Bestellungen und hinterlassen Bewertungen, steigt das Datenvolumen noch weiter. Somit erfüllen E-Commerce-Daten alle Eigenschaften von Big Data, welche oft von NoSQL-Datenbanksystemen am besten verarbeitet und gespeichert werden können (vgl. Kaufmann und Meier, 2023).

Banking-Daten können ebenfalls an Volumen zunehmen, wenn neue Kunden gewonnen werden und dadurch die Menge der Transaktionsdaten wächst. Die von einer Bank gespeicherten Daten ändern sich jedoch mit vergleichsweise geringer Wahrscheinlichkeit und sind zudem meist strukturiert. Wenn nicht davon auszugehen ist, dass das Datenvolumen schnell anwächst, kann MySQL aufgrund der Eigenschaften dieser Daten eine geeignete Wahl sein.

5.2.3 Datensicherung

Eine Bank stellt höchste Anforderungen an ihre Datenbanken, insbesondere in Bezug auf die Datensicherung, da selbst im Falle eines Datenverlusts die vollständige Wiederherstellbarkeit der Daten gewährleistet sein muss. Auch bei E-Commerce-Daten wäre ein Datenverlust problematisch, jedoch nicht so kritisch wie bei einer Bank. Sowohl in MySQL als auch in MongoDB können vollständige Backups mit Tools wie Mysqldump bzw. Mongodump erstellt werden. Mit Binary Logs sind Banking-Daten besonders gut in MySQL gesichert, da jede Transaktion, die bisher nicht vollständig abgeschlossen wurde, dauerhaft gespeichert wird (vgl. Oracle, 2025a). Dies erfüllt die hohen Anforderungen an Konsistenz. In MongoDB können Daten bei häufigen Änderungen mithilfe von Snapshots gesichert werden (vgl. MongoDB, 2024a).

5.2.4 Datenverteilung

Banking-Daten unterliegen häufig gesetzlichen Vorgaben, die eine Speicherung im Ausland untersagen, weshalb ihre geografische Verteilung auf das jeweilige Landesgebiet beschränkt bleibt. E-Commerce-Daten hingegen können global verteilt gespeichert werden, was sowohl die Performance als auch die Konsistenz verbessern kann. Die Unterstützung der geografischen Datenverteilung von MongoDB und MySQL ist auf Aspekte wie Skalierung und Partitionstoleranz zurückzuführen, die in den oberen Abschnitten bereits erläutert wurden. Erwähnenswert ist dennoch das Sharding, das von MongoDB für die Verteilung der Daten auf mehrere Rechner eingesetzt wird (vgl. Kaufmann und Meier, 2023). Daten mit MySQL geografisch verteilt zu speichern, kann mit Replikation ermöglicht werden.

5.2.5 Verarbeitungsanforderungen

Sowohl Bankdaten als auch Online-Shop-Daten können auf die Echtzeitverarbeitung angewiesen sein. „Eine oft zitierte Anwendung zu Echtzeitverarbeitung ist das zeitnahe Entdecken von Betrugsfällen bei monetären Transaktionen, wie bei Kreditkartenbetrug.“ (Fasel, 2016, S. 134). E-Commerce-Daten fallen häufig in die Kategorie Big Data. Nach einigen Definitionen zählt die Echtzeitverarbeitung zu den Merkmalen von Big Data (vgl. Akter und Wamba, 2016). Daher ergibt eine Echtzeitverarbeitung vor allem bei großen

Datenmengen Sinn. Zentralisierte Datenbanksysteme sind für eine effiziente Echtzeitverarbeitung der Daten daher ungeeignet. Der Grund dafür liegt darin, dass die Daten ausschließlich auf einem einzigen Server gespeichert werden, was die Anforderungen an eine schnelle Verarbeitung großer Datenmengen nicht erfüllt (vgl. Liu u. a., 2022). Das heißt, MySQL kommt in der traditionellen Form als Single-Node-DBMS hier nicht in Betracht. Im Gegensatz dazu sind verteilte nicht relationale Datenbanksysteme aufgrund der Lastverteilung auf mehrere Server, der Unabhängigkeit der Daten voneinander und der fehlenden strengen Konsistenzanforderungen besser geeignet (vgl. Liu u. a., 2022). Die Frage bleibt jedoch, ob eine Bank mit der schwächeren Konsistenz, die MongoDB bietet, zufrieden wäre, da dies eine der wichtigsten Anforderungen ist.

5.2.6 Wirtschaftliche Aspekte

Die Lizenz- und Preismodelle sowohl bei MongoDB als auch MySQL unterscheiden sich nicht nur bei dem Kostenaspekt, sondern auch bei den zusätzlichen Funktionen, die damit verbunden sind. Den Wunsch nach Kosteneffizienz wurde in der User-Story [O5] ausgedrückt. In dem folgenden Abschnitt werden nur für diese Arbeit relevanten Unterschiede erläutert.

MongoDB Kosten & Lizenzierung

Die MongoDB Community Edition unter Server Side Public License ist kostenlos verfügbar und eignet sich für Nutzer, die eine Open-Source-Lösung bevorzugen. Für Unternehmen mit höheren rechtlichen Anforderungen bietet MongoDB die kommerzielle Lizenz an. Der vollständig verwaltete Cloud-Service MongoDB Atlas bietet unterschiedliche Tarife in Abhängigkeit von Faktoren wie Speicher, RAM und vCPUs. Die Kosten werden dabei in „pay as you go“ abgerechnet, was dem Kunden Kosteneffizienz bietet. Dabei kann auch die Skalierung flexibel an den Bedarf angepasst werden (vgl. MongoDB, 2025).

MySQL Kosten & Lizenzierung

MySQL Community Edition ist eine unter General Public License kostenfreie Open-Source-Lizenz. Auch kommerzielle Editionen von MySQL sind möglich, die Preise unterscheiden sich je nach Funktionsumfang mit einer jährlichen Abrechnung. Für hohe Verfügbarkeitsanforderungen wird das MySQL Cluster angeboten, dessen Preise sich nach

den spezifischen Anforderungen richten. Wird eine cloudbasierte Lösung mit MySQL angestrebt, so muss zu externen Anbietern wie Microsoft Azure oder Amazon Web Services (AWS) gegriffen werden (vgl. Oracle, 2025b).

Ökosystem und Support

Sowohl MySQL als auch MongoDB verfügen über ein breites Ökosystem mit zahlreichen Tools, Cloud-Integrationen und Community-Support. Beide DBMS bieten neben Open-Source-Community-Support, offiziellen Dokumentationen und Support-Foren auch kommerzielle Modelle mit 24/7 Support. Da MySQL bereits länger auf dem Markt ist als MongoDB, hat das relationale DBMS eine größere und etabliertere Community. Da relationale Datenbanksysteme um einige Jahre länger im Einsatz in vielen Unternehmen sind, kann es auch sein, dass dies zu einer größeren Entwicklerbasis geführt hat.

In der Tabelle 5.2 werden die wesentlichen Unterschiede oder Gemeinsamkeiten zusammengefasst.

Nr.	Kategorie	Kriterium	MySQL	MongoDB
3.1	Leistung	Lese- und Schreibgeschwindigkeit	Indexierung kann Performance verbessern	Besseren Durchsatz und Laufzeit
3.2		Skalierbarkeit	Zentralisiert: nur vertikal; Verteilt: horizontal	Horizontal
4.1	Datencharakteristika	Datenvolumen	Für nicht schnell steigendes Datenvolumen geeignet	Für Big Data Anwendungen geeignet
4.2		Datenvielfalt	Speichert strukturierte Daten am besten	Semi-strukturierte & unstrukturierte Daten
6	Datensicherung	Backup & Wiederherstellung	Vollständige Backups; binary logs	Vollständige Backups; snapshots
7	Datenverteilung	Geografische Verteilung	Kommt auf die Skalierung an; mit Replikation möglich	Da horizontal skalierbar, kein Problem
8	Verarbeitungsanforderungen	Echtzeitverarbeitung	Nur wenn als Cluster oder mit Replika	Gut geeignet mit womöglich schwächerer Konsistenz
9.1 & 9.2	Wirtschaftliche Aspekte	Kosten & Lizenzierung	Open Source & kommerzielle Editionen; i. d. R. jährliche Abrechnung	Open Source & kommerzielle Editionen; pay-as-you-go
9.3		Ökosystem & Support	Umfangreiches Ökosystem und Supportmöglichkeiten; länger auf dem Markt	Umfangreiches Ökosystem und Supportmöglichkeiten

Tabelle 5.2: Nicht funktionale Vergleichskriterien

5.3 Performance-Analyse

Die Performance-Analyse beinhaltet acht Testfälle, in denen der Durchsatz (die Anzahl der Operationen pro Sekunde) und die Latenz (Zeit, die eine einzelne Anfrage benötigt) getestet werden. Es wurden die zu Beginn des Kapitels beschriebenen Anwendungsfälle von Bank- und E-Commerce-Daten eingesetzt und weiter ausgebaut. Hierfür wurden randomisierte Daten mit Python generiert und in CSV-Dateien gespeichert. Diese wurden dann in beide Datenbanken geladen.

5.3.1 Testumgebung

Die Tests wurden auf Single-Node mit einem MacBook Air macOS, 3.2 GHz Apple M1, 16 GB RAM und 256GB SSD durchgeführt. Dabei wurden sie einzeln nacheinander durchgeführt, da die Methoden bzw. Tests um Systemressourcen konkurrieren könnten, was zu verfälschten Ergebnissen führen kann. Es wurde die *MongoDB 7.0.11 Community Edition in dem MongoDB Compass Version 1.45.0 (1.45.0)* benutzt. Sowie der *MySQL Community Server 9.1.0* und die *MySQL Workbench* als das GUI-Tool.

5.3.2 Testscenarien

Die für die Testscenarien erstellten Daten sollen lediglich einen Ausschnitt möglicher Daten des jeweiligen Anwendungsfalls darstellen. Dabei stehen die Daten teilweise in Beziehung zueinander, beispielsweise sind die Konten mit den Kunden verknüpft. Sowohl die Bankdaten als auch die Daten des Online-Shops sind strukturiert. Die E-Commerce-Daten enthalten in den „Products“-Datensätzen längere Beschreibungen, die theoretisch durch Bilder, Videos oder noch ausführlichere Texte ergänzt werden könnten, um die Strukturiertheit der Daten aufzulösen. Von dieser Erweiterung wurde jedoch abgesehen, da sie den Rahmen dieser Arbeit überschreiten würde. Außerdem wurde entschieden, trotz der Tatsache, dass E-Commerce-Daten häufig unter den Begriff Big Data fallen, eine kleinere Anzahl an Datensätzen im Vergleich zu den Bankdaten zu generieren. Dies wurde gemacht, um das Experiment auch mit einer geringeren Datenmenge durchführen zu können. Da beide Datensätze eine ähnliche Struktur aufweisen, fiel die Wahl auf E-Commerce-Daten.

Banking

Für die Tests im Kontext der Banking-Daten wurden folgende CSV-Dateien generiert:

- Bank Customers mit 500.000 Datensätzen
- Bank Accounts mit 1 Million Datensätzen
- Bank Transactions 10 Millionen Datensätzen

E-Commerce

Die E-Commerce-Daten sind in folgende CSV-Dateien aufgeteilt:

- E-Commerce Customers mit 500.000 Datensätzen
- E-Commerce Products mit 1.000 Datensätzen
- E-Commerce Orders mit 1 Million Datensätzen

5.3.3 Test Set-up

Für die Tests wurde *Python der Version 3.12.3* mit *pymongo* sowie *pymysql* verwendet. So konnten die Tests standardisiert und wiederholend durchgeführt werden, was zur aussagekräftigeren Auswertung der Ergebnisse beiträgt. Als IDE wurde PyCharm ausgewählt, wo ganz einfach die Verbindungen zu MySQL und MongoDB hergestellt werden konnten.

Folgende CRUD-Operationen wurden getestet:

- `simple_read_test()` – Einzelabrufe anhand einer ID
- `complex_read_test()` – komplexe Abfragen mit Filterung und Aggregation
- `simple_insert_test()` – Einzeleinfügungen
- `batch_insert_test()` – Batch-Einfügungen
- `simple_update_test()` – Einzelupdates
- `complex_update_test()` – Komplexe Updates von zwei Feldern

- `simple_delete_test2()` – Einzelnes Löschen
- `batch_delete_test()` – Batch-Löschungen

Wichtig ist zudem, dass weder MongoDB noch MySQL auf keine Art und Weise optimiert wurden. Bei MySQL würde es insbesondere Sinn ergeben, die Indexierung einzuführen. Die fehlende Indexierung macht sich auch auf den Ergebnissen bemerkbar. Da dies aber Konfigurationen sind, die zusätzlich durchgeführt werden müssen, wurde hier die Entscheidung getroffen, mit den DBMS so zu arbeiten, wie sie vom Hersteller geliefert werden.

Der Fokus der Tests liegt auf der Latenz und dem Durchsatz. Die Latenz gibt an, wie lange es dauert, eine Operation auszuführen, gemessen in Sekunden pro Abfrage. Eine hohe Latenz bedeutet, dass der Nutzer länger warten muss, bis das Abfrageergebnis angezeigt wird. Der Durchsatz hingegen misst, wie viele Anfragen pro Sekunde von der Datenbank verarbeitet werden können, ausgedrückt in Operationen pro Sekunde. Ein höherer Durchsatz ist vorteilhaft, da er zeigt, wie effizient die Datenbank unter hoher Last arbeitet.

5.3.4 Ergebnisse

Die Ergebnisse der Performance-Tests werden für jeden Anwendungsfall separat präsentiert und erläutert. Die Testfälle sind innerhalb der Kategorien nummeriert, wobei „B1.x“ beispielsweise für Lesetests mit den Banking-Daten steht.

Banking

Read Die größte Last kommt in der Regel mit erhöhten Lese- und Schreiboperationen. Abbildung 5.1 zeigt den Durchsatz der Leseoperation. In dem Simple Read Test werden 10.000 Leseabfragen anhand einer zufälligen `account_id` durchgeführt, indem der Datensatz mit dieser ID herausgesucht wird. MongoDB ist hier um 68 % schneller als MySQL. Der Complex Read Test zählt 1.000 Mal die Transaktionen einer zufällig ausgewählten `account_id` im Jahr 2020. Somit sind es zwei Filterbedingungen, die vorgegeben werden, was potenziell mehr Berechnungen erfordert. Dem Durchsatz nach zeigt MongoDB in diesem Test eine ca. 50 % bessere Leistung als MySQL.

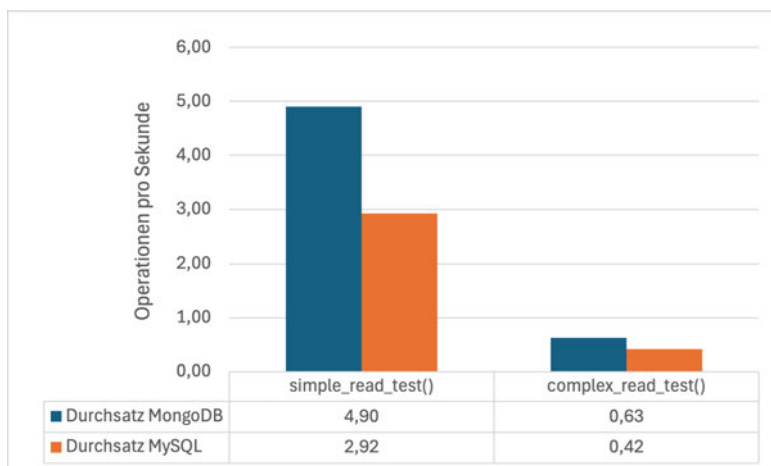


Abbildung 5.1: Durchsatz bei Simple & Complex Read Tests (Banking Daten)

Abbildung 5.2 zeigt, dass MongoDB bei dem Simple Read Test eine um 81 % niedrigere Latenz hat als MySQL. Im Complex Read Test ist die Latenz von MongoDB um etwa 33 % besser als bei MySQL.

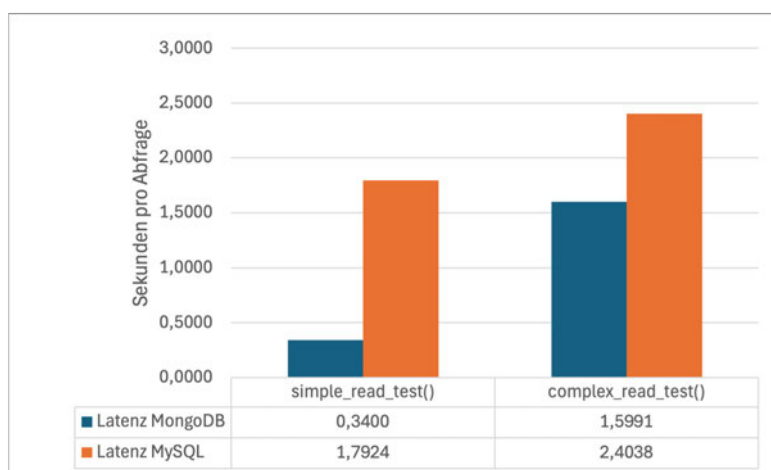


Abbildung 5.2: Latenz bei Simple & Complex Read Tests (Banking Daten)

Insert In den Insert-Tests werden 10.000 Datensätze auf zwei verschiedene Arten eingefügt: als einzelne Datensätze und in Batches zu jeweils 1.000 Datensätzen. Eine Gemeinsamkeit, die MongoDB und MySQL im Durchsatz zeigen, ist, dass Batch-Einfügungen zu einem höheren Durchsatz führen. Das liegt daran, dass die Datenbank weniger Schreiboperationen verarbeiten muss. In Abbildung 5.3 ist jedoch deutlich zu sehen, dass MySQL

mit einfachen Einfügungen mit einem 33,8 % besserem Durchsatz besser zurechtkommt als MongoDB. In dem Batch-Insert ist jedoch MongoDB 169,5 % schneller als MySQL.

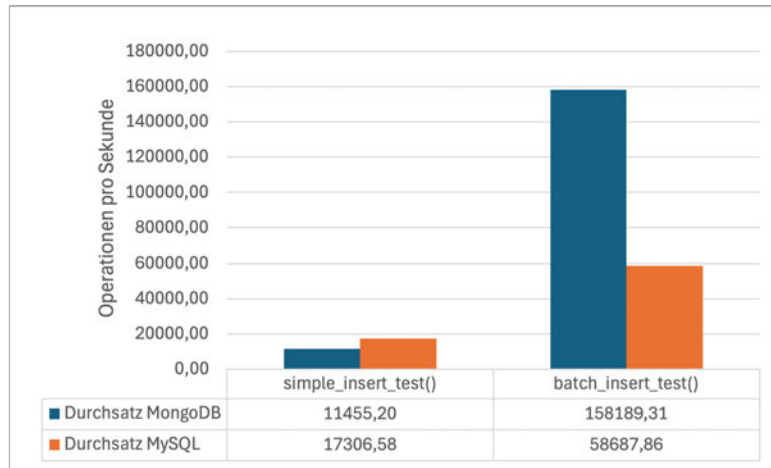


Abbildung 5.3: Durchsatz bei Simple & Batch Insert Tests (Banking Daten)

Hinsichtlich der Zeit, die pro Abfrage benötigt wird, zeigt sich, dass das Einfügen von Daten bei beiden Datenbanksystemen, unabhängig von der Methode, so effizient ist, dass, wie in Tabelle 5.3 zu erkennen, die Latenzen nahezu 0 erreichen.

		Latenz MongoDB (Sek./ Abfr.)	Latenz MySQL (Sek./ Abfr.)
B2.1	simple insert test	0,0001	0,0001
B2.1	complex insert test	0,0000	0,0000

Tabelle 5.3: Latenz bei Simple & Batch Insert Tests (Banking Daten)

Update Die Update-Tests ändern die Datensätze, in dem beim Simple Update Test nur der Status von 10.000 zufällig gewählten `account_id`'s verändert wird. Dabei muss beachtet werden, dass es hier nicht nur um die Updateoperation geht, sondern auch, dass der Datensatz erst einmal gefunden werden muss. Es wäre daher bei sowohl MySQL als auch MongoDB vorteilhaft, wenn `account_id` indiziert wäre. Das zeigt sich auch in den Ergebnissen, denn wie in Abbildung 5.4 zu sehen ist, liegt die relationale Datenbank mit einem Simple Update Test Durchsatz von ca. 94 % kleinerem Durchsatz hinter MongoDB. Sollen mehrere Felder aktualisiert werden, so sind beide DBMS ziemlich langsam, dabei zeigt MongoDB eine circa 15 % bessere Leistung.

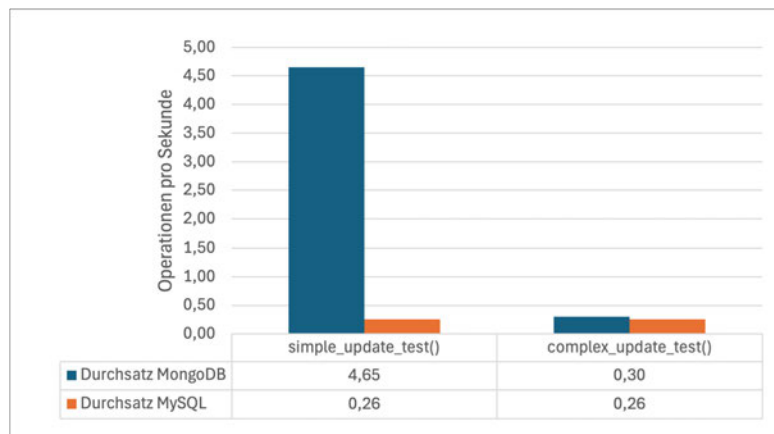


Abbildung 5.4: Durchsatz bei Simple & Complex Update Tests (Banking Daten)

Auch bei der Latenz zeigt MongoDB mit einem um 94,4 % niedrigerem Ergebnis beim Simple Update bessere Leistung. Bei den komplexeren Updates mit zwei zu verändernden Zeilen sind auch in Hinsicht auf die Latenz beide Datenbanksysteme langsam, wie in Abbildung 5.5 zu erkennen ist.

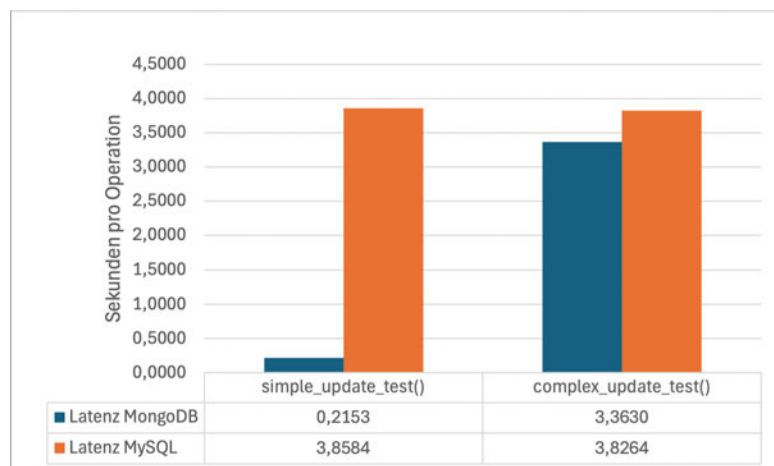


Abbildung 5.5: Durchsatz bei Simple & Complex Update Tests (Banking Daten)

Delete Die letzte CRUD-Operation ist das Löschen von Datensätzen. Werden Zeilen in dem Simple Delete Test einzeln anhand der account_id gelöscht, zeigt MongoDB mit einem deutlich höheren Durchsatz bessere Ergebnisse als MySQL. Wie in Abbildung 5.6 zu sehen, erhöht sich bei beiden Datenbanksystemen der Durchsatz um ein Vielfaches,

wenn Datensätze in Batches von 100 gelöscht werden, anstatt alle 1.000 auf einmal. Dabei ist erneut zu beachten, dass in beiden Delete-Testfällen zunächst nach der zufälligen `transaction_id` gesucht wird, bevor die Löschoperation ausgeführt wird. Das bedeutet, dass bei einer korrekten Indexierung womöglich ganz andere Ergebnisse erzielt werden könnten.

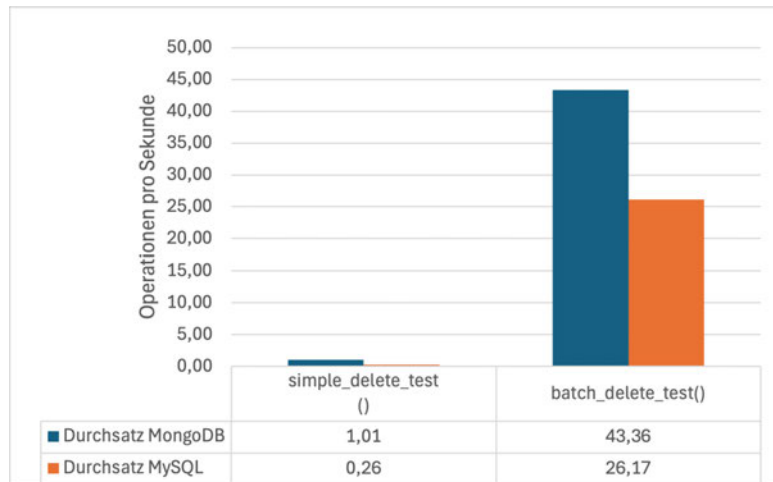


Abbildung 5.6: Durchsatz bei Simple & Batch Delete Tests (Banking Daten)

Die um 74 % höhere Latenz von MySQL im Vergleich zu MongoDB zeigt, dass MongoDB besser mit einzelnen Löschungen umgehen kann. Wie jedoch in Abbildung 5.7 zu erkennen ist, sinkt die Latenz bei beiden DBMS, wenn mit Batches gelöscht wird. Der Unterschied zwischen MongoDB und MySQL ist diesem Fall nicht erheblich.

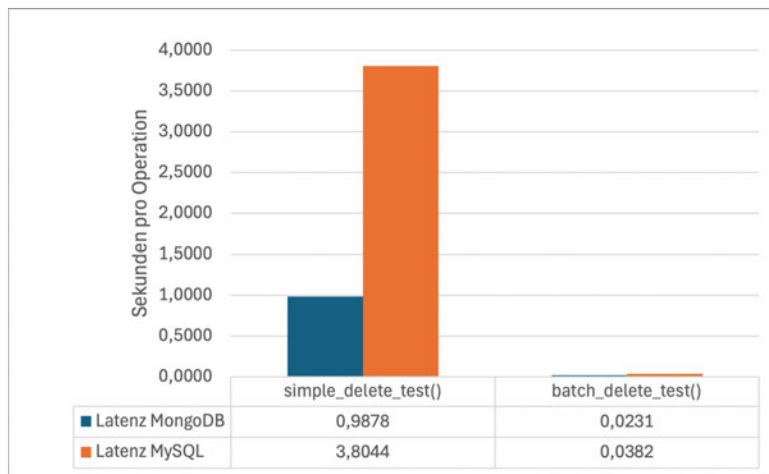


Abbildung 5.7: Latenz bei Simple & Batch Delete Tests (Banking Daten)

E-Commerce

Da Daten bei dem Anwendungsfall E-Commerce ähneln von der Struktur her den Daten der Bank, jedoch sind es die kleinen Unterschiede in den Daten und Tests, die den Unterschied in den Ergebnissen machen können.

Read Die Read-Tests für die E-Commerce-Daten weichen leicht von denen für die Banking-Daten ab. Statt durch 10 Millionen Datensätze wurde hier durch eine deutlich kleinere Menge iteriert, nämlich durch die Tabelle bzw. Kollektion „Customers“ mit 500.000 Datensätzen. Dies wirkte sich auf den Durchsatz aus, wie in Abbildung 5.8 zu sehen ist. Besonders auffällig ist die deutliche Leistungssteigerung von MySQL. Vergleicht man jedoch die Ergebnisse von MongoDB und MySQL, wird deutlich, dass MongoDB weiterhin sowohl beim Simple Test als auch beim Complex Test weiterhin die besseren Ergebnisse erzielt. Auch die Latenz hat sich verändert und zeigt bei beiden, MongoDB und MySQL, bessere Ergebnisse, wobei MongoDB nur einen minimalen Vorsprung hat.

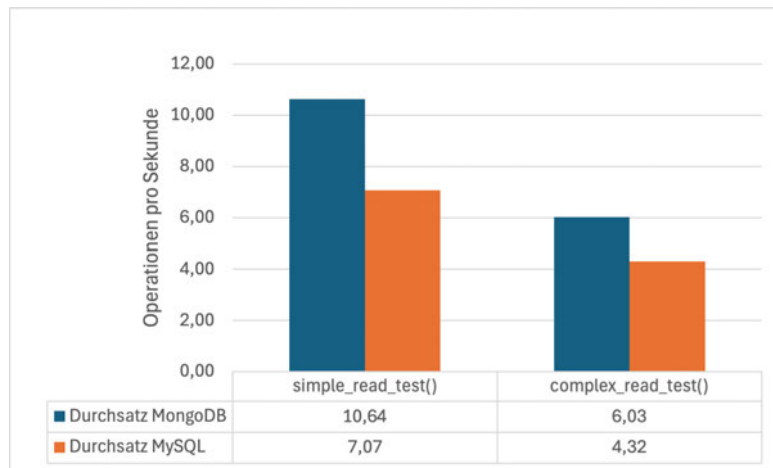


Abbildung 5.8: Durchsatz bei Simple & Complex Read Tests (E-Commerce Daten)

Insert MongoDB ist bei dem Einfügen neuer Datensätze um ca. 58 % schneller als MySQL. Dies kann an der Schemafreiheit von MongoDB und den weniger strengeren Anforderungen an die Konsistenz liegen. Wie in Abbildung 5.9 zu sehen ist, erzielen beide Datenbanksysteme deutlich bessere Ergebnisse, wenn Datensätze in Batches eingefügt werden. Die Latenz ist, ähnlich wie bei den Banking-Daten, so gering, dass dazu nicht viel gesagt werden kann.

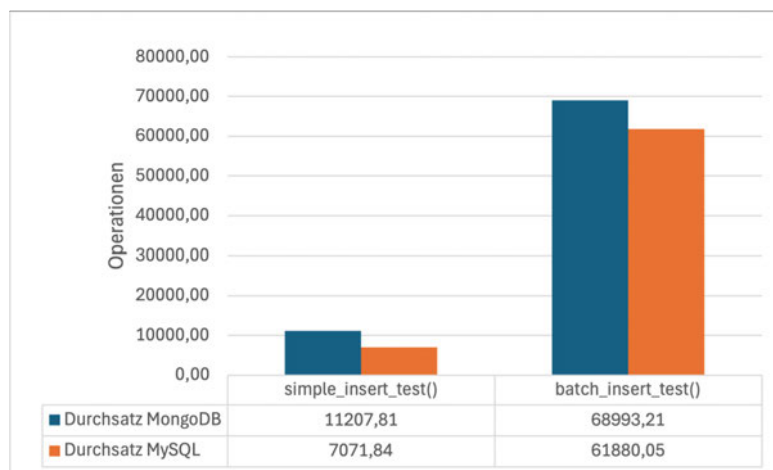


Abbildung 5.9: Durchsatz bei Simple & Batch Insert Tests (E-Commerce Daten)

Update Das einfache sowie komplexere Aktualisieren der Datensätze zeigt einen Durchsatzvorteil von etwa 40 % für MongoDB, wie in Abbildung 5.10 dargestellt. Dieser Vorteil wird durch die Latenzwerte zusätzlich bestätigt, wobei die Latenz bei beiden DBMS sehr niedrig ist, wie auf Abbildung 5.11 zu erkennen ist. In den Tests wird bei MongoDB der Operator \$inc verwendet, um das Feld total_amount zu erhöhen. Dieser Operator ist innerhalb eines Dokuments atomar, was bedeutet, dass keine anderen Operationen das Update während seiner Ausführung beeinträchtigen können (vgl. GeeksforGeeks, 2025).

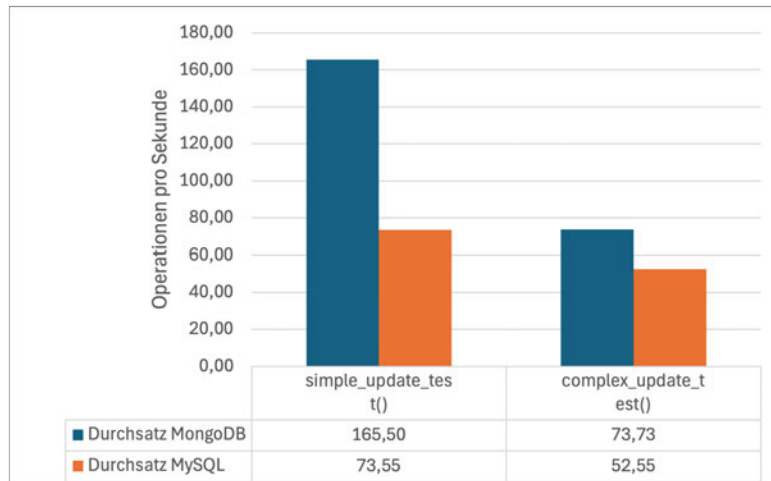


Abbildung 5.10: Durchsatz bei Simple & Complex Update Tests (E-Commerce Daten)

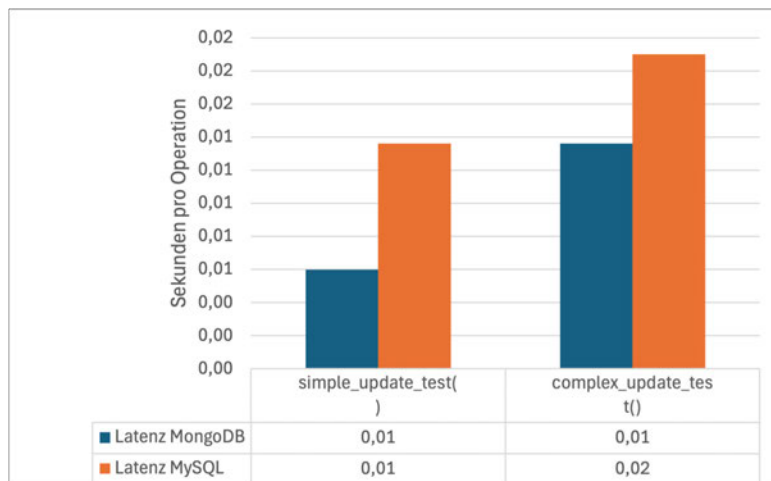


Abbildung 5.11: Latenz bei Simple & Complex Update Tests (E-Commerce Daten)

Delete Sowohl bei MongoDB als auch bei MySQL könnte eine Indexierung der richtigen Felder zu einem besseren Ergebnis beim Löschen der Datensätze führen. So müsste MongoDB nicht durch die komplette Kollektion nach den richtigen Dokumenten suchen wie auch MySQL nicht durch die ganze Tabelle. Die Batch Delete Tests zeigen, wie in Abbildung 5.12 und 5.13 zu sehen, deutlich bessere Ergebnisse sowohl bei dem Durchsatz als auch bei der Latenz.

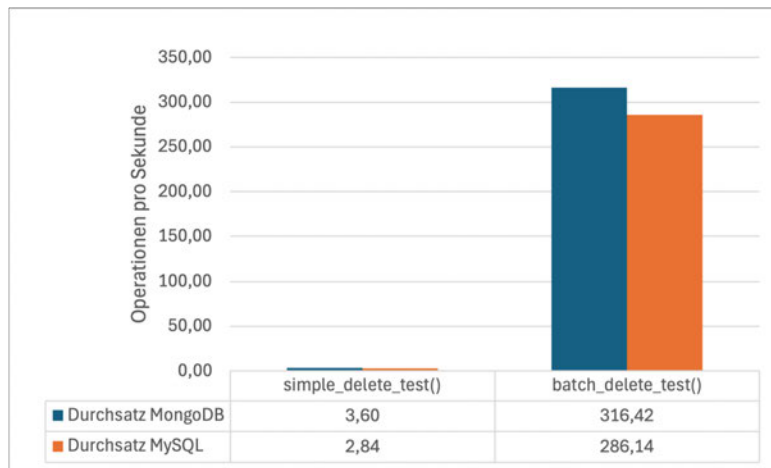


Abbildung 5.12: Durchsatz bei Simple & Batch Delete Tests (E-Commerce Daten)

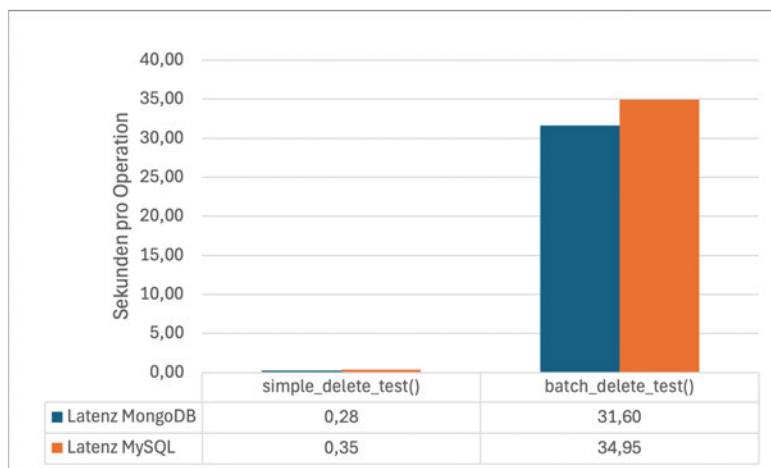


Abbildung 5.13: Latenz bei Simple & Batch Delete Tests (E-Commerce Daten)

5.3.5 Fazit

Die Performance-Analyse zeigt deutlich, dass MongoDB bei allen Tests ein besseres Ergebnis erzielt als MySQL, in einigen Fällen sogar mit mehrfachem Vorsprung. Dabei ist zu beachten, dass der Unterschied zwischen den zwei Datenbanksystemen voraussichtlich noch höher wäre, wenn es sich um semistrukturierte oder gar unstrukturierte Daten handeln würde, da MongoDB genau für solche Art von Daten gedacht ist.

6 Nutzwertanalyse

In diesem Abschnitt erfolgt die Konzeption einer Nutzwertanalyse, deren Ziel die strukturierte Darstellung des Vergleichs zwischen MongoDB und MySQL für die Anwendungsszenarien Banking und E-Commerce-Daten ist. Die Bewertung der DBMS basiert zum überwiegenden Teil auf den im Kapitel „Vergleichende Analyse“ zusammengefassten Erkenntnissen. Des Weiteren wurde festgelegt, dass MySQL im Rahmen der Nutzwertanalyse in der traditionellen Form als ein zentralisiertes und MongoDB als ein verteiltes Datenbanksystem betrachtet wird. Deswegen wird in dem Kriterium „Leistung“ bzw. „Schreib- und Lesegeschwindigkeit“ nur teilweise auf die Ergebnisse aus der Performance-Analyse zurückgegriffen, da beide DBMS in den Tests als zentralisierte Datenbanksysteme eingesetzt werden. Sowohl die Gewichtung als auch die Bewertung basieren auf subjektiven Einschätzungen und persönlicher Wahrnehmung.

In den nachfolgenden Abschnitten wird eine Erläuterung der einzelnen Schritte der Nutzwertanalyse vorgenommen, wobei die Grundprinzipien der Nutzwertanalyse gemäß (Kühnapfel, 2021, S. 6) Grundlage sind.

6.1 Schritt 1: Kriterien

Die Bewertungskriterien wurden basierend auf der Vergleichskriterienliste aus Kapitel 3 zusammengestellt. Dabei wurden folgende Änderungen vorgenommen:

- „Datenstruktur“ wurde zu dem Kriterium „Datenflexibilität“, da die Bewertung der Flexibilität der Datenschemata in dem jeweiligen Szenario und der Umgang der Datenbanksysteme damit im Fokus stehen soll.
- Für den Aspekt „Datenvolumen und Datenvielfalt“ wurde die Kategorie „Datencharakteristika“ als Kriterium übernommen, da beide Aspekte darunter gut zusammengefasst werden können.

- „Kosten“ und „Lizenzierung“ unter der Kategorie „Wirtschaftliche Aspekte“ wurden ebenso in ein Kriterium zusammengefasst.
- Das Kriterium der Skalierbarkeit zielt darauf ab, die Flexibilität der Datenbanksysteme in Bezug auf die horizontale Skalierbarkeit zu bewerten, da es hier sonst an fester Grundlage für den Vergleich fehlen würde.

6.2 Schritt 2: Gewichtung

Die Gewichtung erfolgt in einer „dreistufigen Skala“, nach der die Wichtigkeit jedes Kriteriums für den bestimmten Anwendungsfall mit einem bis drei Punkten gewichtet wird (Kühnapfel, 2021, S. 43):

- 1 Punkt: das Kriterium hat für den Anwendungsfall wenig Bedeutung
- 2 Punkte: das Kriterium hat für den Anwendungsfall mittlere Bedeutung
- 3 Punkte: das Kriterium hat für den Anwendungsfall hohe Bedeutung

Somit wird festgelegt, welche Kriterien bei der Auswahl des Datenbanksystems die größte Rolle spielen. Die „Datenflexibilität“ für Banking-Daten wurde beispielsweise mit einem Punkt bewertet und entsprechend mit 3,13 % gewichtet, da Banken in der Regel eher über strukturierte Daten verfügen und eine Flexibilität in dem Datenschema keinen signifikanten Mehrwert bieten würde.

In Bezug auf die Kategorie Datenintegrität wurde zum Beispiel darauf geachtet, den Bezug auf das CAP-Theorem beizubehalten. So wurde im E-Commerce-Szenario eine Bewertung von jeweils drei Punkten für Verfügbarkeit und Partitionstoleranz vergeben. Die Konsistenz wurde hierbei mit einem Punkt bewertet, was bedeutet, dass auch eine „weiche Konsistenz“ akzeptabel wäre. Eine Alternative wäre die Vergabe von null Punkten für die Konsistenz, was zur Folge hätte, dass das Kriterium aus dem Gesamtergebnis ausgeschlossen wäre. Diese Option wurde jedoch verworfen, da im Kapitel „Vergleichende Analyse“ bereits darauf hingewiesen wurde, dass die Konsistenz oft eine entscheidende Anforderung ist, die unabhängig vom spezifischen Anwendungsfall gilt.

6.3 Schritt 3: Bewertung

Die Bewertung der DBMS wird in einer „Drei-Punkte-Skala“ ähnlich wie in (Kühnapfel, 2021, S. 74) durchgeführt:

- 1 Punkt: die DBMS erfüllt das Kriterium nicht in befriedigendem Maße
- 2 Punkte: die DBMS erfüllt das Kriterium grundsätzlich, wenn auch mit Mängeln
- 3 Punkte: die DBMS erfüllt das Kriterium vollständig

Die Bewertung erfolgt dabei basierend auf den Eigenschaften der DBMS und ist unabhängig von dem Anwendungsfall. So können beide DBMS anhand der einzelnen Kriterien-Bewertungen oder des Gesamt-Scores verglichen werden.

6.4 Banking Nutzwertanalyse

In diesem Abschnitt ist die Nutzwertanalyse für den Anwendungsfall der Banking-Daten dargestellt. MongoDB hat, wie in Abbildung 6.1 zu sehen, einen 35,09 % höheren Gesamt-Score als MySQL. Das heißt, laut den Ergebnissen der Nutzwertanalyse ist MongoDB als ein verteiltes DBMS den Anforderungen einer Bank besser gerecht.

Banking							
		Bewertung	Gewichtung	MySQL		MongoDB	
				Punktwert	Teilnutzenwert	Punktwert	Teilnutzenwert
1	Datenflexibilität	1	3,70%	2	0,07	3	0,11
2	Datenintegrität						
2.1	Konsistenz	3	11,11%	3	0,33	2	0,22
2.2	Verfügbarkeit	1	3,70%	3	0,11	3	0,11
2.3	Partitionstoleranz	3	11,11%	1	0,11	3	0,33
3	Leistung						
3.1	Lese- und Schreibgeschwindigkeit	2	7,41%	2	0,15	3	0,22
3.2	Skalierbarkeit (flexible horizontale Skalierbarkeit)	2	7,41%	1	0,07	3	0,22
4	Datencharakteristika (Datenvolumen, Datenvielfalt)	2	7,41%	2	0,15	3	0,22
5	Sicherheit (Zugriffskontrollen , CIA-Triade, etc.)	3	11,11%	3	0,33	3	0,33
6	Datensicherung (Backup & Wiederherstellung)	3	11,11%	3	0,33	3	0,33
7	Datenverteilung	1	3,70%	1	0,04	3	0,11
8	Echtzeitverarbeitung	2	7,41%	1	0,07	3	0,22
9	Wirtschaftliche Aspekte						
9.1	Anschaffungskosten & Lizenzierung	3	11,11%	2	0,22	3	0,33
9.2	Ökosystem & Support	1	3,70%	3	0,11	2	0,07
		27	1	27	2,11	37	2,85

Abbildung 6.1: Nutzwertanalyse Banking

6.5 E-Commerce Nutzwertanalyse

Die Nutzwertanalyse für den Anwendungsfall E-Commerce zeigt, wie in Abbildung 6.2 zu sehen, ein um 49,21 % besseres Ergebnis für MongoDB als bei MySQL. Dies deutet wiederum darauf hin, dass MongoDB womöglich besser als MySQL für die Speicherung der Daten eines Online-Shops geeignet wäre.

E-Commerce							
Nr.	Kriterium	Bewertung	Gewichtung	MySQL		MongoDB	
				Punktwert	Teilnutzenwert	Punktwert	Teilnutzenwert
1	Datenflexibilität	3	9,38%	2	0,19	3	0,28
2	Datenintegrität						
2.1	Konsistenz	1	3,13%	3	0,09	2	0,06
2.2	Verfügbarkeit	3	9,38%	3	0,28	3	0,28
2.3	Partitionstoleranz	3	9,38%	1	0,09	3	0,28
3	Leistung						
3.1	Lese- und Schreibgeschwindigkeit	2	6,25%	2	0,13	3	0,19
3.2	Skalierbarkeit (flexible horizontale Skalierbarkeit)	3	9,38%	1	0,09	3	0,28
4	Datencharakteristika (Datenvolumen, Datenvielfalt)	3	9,38%	2	0,19	3	0,28
5	Sicherheit (Zugriffskontrollen, CIA-Triade, etc.)	3	9,38%	3	0,28	3	0,28
6	Datensicherung (Backup & Wiederherstellung)	2	6,25%	3	0,19	3	0,19
7	Datenverteilung	2	6,25%	1	0,06	3	0,19
8	Echtzeitverarbeitung	3	9,38%	1	0,09	3	0,28
9	Wirtschaftliche Aspekte						
9.1	Anschaffungskosten & Lizenzierung	3	9,38%	2	0,19	3	0,28
9.2	Ökosystem & Support	1	3,13%	3	0,09	2	0,06
		32	1	27	1,97	37	2,94

Abbildung 6.2: Nutzwertanalyse E-Commerce

7 Fazit und Ausblick

7.1 Zusammenfassung

Das Ziel dieser Bachelorarbeit war, eine Nutzwertanalyse zu konzipieren, die Entscheidungsfindern bei der Wahl der DBMS behilflich sein könnte. Hierfür wurden in Kapitel 2 die wichtigsten theoretischen Begriffe wie das „CAP-Theorem“ oder die „CIA-Triade“ erläutert. Denn diese sind ein Teil der Sammlung aller Vergleichskriterien, die in Kapitel 3 aufgeführt wurden. Im nächsten Schritt wird eine Anforderungsanalyse durchgeführt: Interviews mit einem Anwender und zwei IT-Operations-Mitarbeitern in Kapitel 4. Das Ergebnis hier waren User Storys, die darstellen, was den Nutzern von Datenbanksystemen vor allem während der aktiven Nutzung wichtig ist. Mit den Erkenntnissen aus den Kapiteln 3 und 4 wurde im nächsten Kapitel eine vergleichende Analyse durchgeführt. MongoDB und MySQL wurden in drei Etappen miteinander verglichen: funktionale Anforderungen, nicht funktionale Anforderungen und Performance Analyse. Die Anforderungen sind die etwas angepassten Vergleichskriterien aus Kapitel 2. Bei dem Vergleich wurde schon Bezug auf die Anwendungsfälle genommen, da es bei den Anforderungen in diesem Kapitel bereits darauf ankommt mit welchen Daten das DBMS arbeiten soll. Die Ergebnisse in diesem Abschnitt wurden dann in das nächste und finale Kapitel mitgenommen. Die Nutzwertanalyse wurde basierend auf Erkenntnissen aus den Kapiteln 3 bis 5 konzipiert. Dabei sind zwei Nutzwertanalysen entstanden: jeweils eine für die Anwendungsfälle „Banking Daten“ und „E-Commerce Daten“.

7.2 Fazit

Durch die komplette Arbeit – ab Kapitel 3 bis Kapitel 5 – sieht man, dass MongoDB im Vergleich zu MySQL oft einen Vorsprung hat. Sei es die flexible horizontale Skalierbarkeit oder die Möglichkeit problemlos mit Big Data Anwendungen zu arbeiten. Auch in den Performance Tests ist zu sehen, wie MongoDB um einiges besser mit den CRUD-Operationen zurechtkommt und sowohl im Durchsatz als auch in den Latenzzahlen MySQL überholt. Und zu guter Letzt „gewinnt“ MongoDB in dem Gesamt-Score sowohl bei den Banking-Daten als auch E-Commerce-Daten. Nun bestand das Ziel nicht darin herauszufinden, welche DBMS „besser ist“, sondern wie ein solcher Vergleich systematisch angegangen und dargestellt werden kann. Und gerade, weil es so einfach ist, durch die ganze Bachelorarbeit den Vorsprung von MongoDB zu beobachten, weist dies darauf hin, dass ein Vergleich basierend auf vorher festgelegten Kriterien in Form einer Nutzwertanalyse tatsächlich zur Entscheidungsfindung beitragen kann.

7.3 Ausblick

An dem Thema kann noch mit einigen weiteren Aspekten und Vorhaben weitergearbeitet werden. So könnte man die entstandenen Nutzwertanalysen den Stakeholdern aus Kapitel 4 oder im Idealfall Entscheidungsfindern vorlegen, um ein Feedback einzuholen. Somit könnte das Nutzen der Nutzwertanalyse für die Entscheider bewertet werden. Zudem könnte man weiter in die Zukunft schauen und in den Vergleich hybride Variationen von SQL- und NoSQL-Datenbanken einbeziehen. Hier könnte vor allem auch die Performance Analyse interessante Ergebnisse liefern. Grundsätzlich könnte man jeden der Kapitel 3 - 5 weiter ausbauen und tiefer in das Thema einsteigen. Eine weitere Idee wäre, die Anforderungsanalyse so zu gestalten, dass diese in drei Etappen durchgeführt wird: Einzelinterviews, Workshop und eine Online-Umfrage zum Anschluss. So hätte man die Möglichkeit, auf die Anforderungen der Stakeholder noch genauer einzugehen.

Literaturverzeichnis

- [Akter und Wamba 2016] AKTER, Shahriar ; WAMBA, Samuel F.: Big data analytics in E-commerce: a systematic review and agenda for future research. In: *Electronic Markets* 26 (2016), 3, Nr. 2, S. 173–194. – URL <https://doi.org/10.1007/s12525-016-0219-0>
- [Capris u. a. 2022] CAPRIS, Ticiana ; MELO, Pedro ; GARCIA, Nuno M. ; PIRES, Ivan M. ; ZDRAVEVSKI, Eftim: Comparison of SQL and NoSQL databases with different workloads: MongoDB vs MySQL evaluation. In: *2021 International Conference on Data Analytics for Business and Industry (ICDABI)* (2022), 10, S. 214–218. – URL <https://doi.org/10.1109/icdabi56818.2022.10041513>
- [Fasel 2016] FASEL, Daniel: *Big Data Grundlagen, Systeme und Nutzungspotenziale*. Springer Vieweg Wiesbaden, 2016. – ISBN 978-3-658-11588-3
- [Foote 2023] FOOTE, Keith D.: *A Brief History of Database Management*. 3 2023. – URL <https://www.dataversity.net/brief-history-database-management/>. – Zugriff am 17. Februar 2025
- [GeeksforGeeks 2025] GEEKSFORGEEKS: *MongoDB \$inc Operator*. 2025. – URL <https://www.geeksforgeeks.org/mongodb-increment-operator-inc/>. – Zugriff am 10. Januar 2025
- [Ghavami 2021] GHAVAMI, Peter: *Big Data management*. de Gruyter, 2021
- [Grande 2014] GRANDE, Marcus: *100 Minuten für Anforderungsmanagement: Kompaktes Wissen nicht nur für Projektleiter und Entwickler*. Springer Vieweg, 2014. – ISBN 978-3-658-06434-1
- [Herrmann 2022] HERRMANN, Andrea: *Grundlagen der Anforderungsanalyse Standardkonformes Requirements Engineering*. Springer Vieweg, 2022. – ISBN 978-3-658-35459-6

- [Kaufmann und Meier 2023] KAUFMANN, Michael ; MEIER, Andreas: *SQL- NoSQL-Datenbanken*. Springer Vieweg Berlin, 2023. – ISBN 978-3-662-67092-7
- [Klettke u. a. 2016] KLETTKE, Meike ; STÖRL, Uta ; SCHERZINGER, Stefanie: Herausforderungen bei der Anwendungsentwicklung mit schema-flexiblen NoSQL-Datenbanken. In: *HMD Praxis der Wirtschaftsinformatik* 53 (4) (2016), S. 428–442
- [Kleuker 2023] KLEUKER, Stephan: *Grundkurs Datenbankentwicklung: Von der Anforderungsanalyse Zur Komplexen Datenbankanfrage*. Springer Vieweg, 2023. – ISBN 978-3-658-43022-1
- [Kühnapfel 2021] KÜHNAPFEL, Jörg B.: *Scoring und Nutzwertanalysen*. Springer Gabler, 2021. – ISBN 978-3-658-34809-0
- [Liu u. a. 2022] LIU, Peng ; DENG, Chunyu ; WANG, Dazhong: Design and Application of Distributed Real-time Database System for Massive Data. In: *IEEE* (2022), 11, S. 114–117. – URL <https://doi.org/10.1109/iccsmt58129.2022.00031>
- [Meier 2010] MEIER, Andreas: *Relationale und postrelationale Datenbanken*. Springer Berlin, 2010. – ISBN 978-3-642-05255-2
- [Meier 2017] MEIER, Andreas: *Werkzeuge der digitalen Wirtschaft: Big Data, NoSQL Co*. Springer Vieweg Wiesbaden, 2017. – ISBN 978-3-658-20337-5
- [MongoDB 2024a] MONGODB: *MongoDB Manual v9.0*. 2024. – URL <https://dev.mysql.com/doc/refman/9.0/en/>. – Zugriff am 9. Februar 2025
- [MongoDB 2024b] MONGODB: *Role-Based access control in Self-Managed deployments*. 2024. – URL <https://www.mongodb.com/docs/manual/core/authorization/>. – Zugriff am 9. Februar 2025
- [MongoDB 2025] MONGODB: *MongoDB Pricing*. 2025. – URL <https://www.mongodb.com/pricing>. – Zugriff am 17. Februar 2025
- [Mukherjee u. a. 2015] MUKHERJEE, Shubham ; SEN, Pritam ; BORA, Sudeshna ; PRADHAN, Chittaranjan: SQL Injection: A sample review. In: *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (2015), 7, S. 1–7. – URL <https://doi.org/10.1109/icccnt.2015.7395166>
- [Oracle 2025a] ORACLE: *MySQL 8.4 Reference Manual*. 2025. – URL <https://dev.mysql.com/doc/refman/8.4/en/commit.html>. – Zugriff am 9. Februar 2025

- [Oracle 2025b] ORACLE: *MySQL Products*. 2025. – URL <https://www.mysql.com/products/>. – Zugriff am 17. Februar 2025
- [Rahm u. a. 2015] RAHM, Erhard ; SAAKE, Gunter ; SATTLER, Kai-Uwe: *Verteiltes und paralleles Datenmanagement: Von verteilten Datenbanken zu Big Data und Cloud*. Springer Vieweg Berlin, 2015. – ISBN 978-3-642-45241-3
- [Statista 2024] STATISTA: *Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2023, with forecasts from 2024 to 2028 (in zettabytes) [Graph]*. 11 2024. – URL <https://www.statista.com/statistics/871513/worldwide-data-created>. – Zugriff am 17. Februar 2025
- [Trempp 2022] TREMP, Hansruedi: *Agile objektorientierte Anforderungsanalyse: Planen – Ermitteln – Analysieren – Modellieren – Dokumentieren – Prüfen*. Springer Vieweg, 2022. – ISBN 978-3-658-37193-7
- [Wiese 2015] WIESE, Lena: *Advanced Data Management*. De Gruyter Oldenbourg, 2015
- [Yedilkhan u. a. 2023] YEDILKHAN, Didar ; MUKASHEVA, Assel ; BISSENGALIYEVA, Dariya ; SUYNULAYEV, Yerulan: Performance Analysis of Scaling NoSQL vs SQL: A Comparative Study of MongoDB, Cassandra, and PostgreSQL. In: *2021 IEEE International Conference on Smart Information Systems and Technologies (SIST) (2023)*, 5, S. 479–483. – URL <https://doi.org/10.1109/sist58284.2023.10223568>

A Anhang

A.1 Performance-Test Ergebnisse: Banking

		Durchsatz MongoDB (Op./ Sek.)	Durchsatz MySQL (Op./ Sek.)	Latenz MongoDB (Sek./ Abfr.)	Latenz MySQL (Sek./ Abfr.)
B1.1	simple read test	4,90	2,92	0,20	0,34
B1.2	complex read test	0,63	0,42	1,60	2,40
B2.1	simple insert test	11455,20	17306,58	0,00	0,00
B2.2	batch insert test	158189,31	58687,86	0,00	0,00
B3.1	simple update test	4,65	0,26	0,22	3,86
B3.2	complex update test	0,30	0,26	3,36	3,83
B4.1	simple delete test	1,01	0,26	0,99	3,80
B4.2	batch delete test	43,36	26,17	0,02	0,04

Tabelle A.1: Performance-Test Ergebnisse: Banking

A.2 Performance-Test Ergebnisse: E-Commerce

		Durchsatz MongoDB (Op./ Sek.)	Durchsatz MySQL (Op./ Sek.)	Latenz MongoDB (Sek./ Abfr.)	Latenz MySQL (Sek./ Abfr.)
E1.1	simple read test	10,64	7,07	0,09	0,14
E1.2	complex read test	6,03	4,32	0,17	0,23
E2.1	simple insert test	11207,81	7071,84	0,00	0,00
E2.2	batch insert test	68993,21	61880,05	0,00	0,00
E3.1	simple update test	165,50	73,55	0,01	0,01
E3.2	complex update test	73,73	52,55	0,01	0,02
E4.1	simple delete test	3,60	2,84	0,28	0,35
E4.2	batch delete test	316,42	286,14	31,60	34,95

Tabelle A.2: Performance-Test Ergebnisse: E-Commerce

A.3 CD Anhang

Auf der dieser Bachelorarbeit beigelegten CD sind folgende Dateien zu finden.

Anforderungsanalyse:

- Fragenkatalog
- Transkripte der Interviews

Performance-Tests:

- Python Quell-Code für die Generierung der Testdaten

- CSV-Dateien mit den Testdaten
- Python Quell-Code für die Performance-Tests

Nutzwertanalyse:

- Excel-Datei mit der Nutzwertanalyse

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort

Datum

Unterschrift im Original