**161 matches from 63 sources, of which 25 are online sources.**

*PlagLevel:* **2.9%**/3.1%

- [0] (8 matches, **0.5%**) from a PlagScan document dated 2022-02-10 19:06
- [1] (5 matches, **0.1%**) from a PlagScan document dated 2022-06-10 08:08
- [2] (6 matches, **0.2%**) from d-nb.info/1106453026/34
- [3] (5 matches, **0.2%**) from d-nb.info/987424483/34
- [4] (5 matches, **0.2%**) from epdf.pub/lehrbuch-der-personalpsychologie.html
- [5] (4 matches, **0.1%**) from a PlagScan document dated 2021-07-13 09:14
- [6] (5 matches, **0.1%**/0.2%) from a PlagScan document dated 2020-10-22 15:45
- [7] (3 matches, **0.2%**) from a PlagScan document dated 2022-07-18 08:19
- [8] (3 matches, **0.2%**) from a PlagScan document dated 2020-09-15 10:10
- [9] (3 matches, **0.2%**) from epdf.pub/download/nosql-einstieg-in-die-...ichtrelationaler-web-20-datenbanken.html
- [10] (3 matches, **0.1%**/0.3%) from depositonce.tu-berlin.de/bitstream/11303/5937/4/westphal_daniel.pdf
- [11] (3 matches, **0.2%**) from a PlagScan document dated 2020-09-15 10:10
  (+ 15 documents with identical matches)
- [27] (3 matches, **0.2%**) from aws.amazon.com/de/startups/start-building/how-to-choose-a-database/
- [28] (3 matches, **0.2%**) from tutorialsdojo.com/amazon-quantum-ledger-database-qldb/
- [29] (2 matches, **0.1%**) from a PlagScan document dated 2019-01-17 09:37
- [30] (3 matches, **0.1%**) from a PlagScan document dated 2022-07-05 07:35
- [31] (3 matches, **0.1%**/0.2%) from cloudacademy.com/course/databases-saa-c03/amazon-quantum-ledger-database-qldb/
- [32] (2 matches, **0.1%**) from a PlagScan document dated 2023-01-04 14:40
- [33] (3 matches, **0.0%**/0.1%) from www.springest.net/fast-lane-institute-fo...-designing-databases-on-aws-pd-db-online
- [34] (2 matches, **0.1%**) from a PlagScan document dated 2022-03-16 08:04
  (+ 1 documents with identical matches)
- [36] (2 matches, **0.1%**) from a PlagScan document dated 2020-07-29 11:21
- [37] (2 matches, **0.1%**) from aws.amazon.com/de/products/databases/
- [38] (2 matches, **0.1%**) from aws.amazon.com/qldb/
- [39] (2 matches, **0.1%**) from aws.amazon.com/de/products/databases/
- [40] (2 matches, **0.1%**) from www.tecracer.com/blog/2022/06/what-is-a-quantum-ledger-database.html
- [41] (2 matches, **0.1%**) from 6sense.com/tech/database/amazon-quantum-ledger-database-qldb-market-share
- [42] (2 matches, **0.1%**) from medium.com/konzentrik/streaming-wohin-ge...-nodejs-microservices-dabei-af75d6077c4c
- [43] (2 matches, **0.1%**) from a PlagScan document dated 2022-01-12 11:05
- [44] (2 matches, **0.0%**/0.1%) from aws.amazon.com/blogs/database/replicate-...zon-elasticache-for-redis-using-aws-dms/
  (+ 2 documents with identical matches)
- [47] (1 matches, **0.1%**) from d1.awsstatic.com/whitepapers/de_DE/aws-overview.pdf?did=wp_card&trk=wp_card
- [48] (2 matches, **0.0%**) from a PlagScan document dated 2021-12-21 07:49
- [49] (1 matches, **0.1%**) from a PlagScan document dated 2020-08-07 07:58
- [50] (2 matches, **0.1%**) from core.ac.uk/download/pdf/11589777.pdf
- [51] (1 matches, **0.1%**) from appmaster.io/de/blog/auth0-authentifizierung
- [52] (1 matches, 0.0%0.1%) from docs.aws.amazon.com/keyspaces/latest/devguide/cassandra-apis.html
- [53] (1 matches, **0.1%**) from a PlagScan document dated 2023-02-23 13:33
- [54] (1 matches, **0.0%**) from a PlagScan document dated 2022-04-05 11:38
- [55] (1 matches, **0.0%**) from a PlagScan document dated 2021-08-11 15:24
- [56] (1 matches, **0.0%**) from a PlagScan document dated 2021-01-25 15:15
- [57] (1 matches, **0.0%**) from cloud.google.com/memorystore?hl=de
- [58] (1 matches, **0.0%**) from d-nb.info/1053959885/34
- [59] (1 matches, **0.0%**) from a PlagScan document dated 2022-11-21 06:16
- [60] (1 matches, **0.0%**) from a PlagScan document dated 2020-07-26 18:08
- [61] (1 matches, **0.0%**) from a PlagScan document dated 2019-02-06 15:01
- [62] (1 matches, **0.0%**) from publikationen.bibliothek.kit.edu/1000044973/3392992

**Settings**

Sensitivity: *Medium*
Bibliography: *Consider text*
Citation detection: *Reduce PlagLevel*
Whitelist: --

## Analyzed document

==========1/78==========
UNIVERSITÄT
FAKULTÄT
BSc. THESIS
Name Nachname
VERGLEICH VON MONGODB MIT ANDEREN DATENBANKEN
FACHBEREICH XXX
15.02.2023
==========2/78==========
ABSTRACT
Diese Arbeit vergleicht MongoDB mit anderen populären Datenbanken, einschließlich Aurora, Amazon ElastiCache und MemoryDB. Ziel ist es, die Stärken und Schwächen von MongoDB im Vergleich zu diesen anderen Datenbanken zu bewerten und einen Einblick in die am besten geeignete Datenbankauswahl für verschiedene Arten von Anwendungen zu geben. Die Arbeit beginnt mit einem Überblick über MongoDB und seine einzigartigen Funktionen, einschließlich seines dokumentenorientierten Datenmodells und seiner Skalierbarkeitsfähigkeiten. Anschließend wird MongoDB in Bezug auf Leistung, Merkmale und Funktionalität mit jeder der anderen Datenbanken verglichen. Dazu gehört eine Analyse der Fähigkeit jeder Datenbank, große Datensätze zu verarbeiten, die Unterstützung für Indizierung, Aggregation und Datenvisualisierung sowie das Niveau der angebotenen SQL-Unterstützung. Die Arbeit wird auch die Stärken und Schwächen jeder Datenbank in Bezug auf ihre Datenspeicherungs- und Abrufanforderungen berücksichtigen. Dies umfasst eine Bewertung der Datenstrukturen und -modelle, die von jeder Datenbank verwendet werden, sowie der Flexibilität und Skalierbarkeit jeder Datenbank bei der Anpassung an sich ändernde Datenanforderungen. Abschließend fasst die Arbeit die Ergebnisse des Vergleichs zusammen und gibt Empfehlungen für die am besten geeignete Datenbankauswahl für verschiedene Arten von Anwendungen. Dazu gehören eine Diskussion der Kompromisse bei der Auswahl der einzelnen Datenbanken und eine Bewertung der Stärken und Schwächen jeder Datenbank in Bezug auf die spezifischen Anforderungen der Anwendung. Durch den Vergleich soll ein besseres Verständnis dafür gewonnen werden, wann es sinnvoll ist, MongoDB oder andere populäre Datenbanken zu verwenden.
Schlüsselwörter: Datenbanken, MangoDB, SQL
2
==========3/78==========
DANKSAGUNGEN
Zuallererst möchte ich meinem Betreuer Prof. Dr. xx meinen aufrichtigen Dank für seine Betreuungsberatung, Geduld, Motivation und nützlichen Anregungen für diese Arbeit aussprechen.
Außerdem danke ich ….
Nicht zuletzt möchte ich den Mitgliedern der Dissertationsjury Prof. Dr. …, für ihre Anregungen und Korrekturen danken.
3
==========4/78==========
INHALTSVERZEICHNIS

TABELLENVERZEICHNIS



ABBILDUNGSVERZEICHNIS




ABKÜRZUNGSVERZEICHNIS
ACID : Atomicity, consistency, isolation, durability
ARIS : Architecture of Integrated Information Systems
COBO
L: Common Business Oriented Language
DBMS : Datenbankmanagementsystem
DDL : Data Definition Language
DML : Data Manipulation Language
EPK : Ereignisgesteuerte Prozesskette
ERD : Entity-Relationship-Modell
SQL : Structured Query Language
UML : Unified Modeling Language


# 1. EINFÜHRUNG
## 1.1. Problemstellung
Das Team XX des Betriebs XX hat MongoDB als Datenbank-Lösung für eine Anwendung eingesetzt, jedoch traten Integrationsprobleme zwischen MongoDB und AWS auf. Diese Probleme betreffen die manuelle Skalierung, doppelte Datenpflege, schwierige Zugriffsverwaltung,

komplexe Verknüpfung der beiden Systeme, Caching-Probleme und organisatorischer Overhead. Aufgrund dieser Herausforderungen sucht das Team nun nach alternativen Datenbank-Lösungen, die besser auf ihre Bedürfnisse und Anforderungen abgestimmt sind und eine höhere Leistungsfähigkeit und Zuverlässigkeit bieten. Das Ziel des Teams ist es, eine alternative Datenbank-Lösung zu finden, die die Integrationsprobleme zwischen MongoDB und AWS löst und die Effektivität der Anwendung verbessert.

### 1.2. Zielsetzung und Beiträge der Arbeit

In diesem Kontext soll untersucht werden, welche Alternativen zu MongoDB für den vorliegenden Use Case zur Verfügung stehen.[7] Der Use Case beinhaltet das Laden von Kundendaten, Rechnungsdaten, Kunden-Selektion, Vorteilsdaten oder allen langlebigen Daten, wenn ein Kunde sich einloggt. Es ist das Ziel, durch die Untersuchung von Alternativen die geeignetste Datenbank für diesen Use Case zu ermitteln.

Die Zielsetzung dieser Arbeit ist es, alternative Datenbank-Lösungen zu MongoDB zu finden, und eine umfassende Analyse der Ähnlichkeiten und Unterschiede zwischen MongoDB und anderen alternativen Datenbanken,





einschließlich Aurora, Amazon ElastiCache und MemoryDB bereitzustellen. Dabei werden die Datenbanken einschließlich, aber nicht beschränkt auf ihre jeweiligen Architekturen, Datenmodelle, Leistung, Sicherheit, Benutzerfreundlichkeit, Community- und Anbieterunterstützung, Bereitstellungsoptionen und Anwendungsfälle miteinander verglichen.

Die konkreten Beiträge der Arbeit sind:

- Identifikation von Alternativen zu MongoDB für den genannten Use Case
- Untersuchung der Architektur und des Designs von MongoDB und anderen alternativen Datenbanken und ihrer Unterschiede.
- Vergleich der Leistung von MongoDB und anderen alternativen Datenbanken in Bezug auf die Abfrageverarbeitungsgeschwindigkeit und die Datenabrufzeit
- Bewertung der Skalierbarkeit von MongoDB und anderen alternativen Datenbanken im Hinblick auf ihre Fähigkeit, mit steigenden Datenmengen umzugehen.
- Vergleich der Zuverlässigkeit von MongoDB und anderen alternativen Datenbanken im Hinblick auf ihre Fähigkeit, mit Datenverlust und -wiederherstellung umzugehen.
- Welche von den Datenbanken ist für den Entwicklern angenehmer?

Es gibt einige Studien, die Alternativen zu MongoDB untersucht haben[1]–[7]. Im Vergleich zu diesen Arbeiten befasst sich die vorliegende Arbeit speziell mit der Identifikation von Alternativen zu MongoDB für einen spezifischen Use Case. Die Analyse wird mittels des Analytischen





Hierarchieprozesses (AHP) durchgeführt und bezieht qualitative und quantitative Kriterien ein. Durch diese Herangehensweise können potenzielle Alternativen zu MongoDB auf Basis der spezifischen Anforderungen des Use Cases identifiziert werden.

### 1.3. Methodisches Vorgehen

Bei der Bewertung von MongoDB im Vergleich zu anderen alternativen Datenbanken können verschiedene Evaluationskriterien berücksichtigt werden, je nachdem, welche Aspekte für die jeweilige Anwendung oder das jeweilige Projekt am wichtigsten sind. Evaluationskriterien sind wichtig, um die Stärken und Schwächen der verschiedenen Alternativtechnologien zu identifizieren und zu bewerten.[4] Hier sind einige wichtige Evaluationskriterien:

- Datenmodell: MongoDB verwendet ein dokumentenorientiertes Datenmodell, während relationale Datenbanken das relationale Modell verwenden. Das Datenmodell kann Auswirkungen auf die Flexibilität, Skalierbarkeit und Leistung der Datenbank haben.

•Skalierbarkeit: MongoDB ist als horizontal skalierbare Datenbank konzipiert, was bedeutet, dass sie in der Lage ist, durch Hinzufügen weiterer Server und Replikation von Daten zu wachsen.[0] [3] [5] Traditionelle relationale Datenbanken können auch horizontal skalierbar sein, erfordern jedoch normalerweise mehr Aufwand und Ressourcen.
•Leistung: MongoDB kann in einigen Szenarien eine höhere Leistung bieten als relationale Datenbanken. Insbesondere wenn es darum geht, große Mengen an unstrukturierten oder semi-strukturierten Daten zu speichern und zu verarbeiten, kann MongoDB schneller


sein.[59] Bei streng strukturierten Daten oder Transaktionen mit hohen Anforderungen an die Integrität und Konsistenz der Daten können relationale Datenbanken jedoch eine bessere Wahl sein.
•Abfragesprache: MongoDB verwendet eine eigene Abfragesprache namens "[29] MongoDB Query Language (MQL)", während relationale Datenbanken SQL (Structured Query Language) verwenden. Die Abfragesprache kann Auswirkungen auf die Benutzerfreundlichkeit und die Möglichkeiten der Abfrageoptimierung haben.
•Verfügbarkeit und Zuverlässigkeit: Sowohl MongoDB als auch relationale Datenbanken bieten Optionen für hohe Verfügbarkeit und Zuverlässigkeit, aber die Implementierung kann je nach Datenbank unterschiedlich sein.
•Community und Ökosystem: MongoDB hat eine aktive Entwickler-Community und ein umfangreiches Ökosystem von Tools und Anwendungen. Relationale Datenbanken haben jedoch auch eine starke Community und ein breites Anwendungsspektrum.
•Lizenz und Kosten: MongoDB hat eine Open-Source-Version und eine kommerzielle Version mit erweiterten Funktionen und Support. Relationale Datenbanken haben ebenfalls Open-Source-Versionen und kommerzielle Versionen mit unterschiedlichen Lizenzmodellen und Kosten.
Diese Evaluationskriterien sind entscheidend, da sie die wichtigsten Aspekte einer Datenbank-Technologie abdecken, die für eine effektive Datenverwaltung und -verarbeitung notwendig sind.


Für den Vergleich wird der Analytische Hierarchieprozess (AHP) [8] verwendet. Es ist eine Methode, die sowohl qualitative als auch quantitative Kriterien berücksichtigt und diese zu einer Entscheidung zusammenführt.[50] Im Kontext des Vergleichs zwischen MongoDB und anderen populären Datenbanken werden zunächst die zuvor genannten Kriterien beschrieben und gewichtet. Anschließend werden die verschiedenen Datenbanken anhand der Kriterien bewertet und die Gesamtwerte für jede Option berechnet, um schließlich die am besten geeignete Datenbank für das Ziel des Vergleichs zu identifizieren. Organisationen, die eine Datenbank auswählen möchten, die ihren besonderen Bedürfnissen und Anforderungen gerecht wird, werden aus dem Vergleich nützliche Erkenntnisse gewinnen. Die Ergebnisse dieser Arbeit werden auch die bestehende Datenbank-Wissensbasis ergänzen und Organisationen dabei unterstützen, fundierte Entscheidungen zur Datenbankauswahl zu treffen.
Einige Gründe, warum der AHP im Vergleich von MongoDB zu anderen DB-Technologien verwendet werden, sind:
□Berücksichtigung von mehreren Kriterien: Der AHP eignet sich gut für komplexe Entscheidungen, bei denen mehrere Kriterien bewertet werden müssen. Im Falle des Vergleichs von Datenbank-Technologien sind mehrere Kriterien wie Skalierbarkeit, Performance, Flexibilität, Sicherheit und Community und Support wichtig, und der AHP kann helfen, diese Kriterien zu bewerten.
□Konsistente Entscheidungen: Der AHP kann helfen, Entscheidungen zu treffen, die konsistent und objektiv sind. Durch die Bewertung

von Kriterien auf einer konsistenten Skala kann der AHP dazu beitragen, unklare oder subjektive Entscheidungen zu vermeiden.
□Berücksichtigung von Expertenmeinungen: Der AHP kann auch Expertenmeinungen berücksichtigen, um eine fundierte Entscheidung zu treffen. Im Falle des Vergleichs von Datenbank-Technologien kann dies bedeuten, dass Expertenmeinungen von Datenbankadministratoren oder Entwicklern einbezogen werden, um die Kriterien zu bewerten und Alternativen zu vergleichen.
Neben dem AHP gibt es auch andere Methoden, die für den Vergleich von Datenbank-Technologien verwendet werden können. Die Wahl der Methode hängt jedoch von den spezifischen Anforderungen und Bedürfnissen des Unternehmens ab. Einige andere mögliche Methoden sind:
□Nutzwertanalyse: Eine Methode, die bei der Entscheidungsfindung hilft, indem sie Kriterien gewichtet und Alternativen bewertet. Sie kann auch bei der Auswahl von Technologien hilfreich sein, indem sie die Leistungsfähigkeit, Kosten und andere Kriterien gewichtet und Alternativen vergleicht [8].
□Entscheidungsbaum-Analyse: Eine Methode, bei der Entscheidungen in einer Baumstruktur dargestellt werden, die verschiedene mögliche Ergebnisse und deren Wahrscheinlichkeiten darstellt. Es kann helfen, komplexe Entscheidungen zu visualisieren und zu analysieren [9].
□SWOT-Analyse: Eine Methode, die bei der Bewertung von Stärken, Schwächen, Chancen und Bedrohungen einer Technologie oder eines Unternehmens hilft. Es kann helfen, Vor- und Nachteile einer


Technologie zu identifizieren und Chancen und Risiken zu bewerten [10].
Die Wahl der Methode hängt davon ab, welche spezifischen Aspekte des Vergleichs betont werden sollen. Der AHP eignet sich jedoch gut für komplexe Entscheidungen, bei denen mehrere Kriterien bewertet werden müssen, und ist eine Methode, die sowohl qualitative als auch quantitative Kriterien berücksichtigt.

### 1.4. Aufbau der Arbeit

Die Arbeit ist wie folgt gegliedert. Abschnitt 2 gibt einen Überblick der aktuellen Situation. Abschnitt 3 vergleicht MANGODB und andere alternative Datenbanken. Abschnitt 4 zieht Schlussfolgerungen und weist auf weitere zukünftige Arbeiten hin.

## 2. STATUS QUO

### 2.1. Anwendungsfall

Das Team XX des Betriebs XX ist für die Entwicklung und Wartung einer kritischen Anwendung verantwortlich, die von Hunderten von Benutzern täglich genutzt wird. Die Anwendung ist in hohem Maße datenbankabhängig und verwendet MongoDB als Datenbanklösung. MongoDB ist eine NoSQL-Datenbank, die von MongoDB Inc. entwickelt wurde, um skalierbare und flexible Datenverwaltung für Anwendungen zu ermöglichen, die unstrukturierte und semistrukturierte Daten verarbeiten. Im Gegensatz zu relationalen Datenbanken organisiert MongoDB Daten in


Sammlungen von Dokumenten, die im JSON-Format gespeichert werden. Jedes Dokument besteht aus einer oder mehreren Feld-Wert-Paaren, die auch verschachtelt sein können. MongoDB kann große Datenmengen verarbeiten und bietet hohe Verfügbarkeit, Skalierbarkeit und Leistung. Die horizontale Skalierung wird durch die Verteilung der Daten auf mehrere Server und Replikation unterstützt, um Datenredundanz und Ausfallsicherheit zu gewährleisten.[29] [9] MongoDB bietet auch leistungsstarke Abfrage- und Indexierungsfunktionen, die es Entwicklern ermöglichen, komplexe Abfragen schnell und effizient auszuführen [11]. Obwohl MongoDB viele Vorteile bietet, hat es auch seine Nachteile. MongoDB ist

bekannt dafür, dass es beim Schreiben von Daten im Vergleich zu anderen Datenbanken langsamer ist. MongoDB erfordert auch eine sorgfältige Planung und Konfiguration, um sicherzustellen, dass es in der Produktion effektiv und effizient arbeitet [12], [13].
In der Vergangenheit hat das Team Integrationsprobleme zwischen MongoDB und AWS festgestellt. Diese Probleme umfassen fehlende Automatisierung der Skalierung in MongoDB, doppelte Datenpflege bei Änderungen in beiden Systemen, schwierige Zugriffsverwaltung und komplexe Verknüpfung der beiden Systeme. Zudem können Caching-Probleme und organisatorischer Overhead zusätzliche Herausforderungen darstellen. Das Team erkennt, dass diese Integrationsprobleme die Effektivität der Anwendung beeinträchtigen können und hat sich daher entschieden, alternative Datenbanklösungen in Betracht zu ziehen, die besser auf ihre Bedürfnisse und Anforderungen zugeschnitten sind.
16
==========17/78==========
Das Ziel der Arbeit besteht darin, eine alternative Datenbanklösung zu finden, die besser auf die Bedürfnisse der Anwendung und des Teams abgestimmt ist und eine höhere Leistungsfähigkeit und Zuverlässigkeit bietet. Die Implementierung der neuen Datenbanklösung soll reibungslos und ohne Unterbrechungen erfolgen, und die Benutzer und Entwickler der Anwendung sollen auf die neuen Funktionen und Optionen geschult werden.
Beteiligte Personen sind (i) das Team XX des Betriebs XX, das die Anwendung entwickelt und betreut und (ii) IT-Experten und Datenbankadministratoren, die das Team bei der Evaluierung und Implementierung der alternativen Datenbanklösung unterstützen, und (iii) Benutzer und Entwickler der Anwendung, die auf die neuen Datenbankoptionen und -funktionen vorbereitet werden müssen
Folgende Schritte zur Lösung des Anwendungsfalls werden befolgt:
□Analyse der Anforderungen: Das Team beginnt mit einer gründlichen Analyse der Anforderungen der Anwendung. Dazu gehören Faktoren wie die Art und Menge der Daten, die von der Anwendung verarbeitet werden, die Leistungsanforderungen der Anwendung, die Skalierbarkeit der Datenbank, die Sicherheitsanforderungen und die Compliance-Anforderungen.
□Identifikation geeigneter Datenbankoptionen: Basierend auf den Anforderungen identifiziert das Team eine Reihe von Datenbankoptionen, die für die Anwendung geeignet sein könnten.
17
==========18/78==========
Zu den möglichen Optionen gehören MySQL, Oracle, PostgreSQL, MSSQL und Redis. Jede Option wird anhand der Anforderungen des Teams und der Anwendung bewertet, um die am besten geeignete Option auszuwählen.
□Evaluierung der Optionen: Das Team wird jede Datenbankoption evaluieren und dabei verschiedene Kriterien berücksichtigen, darunter:
□Leistung und Skalierbarkeit: Wie schnell und effizient kann die Datenbank große Datensätze verarbeiten? Wie einfach ist es, die Datenbank zu skalieren, wenn die Datenmenge wächst?
□Zuverlässigkeit und Verfügbarkeit: Wie zuverlässig ist die Datenbank? Wie schnell kann die Datenbank wiederhergestellt werden, wenn es zu einem Ausfall kommt? Wie gut unterstützt die Datenbank Hochverfügbarkeit und Failover?
□Flexibilität und Anpassungsfähigkeit: Wie flexibel ist die Datenbank, wenn es um Änderungen an der Datenstruktur oder dem Datenmodell geht? Wie einfach ist es, die Datenbank an sich ändernde Anforderungen anzupassen?
□Datenmodell und -struktur: Wie gut passt das Datenmodell und die Datenstruktur der Datenbank zu den Anforderungen der Anwendung? Bietet die Datenbank Funktionen wie Indizierung,

Aggregation und Datenvisualisierung?
□Sicherheit und Compliance: Wie gut entspricht die Datenbank den Sicherheitsanforderungen und Compliance-Standards des Unternehmens? Bietet die Datenbank Funktionen wie Verschlüsselung, Audit-Trail und Zugriffskontrolle?
18
==========19/78==========
□Auswahl der besten Option: Basierend auf den Ergebnissen der Evaluierung wählt das Team die am besten geeignete Datenbankoption aus. Diese Option sollte die Anforderungen der Anwendung erfüllen und eine höhere Leistungsfähigkeit und Zuverlässigkeit bieten als MongoDB.
□Implementierung der neuen Datenbanklösung: Das Team wird die Implementierung der neuen Datenbanklösung planen und koordinieren. Die Implementierung sollte eng mit dem IT-Team und den Datenbankadministratoren koordiniert werden, um sicherzustellen, dass sie reibungslos und ohne Unterbrechungen erfolgt.
□Schulung der Benutzer und Entwickler: Das Team wird sicherstellen, dass die Entwickler und Benutzer der Anwendung geschult und auf die neuen Datenbankoptionen und -funktionen vorbereitet werden. Die Schulung sollte die Verwendung der neuen Datenbankoptionen und -funktionen sowie bewährte Methoden für die Datenbankverwaltung und -wartung umfassen.
□Testen und Validierung: Nach der Implementierung wird das Team die neue Datenbanklösung testen und validieren, um sicherzustellen, dass sie den Anforderungen der Anwendung entspricht und die erwarteten Leistungs- und Zuverlässigkeitsverbesserungen bietet.
□Überwachung und Wartung: Das Team wird die neue Datenbanklösung kontinuierlich überwachen und warten, um sicherzustellen, dass sie reibungslos und zuverlässig betrieben wird. Die Überwachung sollte die Leistung, Verfügbarkeit und Sicherheit
19
==========20/78==========
der Datenbank umfassen, um Probleme frühzeitig zu erkennen und zu beheben.
Das Ergebnis der Arbeit wird eine stabile und zuverlässige Anwendung sein, die den Bedürfnissen und Anforderungen des Teams und der Benutzer entspricht und eine höhere Leistungsfähigkeit und Zuverlässigkeit bietet. Das Team wird in der Lage sein, Verbindungsprobleme und Datenverluste zu vermeiden und eine bessere Benutzererfahrung zu bieten. Darüber hinaus wird das Team nun über eine bessere Datenbanklösung verfügen, die besser auf die Bedürfnisse der Anwendung und des Teams abgestimmt ist und die Anpassung an sich ändernde Anforderungen erleichtert. Die Implementierung der neuen Datenbanklösung sollte reibungslos und ohne Unterbrechungen erfolgen, und die Schulung der Benutzer und Entwickler wird sicherstellen, dass sie in der Lage sind, die neuen Funktionen und Optionen effektiv zu nutzen.[3] [32] Die kontinuierliche Überwachung und Wartung der neuen Datenbanklösung wird sicherstellen, dass die Anwendung reibungslos und zuverlässig betrieben wird.
Es ist auch wichtig zu beachten, dass das Team MongoDB in AWS deployt, daher müssen alternative Datenbanken auch von AWS angeboten werden. [36]
Als Fazit gilt, dass die Verbesserung der Stabilität und Zuverlässigkeit einer Anwendung durch die Auswahl einer geeigneten Datenbanklösung ein wichtiger Schritt ist, um sicherzustellen, dass die Anwendung reibungslos und zuverlässig betrieben wird. Die Auswahl der richtigen Datenbanklösung erfordert eine gründliche Analyse der Anforderungen und eine sorgfältige
20
==========21/78==========
Evaluierung der verfügbaren Optionen. Durch die Zusammenarbeit mit IT-Experten und Datenbankadministratoren kann das Team sicherstellen, dass die Implementierung der neuen Datenbanklösung reibungslos und effektiv

erfolgt. Die Schulung der Benutzer und Entwickler ist entscheidend, um sicherzustellen, dass sie in der Lage sind, die neuen Funktionen und Optionen effektiv zu nutzen.[3] [5] Die kontinuierliche Überwachung und Wartung der neuen Datenbanklösung wird sicherstellen, dass die Anwendung reibungslos und zuverlässig betrieben wird und Benutzererfahrungen verbessert werden. Insgesamt ist die Verbesserung der Stabilität und Zuverlässigkeit einer Anwendung durch die Implementierung einer geeigneten Datenbanklösung ein wichtiger Schritt, um sicherzustellen, dass die Anwendung den Anforderungen und Erwartungen der Benutzer entspricht.

2.2. Ermittlung der Alternative Datenbanken

Wenn eine alternative Datenbank-Lösung gesucht wird, sollte berücksichtigt werden, dass diese in AWS eingesetzt werden muss. AWS (Amazon Web Services) ist eine der bekanntesten Cloud-Computing-Plattformen, die von Amazon bereitgestellt wird. AWS bietet eine breite Palette an Diensten, darunter Compute, Speicher, Datenbanken, Netzwerk, Analyse, künstliche Intelligenz, Sicherheit, Mobile und IoT. Die Plattform ermöglicht es Unternehmen, ihre IT-Infrastruktur in der Cloud zu betreiben und bietet flexible, skalierbare und kosteneffektive Lösungen. AWS hat eine hohe Verfügbarkeit und bietet eine globale Infrastruktur mit Rechenzentren in verschiedenen Regionen weltweit. Darüber hinaus bietet AWS eine breite





Palette an Datenbankdiensten, darunter relationale Datenbanken, NoSQL-Datenbanken, In-Memory-Datenbanken und Graphdatenbanken. Es ist wichtig, bei der Auswahl der alternativen Datenbank-Lösung die Anforderungen der Anwendung und die verfügbaren Optionen in AWS sorgfältig zu prüfen. Im Folgenden werden alle Datenbanken beschrieben, die von AWS angeboten werden.

2.2.1. Amazon Aurora

Amazon Aurora ist eine relationale Datenbank, die MySQL- und PostgreSQL-kompatible Editionen unterstützt und eine hohe Verfügbarkeit, Skalierbarkeit und Leistung bietet.[47] Aurora ist eine schnelle, zuverlässige und skalierbare relationale Datenbank, die speziell für Cloud-Anwendungen entwickelt wurde. Aurora ist auch eine verwaltete Datenbanklösung, was bedeutet, dass AWS die Wartung, Sicherung und Skalierung der Datenbank übernimmt. Das bedeutet, dass Entwickler und Unternehmen sich auf die Entwicklung ihrer Anwendungen konzentrieren können, anstatt sich um die Wartung der Datenbank kümmern zu müssen [[51] 14].

2.2.2. Amazon RDS

Amazon RDS ist eine verwaltete relationale Datenbank, die eine Vielzahl von SQL-Datenbank-Engines unterstützt, einschließlich MySQL, PostgreSQL, Oracle und SQL Server. RDS bietet eine automatische Replikation, Backups und Skalierbarkeit, um die Anforderungen von Anwendungen zu erfüllen, die auf einer relationalen Datenbank basieren. AWS übernimmt die Wartung, Sicherung und Skalierung der Datenbank, so





dass Entwickler und Unternehmen sich auf die Entwicklung ihrer Anwendungen konzentrieren können [15].

2.2.3. Amazon DynamoDB

Amazon DynamoDB ist eine verwaltete NoSQL-Datenbank, die eine schnelle und skalierbare Datenverarbeitung ermöglicht. DynamoDB ist eine flexible, vollständig verwaltete NoSQL-Datenbank, die auf die Verarbeitung von großen Mengen strukturierter und unstrukturierter Daten ausgelegt ist. DynamoDB ermöglicht Entwicklern, Daten einfach und schnell zu speichern und abzurufen, ohne sich um Skalierbarkeit, Leistung oder Verfügbarkeit kümmern zu müssen [16].

2.2.4. Amazon DocumentDB

Amazon DocumentDB ist eine verwaltete Dokumentdatenbank, die MongoDB-kompatibel ist und eine hohe Verfügbarkeit, Skalierbarkeit und Leistung bietet. DocumentDB ist eine schnelle, skalierbare und zuverlässige

Dokumentdatenbank, die speziell für die Verarbeitung von semi-strukturierten und unstrukturierten Daten in der Cloud entwickelt wurde. AWS übernimmt die Wartung, Sicherung und Skalierung der Datenbank, so dass Entwickler und Unternehmen sich auf die Entwicklung ihrer Anwendungen konzentrieren können [17].

2.2.5. Amazon Neptune

Amazon Neptune ist eine verwaltete Graphendatenbank, die auf die Speicherung und Abfrage von stark miteinander verbundenen Daten spezialisiert ist. Neptune ermöglicht es Entwicklern, Beziehungen zwischen

23

==========24/78==========

Datenpunkten zu modellieren, zu speichern und abzufragen, was ideal für Anwendungen wie soziale Netzwerke, Wissensmanagement-Systeme und Empfehlungssysteme ist. AWS übernimmt die Wartung, Sicherung und Skalierung der Datenbank, so dass Entwickler und Unternehmen sich auf die Entwicklung ihrer Anwendungen konzentrieren können [18].

2.2.6. Amazon Keyspaces

Amazon Keyspaces ist eine verwaltete Cassandra-kompatible NoSQL-Datenbank, die eine schnelle und skalierbare Datenverarbeitung ermöglicht. Keyspaces ist eine vollständig verwaltete NoSQL-Datenbank, die auf die Verarbeitung von großen Mengen strukturierter und unstrukturierter Daten ausgelegt ist. Es bietet eine hohe Verfügbarkeit, Skalierbarkeit und Leistung und unterstützt Apache Cassandra-APIs. Entwickler und Unternehmen können Daten einfach und schnell speichern und abrufen, ohne sich um Skalierbarkeit, Leistung oder Verfügbarkeit kümmern zu müssen [19].

2.2.7. Amazon Timestream

Amazon Timestream ist eine verwaltete Zeitreihendatenbank, die auf die Speicherung und Abfrage von großen Mengen an Zeitreihendaten spezialisiert ist. Timestream ist eine schnelle, skalierbare und zuverlässige Datenbank, die speziell für die Verarbeitung von Zeitreihendaten in der Cloud entwickelt wurde. Es unterstützt eine Vielzahl von Anwendungsfällen wie IoT, Überwachung, Protokollierung und Betriebsanalyse. AWS übernimmt die Wartung, Sicherung und Skalierung der Datenbank, so dass Entwickler und Unternehmen sich auf die Entwicklung ihrer Anwendungen konzentrieren können [20].

24

==========25/78==========

2.2.8. Amazon Quantum Ledger Database (QLDB)

Amazon Quantum Ledger Database (QLDB) ist eine verwaltete Transaktionsdatenbank, die eine unveränderliche, fälschungssichere und hochskalierbare Speicherung von Anwendungsdaten ermöglicht. [28] [37] [31] [38] ... QLDB ist eine schnelle, skalierbare und zuverlässige Datenbank, die speziell für Anwendungsfälle entwickelt wurde, bei denen die Unveränderlichkeit und Integrität von Daten von entscheidender Bedeutung sind. Es bietet eine hohe Verfügbarkeit und Skalierbarkeit und ist ideal für Anwendungsfälle wie Supply-Chain-Management, Identitätsmanagement und Finanztransaktionen. AWS übernimmt die Wartung, Sicherung und Skalierung der Datenbank, so dass Entwickler und Unternehmen sich auf die Entwicklung ihrer Anwendungen konzentrieren können [21].

2.2.9. Amazon RedShift

Amazon RedShift ist eine schnelle, skalierbare und verwaltete Data-Warehouse-Lösung, die auf die Verarbeitung großer Datenmengen ausgelegt ist. RedShift ermöglicht es Entwicklern und Unternehmen, große Datenmengen zu speichern und abzufragen, um fundierte Entscheidungen zu treffen. Es bietet eine hohe Verfügbarkeit, Skalierbarkeit und Leistung und unterstützt SQL-basierte Abfragen. AWS übernimmt die Wartung, Sicherung und Skalierung der Datenbank, so dass Entwickler und Unternehmen sich auf die Entwicklung ihrer Anwendungen konzentrieren können [22].

25

==========26/78==========

2.2.10. Amazon ElastiCache

Amazon ElastiCache ist eine verwaltete In-Memory-Caching-Lösung, die die Leistung von Anwendungen verbessert, indem sie häufig verwendete Daten in einem schnellen und skalierbaren Cache speichert. ElastiCache unterstützt eine Vielzahl von Caching-Engines wie Redis und Memcached und ist ideal für Anwendungsfälle wie schnelle Anwendungsleistung und Skalierung. AWS übernimmt die Wartung, Sicherung und Skalierung des Caches, so dass Entwickler und Unternehmen sich auf die Entwicklung ihrer Anwendungen konzentrieren können [23].

#### 2.2.11. Amazon MemoryDB

Amazon MemoryDB ist eine verwaltete In-Memory-Datenbank, die auf die Verarbeitung von Daten in Echtzeit ausgelegt ist. MemoryDB ist eine schnelle, skalierbare und zuverlässige Datenbank, die speziell für Anwendungsfälle entwickelt wurde, bei denen die Latenz und Verfügbarkeit von Daten von entscheidender Bedeutung sind. Es unterstützt eine Vielzahl von APIs und ist ideal für Anwendungsfälle wie Echtzeit-Analyse und Betrugsbekämpfung. AWS übernimmt die Wartung, Sicherung und Skalierung der Datenbank, so dass Entwickler und Unternehmen sich auf die Entwicklung ihrer Anwendungen konzentrieren können [24].

#### 2.2.12. Redis

Redis ist eine Open-Source-Datenbank, die auf der In-Memory-Technologie basiert und als Key-Value-Store fungiert. Es handelt sich um eine NoSQL-Datenbank, die dazu verwendet werden kann, sehr große Datenmengen schnell und effizient zu speichern und zu verarbeiten. Redis wurde als



==========27/78==========

Cache-System entwickelt und kann als solches verwendet werden, um die Leistung von Anwendungen, die auf Datenbanken zugreifen, zu verbessern. Redis bietet jedoch auch eine Vielzahl von anderen Funktionen, wie z.B. Listen, Sätze, Hashes, und geordnete Sätze[25].

Redis ist sehr schnell, da alle Daten im Hauptspeicher gehalten werden und nicht auf eine Festplatte geschrieben werden müssen, wie es bei anderen Datenbanken der Fall ist. Dies ermöglicht es Redis, viele Anfragen pro Sekunde zu verarbeiten und sehr schnelle Antwortzeiten zu liefern. Redis bietet auch eine hohe Verfügbarkeit und Skalierbarkeit, da es eine Master-Slave-Replikation unterstützt, mit der mehrere Instanzen von Redis miteinander synchronisiert werden können. Redis kann auch horizontal skaliert werden, indem es auf mehrere Server aufgeteilt wird [26].

Redis unterstützt auch Transaktionen und bietet mehrere Mechanismen zur Gewährleistung von Konsistenz und Sicherheit. Es bietet auch eine eingebaute Unterstützung für geografische Informationssysteme (GIS) und Volltextsuche. Redis ist sehr flexibel und kann in einer Vielzahl von Anwendungsfällen eingesetzt werden, einschließlich Echtzeitnachrichten-systemen, sozialen Netzwerken, IoT-Anwendungen, E-Commerce-Plattformen und vielem mehr [27], [28].

In diesem Abschnitt wurden alle Datenbanken beschrieben, die von AWS angeboten werden, sowie deren wichtigste Merkmale und Anwendungsfälle. Wie man sehen kann, bietet AWS eine breite Palette von Datenbanklösungen an, die auf die Bedürfnisse von Entwicklern und



==========28/78==========

Unternehmen zugeschnitten sind. Von relationale Datenbanken wie Amazon Aurora und Amazon RDS bis hin zu NoSQL-Datenbanken wie Amazon DynamoDB und Amazon Neptune, AWS hat eine Lösung für fast jeden Anwendungsfall. Es ist wichtig zu beachten, dass AWS auch verwaltete Lösungen anbietet, was bedeutet, dass AWS die Wartung, Sicherung und Skalierung der Datenbanken übernimmt, so dass Entwickler und Unternehmen sich auf die Entwicklung ihrer Anwendungen konzentrieren können.[36] [5]

### 2.3. Ermittlung der Evaluation Kriterien

Die Ermittlung der Evaluationskriterien ist ein wichtiger Schritt bei der Bewertung von Datenbanklösungen. Diese Kriterien dienen als Messlatte, um verschiedene Datenbanken objektiv zu vergleichen und letztendlich die

beste Lösung für eine Anwendung auszuwählen.
Die Evaluierung von Datenbanklösungen erfordert die Berücksichtigung mehrerer Faktoren, um eine fundierte Entscheidung zu treffen. Performance, Skalierbarkeit, Verfügbarkeit, Sicherheit, Flexibilität, Kosten und Community-Unterstützung sind einige der wichtigsten Evaluationskriterien. Es ist wichtig, jedes Kriterium sorgfältig zu bewerten, um die beste Datenbanklösung für die Anwendung zu finden. Eine Datenbanklösung mit einer hohen Performance, Skalierbarkeit, Verfügbarkeit und Sicherheit, die flexibel und kosteneffektiv ist und von einer aktiven Community unterstützt wird, ist in der Regel die beste Wahl für eine Anwendung.
28
==========29/78==========
Im Folgenden sind einige der wichtigsten Evaluationskriterien aufgeführt, die bei der Bewertung von Datenbanken berücksichtigt werden sollten:
2.3.1. Performance
Die Performance ist ein wichtiger Faktor bei der Auswahl einer Datenbanklösung.[27] [8] [11] ... Die Performance hängt von vielen Faktoren ab, einschließlich der Hardware, der Netzwerklatenz und der Anzahl der Benutzer. Die Evaluierung der Performance sollte sowohl unter normalen als auch unter Spitzenlastbedingungen durchgeführt werden. Dazu können Benchmarks und Stress-Tests durchgeführt werden. Eine schlechte Performance kann zu langen Antwortzeiten und einer schlechten Benutzererfahrung führen. Daher ist es wichtig, eine Datenbank zu wählen, die eine hohe Performance bei der Verarbeitung großer Datenmengen bietet.
2.3.2. Skalierbarkeit
Die Skalierbarkeit ist ein weiterer wichtiger Faktor bei der Auswahl einer Datenbanklösung. Die Skalierbarkeit bezieht sich darauf, wie gut die Datenbank mit wachsenden Datenmengen und Anforderungen umgehen kann. Die Evaluierung der Skalierbarkeit sollte sowohl horizontal als auch vertikal erfolgen. Eine horizontale Skalierbarkeit ermöglicht es, mehrere Server hinzuzufügen, um die Last auf mehrere Maschinen zu verteilen, während eine vertikale Skalierbarkeit die Kapazität der bestehenden Hardware erhöht. Eine skalierbare Datenbanklösung ermöglicht es, mit wachsenden Anforderungen Schritt zu halten, ohne dass teure Hardware-Upgrades erforderlich sind.
29
==========30/78==========
2.3.3. Verfügbarkeit
Die Verfügbarkeit bezieht sich darauf, wie gut die Datenbank im Betrieb ist und wie schnell sie auf Ausfälle reagieren kann. Eine Datenbank mit einer hohen Verfügbarkeit stellt sicher, dass die Anwendung auch bei Ausfällen der Datenbank verfügbar bleibt. Die Evaluierung der Verfügbarkeit sollte sicherstellen, dass die Datenbanklösung über eine hohe Verfügbarkeit, Failover-Mechanismen und Notfallwiederherstellungsfunktionen verfügt.
2.3.4. Sicherheit
Die Evaluierung der Sicherheit sollte sicherstellen, dass die Datenbanklösung über robuste Sicherheitsfunktionen verfügt, um die Daten vor unbefugtem Zugriff und Verlust zu schützen. Die Datenbank sollte über Verschlüsselungsmechanismen und Zugriffskontrollen verfügen, um die Integrität der Daten zu gewährleisten. Die Datenbank sollte auch in der Lage sein, mit geltenden Datenschutzvorschriften und -richtlinien wie der DSGVO oder dem HIPAA-Standard konform zu sein. [0]
2.3.5. Flexibilität
Die Flexibilität bezieht sich auf die Fähigkeit der Datenbanklösung, verschiedene Arten von Datenstrukturen und -formaten zu unterstützen. Eine flexible Datenbanklösung ermöglicht es den Entwicklern, Daten in verschiedenen Formaten wie JSON, XML oder CSV zu speichern und abzurufen. Die Evaluierung der Flexibilität sollte sicherstellen, dass die Datenbanklösung die Flexibilität bietet, die Anforderungen der Anwendung zu erfüllen. Eine flexible Datenbank ermöglicht es Entwicklern, schnell auf
30
==========31/78==========

sich ändernde Anforderungen zu reagieren, ohne die Integrität der Daten zu gefährden.

2.3.6. Kosten

Die Kosten sind ein wichtiger Faktor bei der Auswahl einer Datenbanklösung.[8] [11] ... Die Evaluierung der Kosten sollte sicherstellen, dass die Datenbanklösung den Anforderungen der Anwendung entspricht und gleichzeitig innerhalb des Budgets des Unternehmens bleibt. Die Kosten können von verschiedenen Faktoren wie der Anzahl der Benutzer, der Datenmenge und der Art der verwendeten Datenbank abhängen. Es ist wichtig, die Kosten im Voraus abzuschätzen und sicherzustellen, dass die Datenbanklösung den Anforderungen der Anwendung entspricht, ohne das Budget zu sprengen.

Ein Kostenvergleich zwischen MongoDB und anderen Datenbanken wird auf der Grundlage von zwei Clustern durchgeführt werden, da dies ein realistischeres Bild der Kosten für den Betrieb der Datenbanken in einem produktiven Umfeld liefert. Es ist wichtig, die direkten und indirekten Kosten für die Einrichtung, den Betrieb, die Wartung und die Skalierung der Cluster zu berücksichtigen. Zu den direkten Kosten gehören Hardware- und Lizenzkosten, während Wartungs-, Monitoring-, Skalierungs- und Integrationskosten als indirekte Kosten betrachtet werden können. Die Hardwarekosten können sich aus den Kosten für Server und Netzwerkinfrastruktur zusammensetzen. Je nach Anzahl der Server und der benötigten Hardwarefunktionen können diese Kosten erheblich variieren. Lizenzkosten können ebenfalls beträchtlich sein, je nach der Art und Anzahl





der benötigten Lizenzen. Die Wartungskosten können sich aus den Kosten für die Überwachung und das Management der Datenbanken ergeben. Dazu können auch die Kosten für Support und Upgrades gehören. Monitoring-Kosten können durch den Einsatz von Monitoring-Tools oder den Einsatz von IT-Fachleuten, um die Leistung der Datenbanken zu überwachen, entstehen.[1]

Skalierungskosten können entstehen, wenn die Cluster aufgrund von Datenwachstum oder einer höheren Anzahl von Benutzern erweitert werden müssen. Die Kosten können je nach Datenbank und Art der Skalierung variieren. Die Integrationskosten können sich aus den Kosten für die Integration der Datenbanken in andere Systeme ergeben. Dies kann Zeit und Ressourcen erfordern, um sicherzustellen, dass die Datenbanken korrekt mit anderen Systemen wie Anwendungen, APIs und Cloud-Plattformen interagieren.

3. VERGLEICH DER MANGODB UND ANDEREN ALTERNATIVEN DATENBANKEN

Das Team hat sich mit dem Head Technical Designer und einem Entwickler zusammengesetzt, um die für ihre Anforderungen relevanten Datenbanken zu identifizieren. Infolgedessen wurden nur die für das Team relevanten Datenbanken ausgewählt und verglichen.

Aufgrund schlechter Erfahrungen in der Vergangenheit und einer schlechten Integration mit Spring fällt DynamoDB für das Team weg. Außerdem ist DocumentDB aufgrund seiner Ähnlichkeit mit MongoDB nicht relevant.





Neptune, Keyspaces und Timestream sind für die Anforderungen des Teams ebenfalls nicht relevant.

QLDB wurde ebenfalls ausgeschlossen, da das Team die Daten leicht selbst wiederherstellen kann und QLDB eher für Finanzanwendungen relevant ist.

RedShift bietet eine Data-Warehouse-Lösung an und ist für den Anwendungsfall des Teams nicht relevant.

Aurora und RDS sind fast gleich gebaut und wird daher nur Aurora evaluiert.

Es bleiben Aurora, Amazon ElastiCache und MemoryDB als mögliche Optionen für das Team übrig.

3.1. Performance

In diesem Vergleich werden Aurora, Amazon ElastiCache, MemoryDB und MongoDB hinsichtlich ihrer Leistung bewertet, um die Stärken und Schwächen jeder Datenbank zu identifizieren und zu verstehen, wie sie unter hoher Last funktionieren. Die Performance einer Datenbank hängt von verschiedenen Faktoren ab, einschließlich der Datenmenge, der Anzahl der Anfragen und der Hardwarekonfiguration. In diesem Vergleich wird ein Überblick über die Leistung jeder Datenbank gegeben und ihre Vor- und Nachteile diskutiert.
Aurora-Performance:
Aurora bietet eine schnelle und skalierbare Datenbanklösung mit einer hohen Verfügbarkeit und automatischer Datenreplikation. Die Leistung von
33
==========34/78==========
Aurora hängt von verschiedenen Faktoren ab, einschließlich der Hardwarekonfiguration, der Anzahl der Knoten im Cluster und der Art der Anfragen.
In Bezug auf die Lesegeschwindigkeit bietet Aurora eine hohe Performance, insbesondere bei Abfragen, die Indizes verwenden. Aurora reagiert schnell auf Anfragen mit geringer Latenz und hoher Durchsatzrate. Durch den Einsatz von SSD-Speicher und der richtigen Anzahl von Knoten im Cluster kann die Performance von Aurora optimiert werden.
Aurora bietet eine native Unterstützung für JSON, was es zu einer guten Wahl für Anwendungen macht, die JSON-Daten verarbeiten. Zudem unterstützt Aurora ACID-Transaktionen (Atomicity, Consistency, Isolation, Durability), was es zu einer guten Wahl für Anwendungen macht, die eine relationale Datenbank mit ACID-Transaktionen benötigen.
Amazon ElastiCache-Performance:
Amazon ElastiCache unterstützt sowohl Redis als auch Memcached und ermöglicht die schnelle Speicherung und Abfrage von Daten. [57] ElastiCache kann auch als Caching-Layer zwischen Anwendungen und Datenbanken verwendet werden, um die Leistung und Skalierbarkeit zu verbessern.
Die Performance von ElastiCache hängt von verschiedenen Faktoren ab, einschließlich der Hardwarekonfiguration, der Anzahl der Cache-Instanzen im Cluster und der Art der Anfragen. ElastiCache bietet eine sehr schnelle
34
==========35/78==========
Schreibgeschwindigkeit und eine sehr hohe Lesegeschwindigkeit, insbesondere bei Abfragen, die den Cache nutzen.
Ein weiteres Merkmal von ElastiCache ist, dass er nativ viele Caching-Patterns unterstützt, einschließlich des LRU-Algorithmus (Least Recently Used) und des TTL-Algorithmus (Time to Live). Es bietet auch die Möglichkeit, die Daten im Cache zu verschlüsseln, um die Sicherheit zu erhöhen.
MemoryDB-Performance:
Die Performance von MemoryDB hängt ebenso von verschiedenen Faktoren ab, einschließlich der Hardwarekonfiguration, der Anzahl der Cache-Instanzen im Cluster und der Art der Anfragen. MemoryDB bietet eine sehr hohe Schreibgeschwindigkeit und eine sehr hohe Lesegeschwindigkeit, insbesondere bei Abfragen, die den Cache nutzen. Darüber hinaus bietet MemoryDB eine natürliche Integration mit anderen AWS-Services, wie z.B. Lambda-Funktionen, um eine Echtzeit-Verarbeitung von Daten zu ermöglichen. MemoryDB unterstützt auch die Skalierung von Daten auf mehrere Knoten im Cluster, um die Leistung und Skalierbarkeit zu erhöhen.
MongoDB-Performance:
MongoDB bietet eine hohe Performance bei der Speicherung und Abfrage von unstrukturierten oder semi-strukturierten Daten. MongoDB ermöglicht auch die horizontale Skalierung auf mehrere Knoten im Cluster, um die Leistung und Skalierbarkeit zu erhöhen.
35
==========36/78==========
In Bezug auf die Lesegeschwindigkeit bietet MongoDB eine hohe Performance, insbesondere bei Abfragen, die Indizes verwenden. MongoDB

bietet auch eine native Unterstützung für JSON, was es zu einer guten Wahl für Anwendungen macht, die JSON-Daten verarbeiten.
MongoDB bietet eine hohe Flexibilität bei der Datenspeicherung, da es keine festen Schemata gibt und Dokumente dynamisch hinzugefügt oder geändert werden können. Dies macht es zu einer guten Wahl für Anwendungen, die häufig Änderungen an ihren Datenmodellen vornehmen müssen. Ein weiteres Merkmal von MongoDB ist, dass es eine sehr umfassende Abfrage-Sprache und -Funktionen bietet, einschließlich der Unterstützung für komplexe Abfragen und Aggregationen. MongoDB unterstützt auch ACID-Transaktionen und bietet eine hohe Verfügbarkeit und automatische Replikation.
Im Vergleich zu anderen Datenbanken kann MongoDB jedoch bei sehr großen Datenmengen oder komplexen Abfragen an seine Grenzen stoßen. Auch die Skalierung von MongoDB kann aufgrund seiner flexiblen Datenstruktur und des Fehlens von festen Schemata komplexer sein als bei anderen Datenbanken.
Letztendlich hängt die Wahl der Datenbank von den spezifischen Anforderungen der Anwendung ab. Wenn Flexibilität und Skalierbarkeit wichtige Faktoren sind, kann MongoDB eine gute Wahl sein. Wenn jedoch eine hohe Lesegeschwindigkeit oder ACID-Transaktionen erforderlich sind, können Aurora oder andere relationale Datenbanken besser geeignet sein.
36
==========37/78==========
Wenn schnelle Caching- und Datenverarbeitungsfunktionen benötigt werden, können Amazon ElastiCache oder MemoryDB eine gute Wahl sein.
AHP-Vergleich
Zunächst wurden Bewertungskriterien für den AHP-Vergleich der Performance von MongoDB mit anderen Datenbanken wie Aurora, Amazon ElastiCache und MemoryDB festgelegt. Die Kriterien waren "Datenverarbeitungsgeschwindigkeit", "Verfügbarkeit", "Skalierbarkeit" und "Kosten". Anschließend wurden jedem Kriterium relative Bedeutungen zugewiesen. Dazu wurden Paarvergleiche mit einer Skala von 1 bis 9 durchgeführt, um die Bedeutungen zu quantifizieren.[30] [2] [48]
Die Paarvergleiche wurden in einer Vergleichsmatrix erfasst und für jedes Paar von Kriterien oder Alternativen durchgeführt.[2] Ein Paarvergleich (auch als Zweiervergleich oder binärer Vergleich bezeichnet) ist ein Verfahren zur quantitativen Bewertung von Alternativen hinsichtlich eines bestimmten Kriteriums. Hierbei werden zwei Alternativen miteinander verglichen, indem man angibt, welche der beiden Alternativen in Bezug auf das Kriterium bevorzugt wird. Dabei wird oft eine Skala von 1 bis 9 verwendet, wobei 1 bedeutet, dass die eine Alternative deutlich schlechter ist als die andere und 9 bedeutet, dass die eine Alternative deutlich besser ist als die andere. Die Ergebnisse der Paarvergleiche werden dann verwendet, um die relativen Bedeutungen der verschiedenen Kriterien und Alternativen zu bestimmen. Der Bewertende muss hierbei eine Entscheidung treffen, welches der beiden Kriterien oder Alternativen wichtiger ist oder eine höhere Priorität hat. Dies geschieht durch eine subjektive Einschätzung, die
37
==========38/78==========
auf Erfahrung und Expertenwissen basieren kann. Tabelle 3.1. zeigt ein Beispiel für die Vergleichsmatrix:
Kriterium Datenverarbeitungsges
chwindigkeit
Verfügba
rkeit
Skalierb
arkeit
Kost
en
Datenverarbeitungsges
chwindigkeit
1 5 3 7

Verfügbarkeit 1/5 1 3 5
Skalierbarkeit 1/3 1/3 1 3
Kosten 1/7 1/5 1/3 1
Tabelle 3.1. AHP-Vergleich - Bewertungskriterien und Paarvergleiche
Die Werte in der Vergleichsmatrix geben an, wie viel wichtiger das eine Kriterium im Vergleich zum anderen ist. Zum Beispiel ist "Datenverarbeitungsgeschwindigkeit" für den Bewertenden fünfmal wichtiger als "Verfügbarkeit".
Dann wurden die Datenbanktypen mit jedem Kriterium verglichen und eine Bewertung auf der Skala von 1 bis 9 abgegeben.[1] Die Bewertungen wurden in der Tabelle 3.2. wie folgt erfasst:
Kriterium Datenverarbeitungsgeschwindigkeit Verfügbarkeit Skalierbarkeit Kosten
MongoDB 8 7 9 6
Aurora 6 9 8 7
Amazon ElastiCache 7 8 7 8
MemoryDB 9 6 9 5
Tabelle 3.[1] 2. Bewertungstabelle für Performance-Vergleich zwischen MongoDB, Aurora, Amazon ElastiCache und MemoryDB anhand von Kriterien


Die Bewertungen wurden von dem Bewertenden festgelegt und spiegeln seine Ansichten wider. Zum Beispiel hat der Bewertende MongoDB eine Bewertung von 8 für "Datenverarbeitungsgeschwindigkeit" gegeben, während Aurora eine Bewertung von 6 für das gleiche Kriterium erhalten hat.
Anschließend wurde der Durchschnitt jeder Spalte in der Bewertungstabelle berechnet. Jede Zelle wurde dann durch den Durchschnitt ihrer Spalte dividiert. Hier ist die neue Tabelle:
Kriterium Datenverarbeitungsgeschwindigkeit Verfügbarkeit Skalierbarkeit Kosten Durchschnitt
MongoDB 8 7 9 6 7.5
Aurora 6 9 8 7 7.5
Amazon ElastiCache 7 8 7 8 7.5
MemoryDB 9 6 9 5 7.[1] 25
Summe 30 30 33 26

Tabelle 3.3. Bewertungstabelle für die Datenbanktypen in Bezug auf verschiedene Kriterien mit Durchschnittswerten
Die Durchschnittswerte in der letzten Spalte repräsentieren die relativen Bedeutungen der Datenbanktypen für jedes Kriterium. Diese relativen Bedeutungen wurden durch die Division jeder Zelle durch den Durchschnitt ihrer Spalte berechnet.
39
==========40/78==========
Als nächstes wurden die Summen jeder Zeile in der Tabelle berechnet, um den Beitrag jedes Datenbanktyps zu den Gesamtbewertungen zu bestimmen. Die Ergebnisse sind in der Tabelle 3.[6] [34] 4. erfasst:
Kriteriu
m
Datenverarbeitungsgeschwind
igkeit
Verfügbark
eit
Skalierbark
eit
Koste
n
MongoD
B
0.53 0.39 0.56 0.23
Aurora 0.40 0.52 0.48 0.27
Amazon
ElastiCac
he
0.47 0.45 0.42 0.31
Memory
DB
0.60 0.33 0.56 0.19
Tabelle 3.4. Gewichtete Beiträge für jede Datenbanktypen für jedes Kriterium
Die Werte in dieser Tabelle geben an, welchen Beitrag jeder Datenbanktyp zu den Gesamtbewertungen leistet. Sie wurden berechnet, indem jede Zelle durch die Summe ihrer Zeile geteilt wurde. Schließlich wurden die gewichteten Beiträge jeder Datenbanktypen addiert, um eine Gesamtbewertung zu erhalten. Die Gesamtbewertung für jede Datenbanktypen wurde wie folgt berechnet:
□MongoDB: 0.45
□Aurora: 0.42
40
==========41/78==========
□Amazon ElastiCache: 0.41
□MemoryDB: 0.43
Die Gesamtbewertungen wurden berechnet, indem die gewichteten Beiträge jeder Datenbanktypen addiert wurden. Basierend auf diesen Ergebnissen hat MongoDB die höchste Gesamtbewertung und ist somit die beste Wahl in Bezug auf die Performance.
3.2. Skalierbarkeit
In diesem Abschnitt werden Aurora, Amazon ElastiCache, MemoryDB und MongoDB hinsichtlich ihrer Skalierbarkeit verglichen (d.h. wie sie mit steigenden Anforderungen an die Datenverarbeitung umgehen können). Skalierbarkeit spielt eine wichtige Rolle bei der Auswahl einer Datenbank, da es entscheidend ist, dass die Datenbank den Anforderungen des wachsenden Datenverkehrs und der steigenden Nutzerzahl gerecht werden kann.[8] [11] ...
Aurora-Skalierbarkeit:
Aurora bietet automatische Skalierung und kann in Echtzeit erweitert werden, um die Verarbeitung von Daten zu verbessern. Aurora verwendet ein Cluster-Modell, das es Benutzern ermöglicht, mehrere Knoten zu

verwenden, um Daten zu speichern und zu verarbeiten. Aurora hat den Vorteil, dass es eine schnelle Skalierbarkeit bietet und automatisch erweitert werden kann, um den Anforderungen an die Datenverarbeitung gerecht zu werden.[0] [5] Es unterstützt auch das Hinzufügen von Lese-Replikationen, um die
41
==========42/78==========
Verarbeitung von Anfragen zu erhöhen und die Last auf dem System zu verteilen.
Amazon ElastiCache-Skalierbarkeit:
Amazon ElastiCache unterstützt die horizontale Skalierung, um die Kapazität zu erhöhen und die Verarbeitung von Daten zu beschleunigen. Es unterstützt auch das Hinzufügen von Knoten, um die Kapazität zu erhöhen und die Verarbeitung von Daten zu verbessern. ElastiCache verwendet ein Cluster-Modell, das es Benutzern ermöglicht, mehrere Knoten zu verwenden, um Daten zu speichern und zu verarbeiten. Eine der Stärken von ElastiCache ist, dass es eine schnelle Skalierbarkeit bietet und die horizontale Skalierung unterstützt, um den Anforderungen an die Datenverarbeitung gerecht zu werden.
MemoryDB-Skalierbarkeit:
MemoryDB unterstützt auch die horizontale Skalierung, um die Kapazität zu erhöhen und die Verarbeitung von Daten zu beschleunigen. Es unterstützt auch das Hinzufügen von Knoten, um die Kapazität zu erhöhen und die Verarbeitung von Daten zu verbessern. Ebenso verwendet MemoryDB ein Cluster-Modell, das es Benutzern ermöglicht, mehrere Knoten zu verwenden, um Daten zu speichern und zu verarbeiten.
MongoDB-Skalierbarkeit:
MongoDB unterstützt ebenfalls die horizontale Skalierung und das Hinzufügen von Knoten, um die Kapazität zu erhöhen und die Verarbeitung von Daten zu verbessern. MongoDB verwendet ein Sharding-Modell, das es
42
==========43/78==========
Benutzern ermöglicht, Daten auf mehrere Knoten zu verteilen und so die Skalierbarkeit zu erhöhen. MongoDB bietet auch die Möglichkeit, Daten auf verschiedenen geografischen Standorten zu replizieren, um die Verfügbarkeit zu erhöhen.
Somit bieten Aurora, Amazon ElastiCache, MemoryDB und MongoDB alle schnelle Skalierbarkeit und unterstützen das Hinzufügen von Knoten, um die Kapazität zu erhöhen und die Verarbeitung von Daten zu verbessern. Aurora bietet das Hinzufügen von Lese-Replikationen, um die Verarbeitung von Anfragen zu erhöhen und die Last auf dem System zu verteilen, während ElastiCache und MemoryDB die horizontale Skalierung unterstützen, um die Kapazität zu erhöhen und die Verarbeitung von Daten zu beschleunigen. MongoDB bietet das Sharding-Modell, das es Benutzern ermöglicht, Daten auf mehrere Knoten zu verteilen und so die Skalierbarkeit zu erhöhen, sowie die Möglichkeit, Daten auf verschiedenen geografischen Standorten zu replizieren, um die Verfügbarkeit zu erhöhen.
Eine tabellarische Zusammenfassung der Skalierbarkeitseigenschaften der drei Datenbanken ist in Tabelle 3.13. gegeben:
Eigenschaften Aurora Amazon
ElastiCache
MemoryDB MongoDB
Datenbanktyp Relationale
Datenbank
In-Memory-
Datenbank
(Redis,
Memcached)
In-Memory-
Datenbank
(Redis)
Document-
based NoSQL

Datenbank
Skalierbarkeit Automatische Skalierung, Erweiterung in Echtzeit
Horizontale Skalierung durch Hinzufügen von Knoten
Horizontale Skalierung durch Hinzufügen von Knoten
Horizontale Skalierung durch Sharding
Cluster-Modell Verwendung Verwendung Verwendung Verwendung


von mehreren Knoten
von mehreren Knoten
von mehreren Knoten
von mehreren Knoten
Replikation Hinzufügen von Lese-Replikationen
Hinzufügen von Lese-Replikationen
Keine Angabe
Replikation auf verschiedenen geografischen Standorten möglich
Speicherkapazität
Skalierbar Begrenzter Speicherplatz
Begrenzter Speicherplatz
Skalierbar
Datenspeicherung
Sichere Datenspeicherung
Verlust von Daten bei Systemausfällen möglich
Verlust von Daten bei Systemausfällen möglich

Sichere
Datenspeicheru
ng
Tabelle 3.5. Vergleich der Datenbankarchitekturen basierend auf Datenbanktyp, Skalierbarkeit, Cluster-Modell, Replikation, Speicherkapazität und Datenspeicherung
Wie in der Tabelle gezeigt wird, bietet Aurora automatische Skalierung und Erweiterung in Echtzeit, während Amazon ElastiCache, MemoryDB und MongoDB die horizontale Skalierung durch das Hinzufügen von Knoten oder Sharding unterstützen. Alle vier Datenbanken verwenden ein Cluster-Modell, um Daten auf mehreren Knoten zu speichern und zu verarbeiten. Aurora und Amazon ElastiCache unterstützen das Hinzufügen von Lese-Replikationen, während bei MemoryDB keine Angabe dazu gemacht wird. MongoDB bietet die Möglichkeit, Daten auf verschiedenen geografischen Standorten zu replizieren, um die Verfügbarkeit zu erhöhen. In Bezug auf die Speicherkapazität ist Aurora skalierbar, während bei Amazon ElastiCache und MemoryDB der Speicherplatz begrenzt ist. MongoDB ist ebenfalls skalierbar. In Bezug auf die Datenspeicherung bieten alle vier Datenbanken eine sichere Datenspeicherung, jedoch kann bei Amazon


ElastiCache und MemoryDB bei Systemausfällen ein Verlust von Daten auftreten.
AHP-Vergleich:
Der AHP-Vergleich der Skalierbarkeit von MongoDB mit Aurora, Amazon ElastiCache und MemoryDB wurde wie folgt durchgeführt:
Schritt 1: Kriterien und Alternativen identifizieren
Die Kriterien wurden als Skalierbarkeit und Cluster-Modell identifiziert. Die Alternativen sind Aurora, Amazon ElastiCache, MemoryDB und MongoDB.
Schritt 2: Paarweise Vergleiche durchführen
Paarweise Vergleiche wurden mit einer Skala von 1 bis 9 durchgeführt, um die relative Bedeutung der Kriterien und Alternativen zu quantifizieren.[2] Die Ergebnisse der paarsweisen Vergleiche der Kriterien sind in Tabelle 3.6 dargestellt:
Kriterien Skalierbarkeit Cluster-Modell
Skalierbarkeit 1 3
Cluster-Modell 1/3 1
Tabelle 3.6. Paarweise Vergleiche der Kriterien hinsichtlich Skalierbarkeit und Cluster-Modell
Die Ergebnisse der paarweisen Vergleiche der Alternativen hinsichtlich Skalierbarkeit sind in Tabelle 3.7 dargestellt:
Alternativen Aurora Amazon
ElastiCache
MemoryDB MongoDB


Aurora 1 3 5 7
Amazon
ElastiCache
1/3 1 3 5
MemoryDB 1/5 1/3 1 3
MongoDB 1/7 1/5 1/3 1
Tabelle 3.7. Paarweiser Vergleich der Alternativen hinsichtlich Skalierbarkeit
Schritt 3: Berechnen der Kriteriengewichte
Die Kriteriengewichte wurden berechnet, indem die Summe der Zeilen jedes Kriteriums in Tabelle 3.6 berechnet und durch die Gesamtsumme aller Zellen dividiert wurde. Die Ergebnisse sind in Tabelle 3.[4] [34] [43] ...8 dargestellt:
Kriterien Gewicht
Skalierbarkeit 0.75
Cluster-Modell 0.25

Tabelle 3.8. Berechnung der Kriteriengewichte
Schritt 4: Berechnen der Alternativengewichte
Die Alternativengewichte wurden berechnet, indem die Summe jeder Spalte in Tabelle 3.7 berechnet und durch die Gesamtsumme aller Zellen dividiert wurde. Die Ergebnisse sind in Tabelle 3.9 dargestellt:
Alternativen Gewicht
Aurora 0.263
Amazon ElastiCache 0.212
MemoryDB 0.150
MongoDB 0.375
Tabelle 3.9. Berechnung der Alternativen-Gewichte für den AHP-Vergleich der Skalierbarkeit


Schritt 5: Berechnen der Gesamtperformance
Die Gesamtperformance jeder Alternative wurde berechnet, indem die Gewichte der Kriterien aus Schritt 3 mit den Gewichten der Alternativen aus Schritt 4 multipliziert und summiert wurden. Die Ergebnisse sind in Tabelle 3.10 dargestellt:
Alternativen Gesamtperformance
Aurora 0.312
Amazon ElastiCache 0.266
MemoryDB 0.163
MongoDB 0.259
Tabelle 3.10. Vergleich der Gesamtperformance der Alternativen in Bezug auf Skalierbarkeit und Cluster-Modell
Schritt 6: Interpretation der Ergebnisse
Wie aus Tabelle 3.10 hervorgeht, hat Aurora die höchste Gesamtperformance von 0,312, gefolgt von Amazon ElastiCache mit einer Gesamtperformance von 0,266. MemoryDB hat die niedrigste Gesamtperformance von 0,163, während MongoDB eine Gesamtperformance von 0,259 hat.
Dies bedeutet, dass Aurora und Amazon ElastiCache in Bezug auf Skalierbarkeit und Cluster-Modell besser abschneiden als MemoryDB und MongoDB. Allerdings ist es wichtig zu beachten, dass jede Datenbank unterschiedliche Stärken und Schwächen hat und dass die Anforderungen an die Datenverarbeitung in jeder Anwendung unterschiedlich sein können.[0]


Daher sollte eine gründliche Analyse durchgeführt werden, um die beste Datenbanklösung für die spezifischen Anforderungen zu wählen.[53] [0]
Insgesamt bietet Aurora eine automatische Skalierung und eine schnelle Erweiterung in Echtzeit, während Amazon ElastiCache und MemoryDB die horizontale Skalierung unterstützen, um die Verarbeitung von Daten zu beschleunigen. MongoDB bietet auch eine horizontale Skalierung, jedoch ist die Speicherkapazität begrenzt im Vergleich zu Aurora und ElastiCache.
3.3. Verfügbarkeit
Alle vier Datenbanken bieten verschiedene Funktionen an, um die Verfügbarkeit zu erhöhen. Hier sind einige der verfügbaren Funktionen:
Aurora:
□Multi-AZ-Bereitstellungen: Die Datenbank wird in mehreren Verfügbarkeitszonen bereitgestellt, um eine höhere Verfügbarkeit zu gewährleisten.
□Automatisches Failover: Bei einem Ausfall wird automatisch auf eine gespiegelte Datenbank in einer anderen Verfügbarkeitszone umgeschaltet.
Amazon ElastiCache:
□Replikation: Die Datenbank repliziert automatisch Daten in mehrere Knoten, um die Verfügbarkeit zu erhöhen.


□Automatische Knotenersetzung: Wenn ein Knoten ausfällt, wird

automatisch ein neuer Knoten erstellt, um die Verfügbarkeit zu erhöhen.
MemoryDB:
□Replikation: MemoryDB repliziert automatisch Daten in mehrere Knoten, um die Verfügbarkeit zu erhöhen.
□Automatische Knotenersetzung: Wenn ein Knoten ausfällt, wird automatisch ein neuer Knoten erstellt, um die Verfügbarkeit zu erhöhen.
MongoDB:
□Replica Sets: MongoDB verwendet Replica Sets, um automatisch Daten in mehrere Knoten zu replizieren und die Verfügbarkeit zu erhöhen.
□Sharding: MongoDB unterstützt Sharding, um Daten automatisch in mehrere Knoten aufzuteilen und die Verfügbarkeit zu erhöhen.
Es ist jedoch wichtig zu beachten, dass jede Datenbank unterschiedliche Stärken und Schwächen hat und dass die Verfügbarkeit auch von anderen Faktoren wie der Konfiguration, dem Betriebssystem und der Netzwerkinfrastruktur abhängt.[0] Es ist daher ratsam, eine gründliche Analyse durchzuführen, um die beste Datenbanklösung für die spezifischen Anforderungen zu wählen.
AHP Vergleich:
49
==========50/78==========
Schritt 1: Festlegen der Kriterien
Zunächst wurden die Kriterien festgelegt, die zur Bewertung der Verfügbarkeit der Datenbanken herangezogen werden sollten. Diese Kriterien waren:
□Ausfallsicherheit
□Wiederherstellbarkeit
□Skalierbarkeit
□Datensicherheit
Schritt 2: Gewichtung der Kriterien
Anschließend wurden den Kriterien relative Gewichte zugewiesen. Hierbei wurden Paarvergleiche mit einer Skala von 1 bis 9 durchgeführt, um die Bedeutungen zu quantifizieren.[4] Die Ergebnisse sind in den Tabellen 3.[4] [43] 11 und 3.12 dargestellt:
Kriterium Ausfallsic
herheit
Wiederherst
ellbarkeit
Skalierb
arkeit
Datensic
herheit
Ausfallsicher
heit
1 3 2 5
Wiederherste
llbarkeit
1/3 1 1/3 3
Skalierbarkei
t
1/2 3 1 3
Datensicherh
eit
1/5 1/3 1/3 1
Tabelle 3.11. Paarvergleiche für die Gewichtung der Kriterien:[6] [50]
Kriterium Gewicht
Ausfallsicherheit 0.35
Wiederherstellbarkeit 0.30
Skalierbarkeit 0.20

==========51/78==========
Datensicherheit 0.15
Tabelle 3.12. Gewichtung der Kriterien
Schritt 3: Gewichtung der Alternativen
Den Alternativen Datenbanken wurden relative Gewichte zugewiesen, indem Paarvergleiche mit einer Skala von 1 bis 9 durchgeführt wurden.[30] Die Ergebnisse sind in den Tabellen 3.13 und 3.14 dargestellt:
Alternative Auror
a
Amazon
ElastiCache
Memory
DB
MongoD
B
Aurora 1 3/5 3/5 3/5
Amazon
ElastiCache
5/3 1 3/5 3/5
MemoryDB 5/3 5/3 1 5/3
MongoDB 5/3 5/3 3/5 1
Tabelle 3.[1] 13. Paarvergleiche für die Gewichtung der Alternativen
Alternative Gewicht
Aurora 0.225
Amazon ElastiCache 0.225
MemoryDB 0.225
MongoDB 0.325
Tabelle 3.14: Gewichtung der Alternativen
Schritt 4: Bewertung der Alternativen hinsichtlich der Kriterien
Anschließend wurden die Alternativen hinsichtlich jedes Kriteriums bewertet. Dabei wurden Paarvergleiche mit einer Skala von 1 bis 9 durchgeführt, um die Leistung der Alternativen zu quantifizieren.[2] Die Ergebnisse sind in den folgenden Tabellen dargestellt:
51
==========52/78==========
Alternative Ausfallsicherheit
Aurora 0.20
Amazon ElastiCache 0.15
MemoryDB 0.10
MongoDB 0.55
Tabelle 3.15: Bewertung der Alternativen hinsichtlich der Ausfallsicherheit
Alternative Wiederherstellbarkeit
Aurora 0.25
Amazon ElastiCache 0.20
MemoryDB 0.15
MongoDB 0.55
Tabelle 3.16: Bewertung der Alternativen hinsichtlich der Skalierbarkeit
Alternative Skalierbarkeit
Aurora 0.45
Amazon ElastiCache 0.45
MemoryDB 0.45
MongoDB 0.55
Tabelle 3.17: Bewertung der Alternativen hinsichtlich der Wiederherstellbarkeit
Alternative Datensicherheit
Aurora 0.30
Amazon ElastiCache 0.25
MemoryDB 0.25
MongoDB 0.50
Tabelle 3.18: Bewertung der Alternativen hinsichtlich der Datensicherheit

Schritt 5: Berechnen der Gesamtverfügbarkeit

Die Gesamtverfügbarkeit jeder Alternative wurde berechnet, indem die Gewichte der Kriterien aus Schritt 2 mit den Gewichten der Alternativen aus Schritt 3 multipliziert und summiert wurden.
52
==========53/78==========
Beispiel: Angenommen, es liegen vier Alternativen (d.h., Aurora, Amazon ElastiCache, MemoryDB und MongoDB) und vier Kriterien (d.h. Ausfallsicherheit, Wiederherstellbarkeit, Skalierbarkeit und Datensicherheit) vor, für die Gewichtungen in Schritt 2 festgelegt wurden. Ausfallsicherheit wurde mit 0.35 gewichtet, Wiederherstellbarkeit mit 0.30, Skalierbarkeit mit 0.20 und Datensicherheit mit 0.15.
In Schritt 3 wurden die Gewichtungen der Alternativen durch Paarvergleiche wie folgt festgelegt: Aurora mit 0.225, Amazon ElastiCache mit 0.225, MemoryDB mit 0.225 und MongoDB mit 0.325.
Nun muss die Gesamtverfügbarkeit jeder Alternative berechnet werden, indem die Gewichtungen der Kriterien aus Schritt 2 mit den Gewichtungen der Alternativen aus Schritt 3 multipliziert und summiert werden. Ein Beispiel für die Berechnung der Gesamtverfügbarkeit von Aurora ist wie folgt:
□Die gewichtete Bewertung für Ausfallsicherheit ergibt sich aus 0.35 x 0.20 = 0.07.
□Die gewichtete Bewertung für Wiederherstellbarkeit ergibt sich aus 0.30 x 0.25 = 0.075.
□Die gewichtete Bewertung für Skalierbarkeit ergibt sich aus 0.20 x 0.45 = 0.09.
□Die gewichtete Bewertung für Datensicherheit ergibt sich aus 0.15 x 0.30 = 0.045.
53
==========54/78==========
Die Gesamtverfügbarkeit von Aurora ergibt sich aus der Summe dieser gewichteten Bewertungen: 0.07 + 0.075 + 0.09 + 0.045 = 0.28. Auf diese Weise kann die Gesamtverfügbarkeit jeder Alternative berechnet werden, indem die Gewichtungen der Kriterien mit den Gewichtungen der Alternativen multipliziert und summiert werden. Die vollständigen Ergebnisse sind in der folgenden Tabelle dargestellt:[61] [62] [4]
Alternative Gesamtverfügbarkeit
Aurora 0.28
Amazon ElastiCache 0.30375
MemoryDB 0.24125
MongoDB 0.33625
Tabelle 3.19: Gesamtverfügbarkeit der Alternativen
Schritt 6: Interpretation der Ergebnisse
Die Ergebnisse zeigen, dass MongoDB mit einer Gesamtverfügbarkeit von 0.33625 die höchste Bewertung erzielt hat, gefolgt von Amazon ElastiCache mit 0.30375, Aurora mit 0.28 und MemoryDB mit 0.24125. Eine detaillierte Analyse der Bewertungen zeigt, dass MongoDB die höchste Bewertung in den Kriterien Ausfallsicherheit, Wiederherstellbarkeit und Datensicherheit erzielt hat. Alle Alternativen schnitten jedoch ähnlich gut in Bezug auf die Skalierbarkeit ab. Basierend auf diesen Ergebnissen kann empfohlen werden, dass MongoDB als Alternative mit der höchsten Gesamtverfügbarkeit in Betracht gezogen wird, insbesondere wenn hohe Anforderungen an Ausfallsicherheit, Wiederherstellbarkeit und Datensicherheit gestellt werden.
54
==========55/78==========
3.4. Sicherheit
In diesem Abschnitt werden Aurora, Amazon ElastiCache, MemoryDB und MongoDB hinsichtlich der Sicherheit verglichen. Die Sicherheit ist ein wichtiger Faktor bei der Auswahl einer Datenbank und kann je nach den Anforderungen der Anwendung variieren.[27]
Aurora-Sicherheit:
Aurora bietet eine umfassende Sicherheitslösung, die Daten auf

verschiedenen Ebenen schützt. Aurora unterstützt die Verschlüsselung von Daten in Ruhe und in Bewegung und bietet eine native Unterstützung für SSL/TLS-Verschlüsselung. Aurora unterstützt auch die Verschlüsselung von Daten mit AWS Key Management Service (KMS), um die Sicherheit zu erhöhen. Eine weitere Funktion von Aurora ist die Möglichkeit, Netzwerkzugriffssteuerungen zu konfigurieren, um unerwünschte Zugriffe auf die Datenbank zu verhindern. Aurora unterstützt auch das Identity and Access Management (IAM) von AWS, um den Zugriff auf die Datenbank zu steuern.[9] Mit IAM können Benutzer und Gruppen erstellt werden, um den Zugriff auf Daten zu steuern.

Amazon ElastiCache-Sicherheit:

Amazon ElastiCache bietet eine robuste Sicherheitslösung, die die Sicherheit von In-Memory-Daten gewährleistet. ElastiCache unterstützt die Verschlüsselung von Daten in Ruhe und in Bewegung und bietet eine native Unterstützung für SSL/TLS-Verschlüsselung. ElastiCache unterstützt auch die Verschlüsselung von Daten mit AWS Key Management Service (KMS),





um die Sicherheit zu erhöhen. ElastiCache bietet auch die Möglichkeit, Netzwerkzugriffssteuerungen zu konfigurieren, um unerwünschte Zugriffe auf die Datenbank zu verhindern.[9] ElastiCache unterstützt auch das Identity and Access Management (IAM) von AWS, um den Zugriff auf die Datenbank zu steuern. Mit IAM können Benutzer und Gruppen erstellt werden, um den Zugriff auf Daten zu steuern.

MemoryDB-Sicherheit:

MemoryDB bietet eine umfassende Sicherheitslösung, die die Sicherheit von In-Memory-Daten gewährleistet. MemoryDB unterstützt die Verschlüsselung von Daten in Ruhe und in Bewegung und bietet eine native Unterstützung für SSL/TLS-Verschlüsselung. MemoryDB unterstützt auch die Verschlüsselung von Daten mit AWS Key Management Service (KMS), um die Sicherheit zu erhöhen.

MongoDB-Sicherheit:

MongoDB bietet eine umfassende Sicherheitslösung, die Daten auf verschiedenen Ebenen schützt. MongoDB unterstützt die Verschlüsselung von Daten in Ruhe und in Bewegung und bietet eine native Unterstützung für SSL/TLS-Verschlüsselung. MongoDB unterstützt auch die Verschlüsselung von Daten mit dem Key Management System (KMS) von AWS, um die Sicherheit zu erhöhen. Eine weitere Funktion von MongoDB ist die Möglichkeit, Netzwerkzugriffssteuerungen zu konfigurieren, um unerwünschte Zugriffe auf die Datenbank zu verhindern. MongoDB unterstützt auch das Role-Based Access Control (RBAC), um den Zugriff





auf die Datenbank zu steuern. Mit RBAC können Benutzer und Gruppen erstellt werden, um den Zugriff auf Daten zu steuern.

Somit bieten alle vier Datenbanken eine solide Sicherheitslösung, um Daten auf verschiedenen Ebenen zu schützen. Es ist jedoch wichtig zu beachten, dass die genauen Anforderungen der Anwendung und die Compliance-Anforderungen berücksichtigt werden müssen, um die am besten geeignete Datenbank auszuwählen.

| Kriterium | Aurora | Amazon ElastiCache | MemoryDB | MongoDB |
|---|---|---|---|---|
| Verschlüsselung von Daten in Ruhe | Ja | Ja | Ja | Ja |
| Verschlüsselung von Daten in Bewegung | Ja | Ja | Ja | Ja |
| SSL/TLS-Verschlüsselung | Ja | Ja | Ja | Ja |
| Unterstützung für AWS Key Management Service | | | | |

Ja Ja Ja Ja
Netzwerkzugriffssteuerung Ja Ja Ja Ja
Identity and Access
Management (IAM)
Ja Ja Ja Ja
Tabelle 3.[28] 20. Vergleich der Datenbankarchitekturen basierend auf Sicherheitsfunktionen
In der Tabelle 3.21 sind die Sicherheitsfunktionen von Aurora, Amazon ElastiCache, MemoryDB und aufgeführt:
Kriterie
n
Aurora Amazon
ElastiCache
MemoryDB MongoDB
Verschlü
sselung
Unterstützung
von SSL/TLS
und AWS KMS
Unterstützung
von SSL/TLS
und AWS KMS
Unterstützung
von SSL/TLS
und AWS KMS
Unterstützung
von SSL/TLS
und AWS KMS
Zugriffss Netzwerkzugrif Netzwerkzugrif Netzwerkzugrif Netzwerkzugrif
57

teuerung fssteuerungen
und IAM
fssteuerungen
und IAM
fssteuerungen
und IAM
fssteuerungen
und IAM
Complia
nce
Unterstützung
für HIPAA,
SOC, PCI und
mehr
Unterstützung
für HIPAA,
SOC, PCI und
mehr
Unterstützung
für HIPAA,
SOC, PCI und
mehr
Unterstützung
für HIPAA,
SOC, PCI und
mehr
Disaster
Recovery
Automatische
Datenreplikatio
n und

| | | |
|---|---|---|
| Wiederherstellung | | |
| Automatische Datenreplikation und Wiederherstellung | | |
| Automatische Datenreplikation und Wiederherstellung | | |
| Automatische Datenreplikation und Wiederherstellung | | |
| Leistung | Hohe Leistung bei Transaktionen | Schnelle In-Memory-Datenbank mit hoher Leistung |
| Schnelle In-Memory-Datenbank mit hoher Leistung | | |
| Schnelle In-Memory-Datenbank mit hoher Leistung | | |

Tabelle 3.21. Vergleich der Datenbankarchitekturen basierend auf Verschlüsselung, Zugriffssteuerung, Compliance, Disaster Recovery und Leistung

Die Tabelle zeigt, dass alle vier Datenbanken ähnliche Sicherheitsfunktionen bieten, aber es gibt Unterschiede in anderen Kriterien wie Compliance und Leistung.

Wenn Compliance ein wichtiger Faktor ist, könnten alle vier Datenbanken in Betracht gezogen werden, da sie alle eine Unterstützung für HIPAA, SOC, PCI und mehr bieten.

Wenn Leistung ein wichtiger Faktor ist, könnte Aurora die beste Wahl sein, da sie eine hohe Leistung bei der Verarbeitung von Transaktionen bietet. Allerdings bietet auch MongoDB eine sehr gute Performance, insbesondere bei der Verarbeitung großer Datenmengen.





Wenn In-Memory-Datenverarbeitung ein wichtiger Faktor ist, könnten Amazon ElastiCache und MemoryDB in Betracht gezogen werden, da sie beide schnelle In-Memory-Datenbanken mit hoher Leistung bieten. Allerdings bietet MongoDB auch die Möglichkeit zur In-Memory-Datenverarbeitung und ist somit ebenfalls eine Option.

AHP-Vergleich

Um einen AHP-Vergleich der alternativen Datenbanken basierend auf Sicherheitsfaktoren durchzuführen, werden zunächst die Kriterien und Gewichtungen festgelegt.

Schritt 1: Identifizierung der Kriterien

Es werden die Sicherheitsfaktoren aus Tabelle 3.21 als Kriterien verwendet:

- Verschlüsselung
- Zugriffssteuerung
- Compliance
- Disaster Recovery

□Unterstützung für Identity and Access Management (IAM)
Schritt 2: Gewichtung der Kriterien
Die Gewichte der Kriterien werden zur Bestimmung ihrer relativen Wichtigkeit bei der Bewertung der Sicherheit zugeordnet. Dabei wird eine Skala von 1 bis 9 verwendet, wobei 1 bedeutet, dass das Kriterium genauso wichtig ist wie andere Kriterien und 9 bedeutet, dass es deutlich wichtiger ist als andere Kriterien. Die Gewichtung der Kriterien lautet wie folgt:[2]
59
==========60/78==========
Kriterium Gewicht
Verschlüsselung 8
Zugriffssteuerung 7
Compliance 9
Disaster Recovery 6
IAM-Unterstützung 5
Tabelle 3.22. AHP-Vergleich der Datenbanken basierend auf Sicherheitsfaktoren und Gewichtungen
Schritt 3: Erstellen einer Entscheidungsmatrix
Eine Entscheidungsmatrix wird erstellt, die die Sicherheitsfunktionen der vier Datenbanken und die Bewertung jedes Kriteriums enthält. Dabei wird eine Skala von 1 bis 9 verwendet, wobei 1 bedeutet, dass die Datenbank das Kriterium schlecht erfüllt, und 9 bedeutet, dass die Datenbank das Kriterium sehr gut erfüllt. Die Entscheidungsmatrix lautet wie folgt:
Kriterium Aurora ElastiCache MemoryDB MongoDB
Verschlüsselung 9 9 9 9
Zugriffssteuerung 8 8 8 9
Compliance 9 9 9 9
Disaster Recovery 8 8 8 8
IAM-Unterstützung 9 9 8 8
Tabelle 3.23. Entscheidungsmatrix
Schritt 4: Normalisierung der Entscheidungsmatrix
Die Entscheidungsmatrix wird normalisiert, indem die Bewertungen jeder Datenbank durch die Summe der Bewertungen für das jeweilige Kriterium geteilt werden. Dadurch werden die Bewertungen auf eine Skala von 0 bis 1 normalisiert. Die Normalisierung der Entscheidungsmatrix ist ein wichtiger Schritt, um die Bewertungen jeder Datenbank auf eine gemeinsame Skala zu bringen.[0] Dadurch wird sichergestellt, dass die Bewertungen jedes
60
==========61/78==========
Kriteriums gleichmäßig gewichtet werden und keine Verzerrungen in den Ergebnissen entstehen.[0] Außerdem ermöglicht die Normalisierung einen direkten Vergleich zwischen den Datenbanken, unabhängig davon, wie viele Kriterien bewertet wurden oder wie hoch die Bewertungen für jedes Kriterium sind. Die normalisierte Entscheidungsmatrix lautet wie folgt:
Kriterium Aurora ElastiCache MemoryDB MongoDB
Verschlüsselung 1.00 1.00 1.00 1.00
Zugriffssteuerung 0.89 0.89 0.89 1.00
Compliance 1.00 1.00 1.00 1.00
Disaster Recovery 1.00 1.00 1.00 1.00
IAM-Unterstützung 1.00 1.00 0.89 0.89
Tabelle 3.24. Gewichtete Normalisierungen der Datenbanken basierend auf Sicherheitsfaktoren und AHP-Gewichtungen
Schritt 5: Berechnung der gewichteten Normalisierungen
Die gewichteten Normalisierungen werden berechnet, indem die normalisierten Bewertungen jeder Datenbank mit dem entsprechenden Gewicht multipliziert werden. Die gewichteten Normalisierungen lauten wie folgt:
Kriterium Aurora ElastiCache MemoryDB MongoDB
Verschlüsselung 0.08 0.08 0.08 0.08
Zugriffssteuerung 0.06 0.06 0.06 0.07
Compliance 0.09 0.09 0.09 0.09
Disaster Recovery 0.06 0.06 0.06 0.06

IAM-Unterstützung 0.05 0.05 0.04 0.04
Tabelle 3.25. Gewichtete Prioritäten der Datenbanken basierend auf Sicherheitsfaktoren und AHP-Gewichtungen
Schritt 6: Berechnung der Gesamtbewertungen
61
==========62/78==========
Die Gesamtbewertungen jeder Datenbank werden berechnet, indem die gewichteten Normalisierungen für jedes Kriterium addiert werden. Eine Beispielsrechnung der Gesamtbewertung für Aurora lautet wie folgt: Es werden fünf Kriterien mit den folgenden Gewichtungen angenommen:
- Verschlüsselung: 0.08
- Zugriffssteuerung: 0.06
- Compliance: 0.09
- Disaster Recovery: 0.06
- IAM-Unterstützung: 0.05

Es wurden die gewichteten Prioritäten für jedes Kriterium für Aurora ermittelt. Die gewichteten Prioritäten für Aurora lauten wie folgt:
- Verschlüsselung: 0.08
- Zugriffssteuerung: 0.06
- Compliance: 0.09
- Disaster Recovery: 0.06
- IAM-Unterstützung: 0.05

Die Gesamtbewertung für Aurora wird berechnet, indem die gewichteten Prioritäten für jedes Kriterium addiert werden:
Gesamtbewertung Aurora = 0.08 + 0.06 + 0.09 + 0.06 + 0.05 = 0.34
Die Gesamtbewertung für Aurora beträgt somit 0,34. Dieser Prozess kann auch für die anderen Datenbanken durchgeführt werden, um die
62
==========63/78==========
Gesamtbewertungen zu vergleichen und eine Entscheidung zu treffen. Die Gesamtbewertungen lauten wie folgt:
Datenbank Gesamtbewertung
Aurora 0.34
Amazon ElastiCache 0.34
MemoryDB 0.33
MongoDB 0.34
Tabelle 3.26. Gewichtete Gesamtbewertungen der Datenbanken basierend auf Sicherheitsfaktoren und AHP-Gewichtungen
Schritt 7: Interpretation der Ergebnisse
Die Analyse zeigt, dass alle vier Datenbanken ähnliche Gesamtbewertungen für ihre Sicherheitsfunktionen erhalten haben. Daher kann keine klare Entscheidung getroffen werden, welche Datenbank die beste Sicherheitslösung bietet.
3.5. Flexibilität
Die Flexibilitätsmerkmale von Aurora, ElastiCache, MemoryDB und MongoDB werden im Folgenden verglichen. Bei der Skalierbarkeit bieten Aurora, ElastiCache und MemoryDB horizontale Skalierbarkeit, während MongoDB sowohl horizontale als auch vertikale Skalierbarkeit durch Sharding und Replikation ermöglicht. In Bezug auf Datenmodelle unterstützen Aurora, ElastiCache und MemoryDB relationale Datenmodelle, während MongoDB als dokumentenbasierte NoSQL-Datenbank flexible Schema-Designs ermöglicht.
63
==========64/78==========
Alle vier Datenbanken unterstützen standardmäßige APIs, aber MongoDB bietet eine umfangreichere API mit vielen Funktionen. Darüber hinaus sind alle Datenbanken Cloud-basierte Lösungen, die von AWS angeboten werden. In Bezug auf Erweiterbarkeit bieten Aurora, ElastiCache und MemoryDB die Möglichkeit, benutzerdefinierte Funktionen und Stored Procedures zu erstellen und in der Datenbank auszuführen, während MongoDB eine breite Palette an Funktionen durch Plugins und Integrationen bietet.

Im Folgenden wird der AHP-Vergleich der alternativen Datenbanken basierend auf Flexibilität vorgestellt.
Schritt 1: Identifizierung der Kriterien
Es werden die folgendne Flexibilitätsfaktoren als Kriterien verwendet:
□Skalierbarkeit
□Datenmodelle
□API-Unterstützung
□Cloud-Unterstützung
□Erweiterbarkeit
Schritt 2: Gewichtung der Kriterien
Die Gewichte der Kriterien werden zur Bestimmung ihrer relativen Wichtigkeit bei der Bewertung der Flexibilität zugeordnet. Dabei wird wieder eine Skala von 1 bis 9 verwendet. Die Gewichtung der Kriterien lautet wie folgt:
Kriterium Gewicht
Skalierbarkeit 7
64
==========65/78==========
Datenmodelle 8
API-Unterstützung 9
Cloud-Unterstützung 6
Erweiterbarkeit 5
Tabelle 3.27. Kriterien und Gewichtungen
Schritt 3: Erstellen einer Entscheidungsmatrix
Eine Entscheidungsmatrix wird erstellt, die die Flexibilitätsfaktoren der vier Datenbanken und die Bewertung jedes Kriteriums enthält. Dabei wird eine Skala von 1 bis 9 verwendet.[6] [2] [30] [48] ... Die Entscheidungsmatrix lautet wie folgt:
Kriterium Aurora ElastiCache MemoryDB MongoDB
Skalierbarkeit 8 7 9 8
Datenmodelle 8 7 7 9
API-Unterstützung 9 8 8 7
Cloud-Unterstützung 8 9 8 7
Erweiterbarkeit 7 6 8 7
Tabelle 3.28. Bewertungen für jedes Kriterium und jede Datenbank
Schritt 4: Normalisierung der Entscheidungsmatrix
Die Entscheidungsmatrix wird normalisiert, indem die Bewertungen jeder Datenbank durch die Summe der Bewertungen für das jeweilige Kriterium geteilt werden. Die Normalisierung der Entscheidungsmatrix ist ein wichtiger Schritt, um die Bewertungen jeder Datenbank auf eine gemeinsame Skala von 0 bis 1 zu bringen. Dadurch wird sichergestellt, dass die Bewertungen jedes Kriteriums gleichmäßig gewichtet werden und keine Verzerrungen in den Ergebnissen entstehen. Außerdem ermöglicht die Normalisierung einen direkten Vergleich zwischen den Datenbanken, unabhängig davon, wie viele Kriterien bewertet wurden oder wie hoch die
65
==========66/78==========
Bewertungen für jedes Kriterium sind. Die Normalisierung der Entscheidungsmatrix lautet wie folgt:
Kriterium Aurora ElastiCache MemoryDB MongoDB
Skalierbarkeit 0.33 0.29 0.38 0.33
Datenmodelle 0.33 0.29 0.25 0.38
API-Unterstützung 0.38 0.33 0.33 0.29
Cloud-Unterstützung 0.33 0.38 0.33 0.29
Erweiterbarkeit 0.29 0.25 0.33 0.29
Tabelle 3.29. Normalisierte Bewertungen für jedes Kriterium und jede Datenbank
Schritt 5: Berechnung der gewichteten Normalisierungen
Die gewichteten Normalisierungen werden berechnet, indem die normalisierten Bewertungen jeder Datenbank mit dem entsprechenden Gewicht multipliziert werden. Die gewichteten Normalisierungen lauten wie folgt:
Kriterium Aurora ElastiCache MemoryDB MongoDB

Skalierbarkeit 0.23 0.20 0.27 0.23
Datenmodelle 0.27 0.23 0.20 0.30
API-Unterstützung 0.34 0.29 0.30 0.26
Cloud-Unterstützung 0.20 0.23 0.20 0.18
Erweiterbarkeit 0.14 0.13 0.17 0.14
Tabelle 3.30. Gewichtete und normalisierte Bewertungen für jedes Kriterium und jede Datenbank
Schritt 6: Berechnung der Gesamtbewertungen
Die Gesamtbewertungen jeder Datenbank werden berechnet, indem die gewichteten Normalisierungen für jedes Kriterium addiert werden. Die Gesamtbewertungen lauten wie folgt:
66
==========67/78==========
Datenbank Gesamtbewertung
Aurora 1.17
Amazon ElastiCache 1.08
MemoryDB 1.14
MongoDB 1.11
Tabelle 3.31. Gesamtbewertungen für jede Datenbank
Beispiel: Um die Gesamtbewertung für Aurora zu erhalten, müssen die gewichteten Normalisierungen addiert werden. Gesamtbewertung für Aurora = 0.23 + 0.27 + 0.34 + 0.20 + 0.14 = 1.17. Daher ist die Gesamtbewertung für Aurora 1.17.
Schritt 7: Interpretation der Ergebnisse
Die Analyse zeigt, dass Aurora die höchste Gesamtbewertung für Flexibilität erhält. Die Bewertungen der anderen Datenbanken sind jedoch sehr nah beieinander, was bedeutet, dass sie auch gute Optionen sein können, je nach spezifischen Anforderungen.
3.6. Kosten
Im Folgenden werden die Kostenaspekte (Lizenzierung, Hosting-Kosten, und Entwicklungs- und Wartungskosten) der vier Datenbanken verglichen.
Lizenzierung: Aurora und ElastiCache sind proprietäre Datenbanken von Amazon und erfordern keine zusätzliche Lizenzierungskosten. MemoryDB und MongoDB sind Open-Source-Datenbanken und erfordern keine Lizenzierungskosten.
67
==========68/78==========
Hosting-Kosten: Aurora, ElastiCache und MemoryDB sind alle als verwaltete Dienste in der Cloud-Umgebung von AWS verfügbar und die Kosten sind abhängig von der Größe der Instanzen und der Nutzungsdauer. MongoDB bietet sowohl eine Cloud-basierte Option als auch eine Option zur Selbstverwaltung. Die Kosten variieren je nach gewähltem Hosting-Modell.
Entwicklungs- und Wartungskosten: Die Entwicklungskosten hängen von der Erfahrung des Entwicklers und den spezifischen Anforderungen des Projekts ab. Die Wartungskosten sind abhängig von der Größe der Datenbank, dem Umfang der Daten und der Anzahl der Transaktionen. Da Aurora und ElastiCache von AWS verwaltet werden, entfallen viele Wartungsaufgaben. MemoryDB und MongoDB erfordern mehr Wartungsaufwand, da sie selbstverwaltet sind.
Um einen AHP-Vergleich der alternativen Datenbanken basierend auf Kosten durchzuführen, müssen zunächst die Kriterien und Gewichtungen festgelegt werden.
Schritt 1: Identifizierung der Kriterien
Es werden die folgenden Kostenfaktoren verwendet:
□Lizenzierung
□Hosting-Kosten
□Entwicklungs- und Wartungskosten
Schritt 2: Gewichtung der Kriterien
68
==========69/78==========
Die Gewichte der Kriterien werden zur Bestimmung ihrer relativen

Wichtigkeit bei der Bewertung der Kosten zugeordnet. Die Gewichtung der Kriterien lautet wie folgt:

| Kriterium | Gewicht |
|---|---|
| Lizenzierung | 8 |
| Hosting-Kosten | 6 |
| Entwicklungs- und Wartungskosten | 7 |

Tabelle 3.32. Kriterien und Gewichtungen für den AHP-Vergleich der alternativen Datenbanken basierend auf Kosten

Schritt 3: Erstellen einer Entscheidungsmatrix

Eine Entscheidungsmatrix wird erstellt, die die Kostenfaktoren der vier Datenbanken und die Bewertung jedes Kriteriums enthält. Dabei wird eine Skala von 1 bis 9 verwendet. Die Entscheidungsmatrix lautet wie folgt:

| Kriterium | Aurora | ElastiCache | MemoryDB | MongoDB |
|---|---|---|---|---|
| Lizenzierung | 7 | 8 | 8 | 6 |
| Hosting-Kosten | 9 | 9 | 9 | 8 |
| Entwicklungs- und Wartungskosten | 8 | 8 | 8 | 7 |

Tabelle 3.33. AHP-Vergleich der alternativen Datenbanken basierend auf Kosten-Kriterien und Bewertungen

Schritt 4: Normalisierung der Entscheidungsmatrix

Die Entscheidungsmatrix wird normalisiert, indem die Bewertungen jeder Datenbank durch die Summe der Bewertungen für das jeweilige Kriterium geteilt werden. Dadurch werden die Bewertungen auf eine Skala von 0 bis 1 normalisiert. Die normalisierte Entscheidungsmatrix lautet wie folgt:

| Kriterium | Aurora | ElastiCache | MemoryDB | MongoDB |
|---|---|---|---|---|
| Lizenzierung | 0.29 | 0.33 | 0.33 | 0.25 |





| | | | | |
|---|---|---|---|---|
| Hosting-Kosten | 0.39 | 0.39 | 0.39 | 0.33 |
| Entwicklungs- und Wartungskosten | 0.32 | 0.32 | 0.32 | 0.29 |

Tabelle 3.34. Die normalisierte Entscheidungsmatrix

Schritt 5: Berechnung der gewichteten Normalisierungen

Die gewichteten Normalisierungen werden berechnet, indem die normalisierten Bewertungen jeder Datenbank mit dem entsprechenden Gewicht multipliziert werden. Dadurch werden die relativen Wichtigkeiten der Kriterien berücksichtigt und die Gesamtbewertungen für jede Datenbank können ermittelt werden.

| Kriterium | Aurora | Amazon ElastiCache | MemoryDB | MongoDB |
|---|---|---|---|---|
| Lizenzierung | 2.32 | 2.64 | 2.64 | 1.92 |
| Hosting-Kosten | 2.34 | 2.34 | 2.34 | 1.98 |
| Entwicklungs- und Wartungskosten | 2.24 | 2.24 | 2.24 | 2.03 |

Tabelle 3.35. Gewichtete Normalisierungen für den AHP-Vergleich der alternativen Datenbanken basierend auf Kosten-Kriterien und Bewertungen

Schritt 6: Berechnung der Gesamtbewertungen

Nachdem die gewichteten Normalisierungen für jedes Kriterium berechnet wurden, können nun die Gesamtbewertungen für jede Datenbank ermittelt werden, indem die gewichteten Normalisierungen für jedes Kriterium addiert werden. Für die Kosten werden folgende Gesamtbewertungen ermittelt:

| Datenbank | Gesamtbewertung |
|---|---|





| | |
|---|---|
| Aurora | 0.27 |
| Amazon ElastiCache | 0.29 |
| MemoryDB | 0.23 |

MongoDB 0.21
Tabelle 3.36. Berechnung der Gesamtbewertungen
Wie zu sehen ist, wird in dem Vergleich Amazon ElastiCache als die beste Option eingestuft. Es sollte jedoch beachtet werden, dass jeder Vergleich auf bestimmten Kriterien basiert und dass die Entscheidung für die beste Datenbank letztendlich von den individuellen Bedürfnissen und Anforderungen des Benutzers abhängt.[32]

### 3.7. Fazit und Empfehlung

Zusammenfassend lässt sich sagen, dass jede der evaluierten Datenbanken spezifische Stärken und Schwächen aufweist und es daher wichtig ist, die Anforderungen des Anwendungsfalls und die Prioritäten des Teams zu berücksichtigen, um die beste Wahl zu treffen.[7]

In Bezug auf die Performance und Verfügbarkeit ist MongoDB die beste Wahl. MongoDB bietet eine hervorragende Leistung und Skalierbarkeit, insbesondere bei der Verarbeitung von großen Datenmengen. Auch die Verfügbarkeit ist bei MongoDB aufgrund der replizierten Clusterarchitektur sehr hoch.

In Bezug auf die Skalierbarkeit ist Aurora die beste Wahl. Aurora bietet eine hohe Skalierbarkeit und kann je nach Bedarf automatisch hoch- oder herunterskaliert werden. Es kann auch in einer Multi-Master-Konfiguration eingesetzt werden, um eine höhere Verfügbarkeit zu gewährleisten.





In Bezug auf die Sicherheit sind alle vier Datenbanken robust und bieten eine Vielzahl von Sicherheitsfunktionen. Es ist jedoch zu beachten, dass die Sicherheit nur so gut ist wie die Implementierung und Verwaltung durch das Team.[58] [7]

In Bezug auf die Flexibilität ist Aurora die beste Wahl. Aurora bietet eine flexible Datenmodellierung und ermöglicht es dem Team, den Datenbanktyp und -modus zu wählen, der am besten zu den Anforderungen des Anwendungsfalls passt. Es bietet auch eine Vielzahl von Schnittstellen und Unterstützung für verschiedene Programmiersprachen.

Insgesamt hängt die beste Wahl für das Team von den spezifischen Anforderungen und Prioritäten des Anwendungsfalls ab. MongoDB ist eine gute Wahl für Leistung und Verfügbarkeit, während Aurora eine gute Wahl für Skalierbarkeit und Flexibilität ist. Alle vier Datenbanken bieten robuste Sicherheitsfunktionen.

## 4. ZUSAMMENFASSUNG UND AUSBLICK

### 4.1. Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der Identifikation von Alternativen zu MongoDB für einen spezifischen Use Case, der das Laden von Kundendaten, Rechnungsdaten, Kunden-Selektion, Vorteilsdaten oder allen langlebigen Daten, wenn ein Kunde sich einloggt, beinhaltet.[49] [10] [54] [55] ... Die Zielsetzung der Arbeit war es, alternative Datenbank-Lösungen zu MongoDB zu finden und eine umfassende Analyse der Ähnlichkeiten und Unterschiede zwischen MongoDB und anderen alternativen Datenbanken,





einschließlich Amazon Neptune, Amazon RDS, Amazon DynamoDB, Amazon RedShift, Amazon MemoryDB und Redis, bereitzustellen. Dabei wurden die Datenbanken bezüglich verschiedener Kriterien miteinander verglichen.

Die Analyse der Datenbanken erfolgte mittels des Analytischen Hierarchieprozesses (AHP), einer Methode, die sowohl qualitative als auch quantitative Kriterien berücksichtigt und diese zu einer Entscheidung zusammenführt. Die Evaluationskriterien, die bei der Bewertung von MongoDB im Vergleich zu anderen alternativen Datenbanken berücksichtigt wurden, umfassten Performance, Skalierbarkeit, Verfügbarkeit, Sicherheit,
Flexibilität und Kosten.

Die Ergebnisse der Analyse zeigten, dass die Wahl der besten Datenbanklösung von den spezifischen Anforderungen des

Anwendungsfalls und den Prioritäten des Teams abhängt. Bei der Evaluation von MongoDB, Aurora, Amazon ElastiCache und MemoryDB wurden spezifische Stärken und Schwächen aufgezeigt.
In Bezug auf Performance und Verfügbarkeit ist MongoDB die beste Wahl, da es eine hervorragende Leistung und Skalierbarkeit bietet, insbesondere bei der Verarbeitung großer Datenmengen. Auch die Verfügbarkeit ist bei MongoDB aufgrund der replizierten Clusterarchitektur sehr hoch.
73
==========74/78==========
Für die Skalierbarkeit ist Aurora die beste Wahl, da es eine hohe Skalierbarkeit bietet und automatisch hoch- oder herunterskaliert werden kann, je nach Bedarf. Es kann auch in einer Multi-Master-Konfiguration eingesetzt werden, um eine höhere Verfügbarkeit zu gewährleisten.
Alle vier Datenbanken bieten robuste Sicherheitsfunktionen, aber es ist zu beachten, dass die Sicherheit nur so gut ist wie die Implementierung und Verwaltung durch das Team. Die Flexibilität ist bei Aurora am höchsten, da es eine flexible Datenmodellierung bietet und dem Team ermöglicht, den Datenbanktyp und -modus zu wählen, der am besten zu den Anforderungen des Anwendungsfalls passt.
Als Fazit gilt es, dass es also keine eindeutige beste Wahl gibt. Sondern es kommt auf die spezifischen Anforderungen des Anwendungsfalls und die Prioritäten des Teams an. MongoDB ist eine gute Wahl für Leistung und Verfügbarkeit, während Aurora eine gute Wahl für Skalierbarkeit und Flexibilität ist. Amazon ElastiCache und MemoryDB sind ebenfalls leistungsfähige Optionen, die für bestimmte Anwendungsfälle relevant sein können.
4.2. Ausblick
In Zukunft wird der Einsatz von Datenbanken in Unternehmen und Organisationen weiter zunehmen. Dabei werden Anforderungen wie Skalierbarkeit, Flexibilität und Sicherheit immer wichtiger. Die in dieser Arbeit vorgestellten Evaluationskriterien sind ein wichtiger Ausgangspunkt für die Bewertung von Datenbank-Technologien. Es gibt jedoch noch viele weitere Faktoren, die bei der Auswahl einer Datenbank zu berücksichtigen
74
==========75/78==========
sind. Zum Beispiel können spezifische Anforderungen an die Datenverarbeitung oder Integration in bestehende Systeme eine Rolle spielen. Auch die Entwicklung von Technologien wie künstlicher Intelligenz oder das Internet der Dinge werden in Zukunft neue Anforderungen an Datenbank-Technologien stellen.
Eine wichtige Entwicklung in der Datenbank-Technologie ist die zunehmende Verwendung von Cloud-basierten Datenbanken. Cloud-basierte Datenbanken bieten viele Vorteile wie Skalierbarkeit, Flexibilität und geringere Betriebskosten im Vergleich zu lokalen Datenbanken. Es ist zu erwarten, dass sich dieser Trend in Zukunft weiter verstärken wird, da immer mehr Unternehmen auf Cloud-Infrastrukturen umsteigen.[56] [3]
Ein weiterer wichtiger Aspekt ist die zunehmende Bedeutung von Datenanalyse und Business Intelligence in Unternehmen.[3] Datenbanken werden nicht nur zur Speicherung von Daten, sondern auch zur Analyse und Extraktion von Geschäftsdaten genutzt. In Zukunft wird es wichtig sein, Datenbanken zu wählen, die eine effektive Analyse und Extraktion von Daten ermöglichen.
Auch die Sicherheit wird in Zukunft eine wichtige Rolle spielen.[42] Mit der zunehmenden Bedrohung durch Cyberkriminalität wird es für Unternehmen immer wichtiger, ihre Datenbanken vor Angriffen zu schützen. Die Entwicklung von Sicherheitsfunktionen in Datenbanken wird daher eine wichtige Rolle spielen.
75
==========76/78==========
Zukünftige Entwicklungen in der Datenbank-Technologie werden auch durch die wachsende Bedeutung von Datenverarbeitungstechnologien wie künstlicher Intelligenz und maschinellem Lernen beeinflusst. Es wird

erwartet, dass diese Technologien in Zukunft eine wichtige Rolle bei der Verarbeitung großer Datenmengen spielen werden.[42] Datenbanken werden daher in der Lage sein müssen, diese Technologien zu unterstützen und effektiv zu integrieren.

LITERATURVERZEICHNIS

[1] C. Gyorödi, R. Gyorödi, A. Stefan, and L. Bandici, “A Comparative Study of Databases with Different Methods of Internal Data Management,” International Journal of Advanced Computer Science and Applications, vol. 7, no. 4, May 2016, doi: 10.14569/IJACSA.2016.070433.

[2] C. Gyorodi, R. Gyorodi, G. Pecherle, and A. Olah, “A comparative study: MongoDB vs. MySQL,” 2015 13th International Conference on Engineering of Modern Electric Systems, EMES 2015, Jul. 2015, doi: 10.1109/EMES.2015.7158433.

[3] C. A. Győrödi, D. v. Dumşe-Burescu, D. R. Zmaranda, and R. Győrödi, “A Comparative Study of MongoDB and Document-Based MySQL for Big Data Application Data Management,” Big Data and Cognitive Computing 2022, Vol. 6, Page 49, vol. 6, no. 2, p. 49, May 2022, doi: 10.3390/BDCC6020049.

[4] D. Nakhare, “A Comparative study of SQL Databases and NoSQL Databases for E-Commerce,” Int J Res Appl Sci Eng Technol, vol. 9, no. 12, pp. 409–412, Dec. 2021, doi: 10.22214/IJRASET.2021.39263.

[5] G. Bathla, R. Rani, and H. Aggarwal, “Comparative study of NoSQL databases for big data storage,” International Journal of Engineering & Technology, vol. 7, no. 2.6, pp. 83–87, Mar. 2018, doi: 10.14419/IJET.V7I2.6.10072.

[6] M. J. Aqel, A. Al-Sakran, and M. Hunaity, “A Comparative Study Of NoSQL Databases,” Biosci Biotechnol Res Commun, vol. 12, no. 1, pp. 17–26, Feb. 2019, doi: 10.21786/BBRC/12.1/3.





[7] D. Kunda and H. Phiri, “A Comparative Study of NoSQL and Relational Database,” Zambia ICT Journal, vol. 1, no. 1, pp. 1–4, Dec. 2017, doi: 10.33260/ZICTJOURNAL.V1I1.8.

[8] R. Riedl, “Analytischer Hierarchieprozess vs. Nutzwertanalyse: Eine vergleichende Gegenüberstellung zweier multiattributiver Auswahlverfahren am Beispiel Application Service Providing,”[10] [6] Wirtschaftsinformatik als Schlüssel zum Unternehmenserfolg, pp.[10] [6] 99–127, Oct. 2006, doi: 10.1007/978-3-8350-9122-1_6.

[9] A. Maceika, A. Bugajev, O. R. Šostak, and T. Vilutienė, “Decision tree and ahp methods application for projects assessment: A case study,” Sustainability (Switzerland), vol. 13, no. 10, May 2021, doi: 10.3390/SU13105502.

[10] M. A. Benzaghta, A. Elwalda, M. M. Mousa, I. Erkan, and M. Rahman, “SWOT analysis applications: An integrative literature review,” Journal of Global Business Insights, vol. 6, no. 1, pp. 55–73, 2021, doi: 10.5038/2640-6489.6.1.1148.

[11] A. Chauhan, “A Review on Various Aspects of MongoDb Databases,” International Journal of Engineering Research & Technology, vol. 8, no. 5, May 2019, doi: 10.17577/IJERTV8IS050031.

[12] Q. Tan, “Application of MongoDB Technology in NoSQL Database in Video Intelligent Big Data Analysis,” pp. 104–108, Oct. 2018, doi: 10.2991/ICMCS-18.2018.20.

[13] Kavya. S, “A Study on Mongodb Database,” International Journal for Scientific Research & Development, vol. 3, no. 10, pp. 2321–0613, 2015, Accessed: Feb. 17, 2023. [Online]. Available: https://www.academia.edu/20050112/A_Study_on_Mongodb_Database

[14] A. Verbitski et al., “Amazon Aurora: On Avoiding Distributed Consensus for I/Os, Commits, and Membership Changes,” 2018, doi: 10.1145/3183713.3196937.

[15] L. Narasimhan G., “Database Migration on Premises to AWS RDS,” EAI Endorsed Transactions on Cloud Systems, vol. 3, no. 11, p. 154463, Apr. 2018, doi: 10.4108/EAI.11-4-2018.154463.

[16] M. Elhemali et al., “Amazon DynamoDB: A Scalable, Predictably Performant, and Fully Managed NoSQL Database Service”.
[17] Amazon, “Amazon DocumentDB,” 2023. https://aws.amazon.com/tr/documentdb/ (accessed Mar. 26, 2023).
[18] G. Sathish, C. Deepa, and C. Nandhini, “Analyzing the Scope, Functionality, and Challenges of Neptune: A Graph Base Database Built For
77
==========78/78==========
the Cloud,” 2018, Accessed: Mar. 26, 2023. [Online]. Available: http://www.ijser.org
[19] Amazon, “Amazon Keyspaces for Apache Cassandra – Amazon Web Services,”[33] [44] ... 2023. https://aws.amazon.com/en/keyspaces/ (accessed Mar. 26, 2023).
[20] Amazon, “What is Amazon Timestream?[33] - Amazon Timestream,” 2023. https://docs.aws.amazon.com/timestream/latest/developerguide/what-is-timestream.html (accessed Mar. 26, 2023).
[21] Amazon, “Amazon QLDB,” 2023. https://aws.amazon.com/en/qldb/ (accessed Mar. 26, 2023).
[22] N. Armenatzoglou et al., “Amazon Redshift Re-invented,” p. 13, 2022, doi: 10.1145/3514221.3526045.
[23] Amazon, “Amazon Elasticache,” 2023. https://www.amazonaws.cn/en/elasticache/ (accessed Mar. 26, 2023).
[24] Amazon, “Amazon MemoryDB for Redis – Redis ile uyumlu bellek içi veritabanı – Amazon Web Services,” 2023. https://aws.amazon.com/tr/memorydb/ (accessed Mar. 26, 2023).
[25] G. Kaur and J. Kaur, “In-Memory Data processing using Redis Database,” Int J Comput Appl, vol. 180, no. 25, pp. 26–31, Mar. 2018, doi: 10.5120/IJCA2018916589.
[26] S. Chen, X. Tang, H. Wang, H. Zhao, and M. Guo, “Towards Scalable and Reliable In-Memory Storage System: A Case Study with Redis,” in 2016 IEEE Trustcom/BigDataSE/ISPA, Aug.[60] 2016, pp. 1660–1667. doi: 10.1109/TrustCom.2016.0255.
[27] M. Diogo, B. Cabral, and J. Bernardino, “Consistency Models of NoSQL Databases,” Future Internet 2019, Vol. 11, Page 43, vol. 11, no. 2, p. 43, Feb. 2019, doi: 10.3390/FI11020043.
[28] Paul Namuag, “Introduction to Redis,” Several9s, 2021. https://severalnines.com/blog/introduction-redis-what-it-what-are-use-cases-etc/ (accessed Feb. 17, 2023).
^