Grundlagen und Anwendungsbeispiele für das NOSQL-Datenbankmanagementsystem DynamoDB

Seminararbeit

vorgelegt am 25. Januar 2017

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Kurs WWI2014I

von

|  |  |
| --- | --- |
| Betreuer in der Ausbildungsstätte: | DHBW Stuttgart: |
| Prof. Dr. Nikolai Preiß DHBW Stuttgart | [Titel, Vorname und Nachname des wissenschaftl. Betreuers/Prüfers] |

Kerstin Farke  
Dhruv Mahandru  
Henning Mohr  
Käthe Vrettos

**Inhaltsverzeichnis**

[Abkürzungsverzeichnis (bei Bedarf) IV](#_Toc469343345)

[Abbildungsverzeichnis (bei Bedarf) V](#_Toc469343346)

[Tabellenverzeichnis (bei Bedarf) VI](#_Toc469343347)

[1 Einleitung 1](#_Toc469343348)

[2 Theoretische Grundlagen 2](#_Toc469343349)

[2.1 Definitionen 2](#_Toc469343350)

[2.2 Datenbank Prinzipien 2](#_Toc469343351)

[2.2.1 ACID 2](#_Toc469343352)

[2.2.2 BASE 3](#_Toc469343353)

[2.2.3 CAP-Theorem 3](#_Toc469343354)

[2.3 Entwicklungen in der Datenbankbranche 4](#_Toc469343355)

[2.4 Datenbankmodelle 5](#_Toc469343356)

[2.4.1 Relationale Datenbanken 5](#_Toc469343357)

[2.4.2 NoSQL Datenbanken 6](#_Toc469343358)

[2.4.3 Zusammenfassung 8](#_Toc469343359)

[3 DynamoDB 10](#_Toc469343360)

[3.1 Generelle Informationen über AWS 10](#_Toc469343361)

[3.1.1 Infrastructre as a Serivice (IaaS)/Kunden 10](#_Toc469343362)

[3.1.2 Gründe für die Verwendung von AWS/Entwicklungsgeschichte (evtl.) 10](#_Toc469343363)

[3.1.3 Ziele 10](#_Toc469343364)

[3.1.4 Vor- und Nachteile von AWS 10](#_Toc469343365)

[3.2 Was ist DynamoDB 10](#_Toc469343366)

[3.2.1 Database as a Service (DBaaS) Definition 10](#_Toc469343367)

[3.2.2 Geschichte der DynamoDB 10](#_Toc469343368)

[3.2.3 Verwendung der DynamoDB in der Amazon Infrastruktur 10](#_Toc469343369)

[3.2.4 Preismodell 10](#_Toc469343370)

[3.3 Art der Datenverwaltung/Datenmodell 10](#_Toc469343371)

[3.3.1 Tabelle, Elemente und Attribute Konzept 10](#_Toc469343372)

[3.3.2 Eigenschaften (Fully managed, Durable, Scalable, Fast, Simple Administration, Flexible, Fault Tolerance, Indexing, Secure, Cost Effective) 10](#_Toc469343373)

[3.3.3 Datenspeicherung über die SSD 10](#_Toc469343374)

[3.3.4 Key-Value Eigenschaft 10](#_Toc469343375)

[3.3.5 Datentypen 10](#_Toc469343376)

[3.4 Vergleich mit anderes NoSQL-Lösungen 10](#_Toc469343377)

[3.4.1 Microsoft Azure 10](#_Toc469343378)

[3.4.2 Redis 10](#_Toc469343379)

[3.4.3 Vorteile und Nachteile von DynamoDB 10](#_Toc469343380)

[4 Praxis 11](#_Toc469343381)

[4.1 Insgesamt zum Zurechtfinden 11](#_Toc469343382)

[4.2 Testdatenbank mit Testdaten 11](#_Toc469343383)

[4.3 Datenbanksprache 11](#_Toc469343384)

[4.4 GUI/API 11](#_Toc469343385)

[4.5 Ablageform der Daten 11](#_Toc469343386)

[5 Schluss 12](#_Toc469343387)

[5.1 Fazit 12](#_Toc469343388)

[5.2 Ausblick 12](#_Toc469343389)

[Anhang (bei Bedarf) 13](#_Toc469343390)

[Quellenverzeichnis 14](#_Toc469343391)

[Ehrenwörtliche Erklärung 15](#_Toc469343392)

# Abkürzungsverzeichnis (bei Bedarf)

AktG = Aktiengesetz

AWS = Amazon Web Services

BFH = Bundesfinanzhof

# Abbildungsverzeichnis (bei Bedarf)

[Abb. 1: Eine Abbildung 2](#_Toc444006772)

# Tabellenverzeichnis (bei Bedarf)

[Tabelle 1: Eine Beispieltabelle 2](#_Toc444006773)

1. Einleitung
2. Theoretische Grundlagen
   1. Definitionen

* Datenbank
* Datenbankmanagementsystem
* Database as a Service
  1. Datenbank Prinzipien

Datenbanken folgen unterschiedlichen Prinzipien. So basieren relationale Datenbanken einem anderen Prinzip als beispielsweise NoSQL Datenbanken. Im Folgenden werden die drei meistgenutzten Prinzipien erläutert und differenziert.

* + 1. ACID

Das ACID-Prinzip steht für Atomarität, Konsistenz, Isolation und Dauerhaftigkeit der durchzuführenden Transaktionen,[[1]](#footnote-2) und wird besonders in der SQL Umgebung genutzt.[[2]](#footnote-3)

Die Atomarität steht sicher, dass eine Transaktion ganz oder gar nicht durchgeführt wird. Kann ein Teil der Transaktion also nicht komplett ausgeführt werden, so schlägt es als Gesamtes fehl.[[3]](#footnote-4) Damit wird entweder die gesamte, oder keine, Transaktion ausgeführt.[[4]](#footnote-5) Die Konsistenz sichert, dass die Datenbank sowohl vor als auch nach der Transaktion einen soliden Status vorweisen kann,[[5]](#footnote-6) während die Isolation sicherstellt, dass mehrere Transaktionen, welche zur selben Zeit ausgeführt werden, nicht die jeweils andere Transaktion beeinflussen, und somit isoliert durchgeführt werden. Dementsprechend müssen gleichzeitig ausgeführte Transaktionen „serializable“ sein.[[6]](#footnote-7) Zur Sicherstellung des permanenten Status einer Transaktion wird das ACID-Prinzip der Dauerhaftigkeit genutzt. Sobald eine Transaktion abgeschlossen worden ist und Daten gespeichert wurden, sollten die geschriebenen Daten dauerhaft verfügbar sein, auch bei einem Crash oder Fehler des Systems.[[7]](#footnote-8)

* + 1. BASE

Das BASE-Paradigma steht für „Basically Available, Soft State, Eventual Consistency“ und stellt generell das Gegenstück zu den ACID-Proportionen dar.[[8]](#footnote-9) Die Konsistenz ist nach einer Abfrage nicht mehr in einem „solid state“, sondern in einem „soft state“ und soll nicht direkt nach Ende einer Transaktion erreicht werden, sondern nach Ende einer Operation.[[9]](#footnote-10) Der Fokus der BASE-Prinzipien besteht in der permanenten Verfügbarkeit,[[10]](#footnote-11) und nicht in der stetigen Konsistenz, welche nach dem ACID-Prinzip bestehen soll.

Mit Blick auf NoSQL Datenbanken sollte die Datenbank zwischen ACID und BASE bestehen.[[11]](#footnote-12)

* + 1. CAP-Theorem

Das CAP-Theorem wurde zum ersten Mal von Eric Brewer im Jahr 2000 vorgestellt.[[12]](#footnote-13) Dabei steht das Theorem aus drei verschiedenen Faktoren: Konsistenz (Consistency), Verfügbarkeit (Availability) und Ausfalltoleranz (Partition Tolerance).[[13]](#footnote-14)

Diese Prinzipien sagen Folgendes aus:[[14]](#footnote-15)

1. Die verfügbaren Daten sind auf allen Maschinen gleich und Updates sollen frequent durchgeführt werden
2. Die Daten müssen permanent verfügbar, und ständig zugänglich sein
3. Bei Fehlern der Maschine sollte die Datenbank weiterhin laufen, ohne die Arbeit stoppen zu müssen.



Abb. 1: CAP-Theorem[[15]](#footnote-16)

Das Theorem besagt, dass nur zwei von den drei Aspekten in einem verteilten System ermöglicht werden können. Demnach muss sich im Voraus also für zwei Aspekte entschieden werden.

* 1. Entwicklungen in der Datenbankbranche

Relationale Datenbanken wurden in den 70er Jahren von E.F. Codd[[16]](#footnote-17) mit dem Ziel entwickelt, strukturierte Daten in einer organisierten Art und Weise in Tabellen zu speichern,[[17]](#footnote-18) und die hierarchischen oder netzwerkartigen Datenbanken als Vorläufer der relationalen Datenbanken abzulösen.[[18]](#footnote-19)

Die hauptsächliche Verantwortlichkeit besteht bei relationalen Datenbanken zurzeit in der Datenverarbeitung, besonders in Verbindung mit der Speicherung von finanziellen Informationen und persönlichen Daten.[[19]](#footnote-20) Beschränkungen bestehen jedoch besonders in dem starken Datenwachstum, durch welchen Datenabfragen durch das große Volumen der Daten nicht mehr so effizient gestaltet werden kann. Außerdem zeichnen sich Schwierigkeiten in der Speicherung und dem Management von größeren Datenbanken ab.[[20]](#footnote-21) Viele Unternehmen setzen aus diesen Gründen nicht-relationale Datenbanken, auch unter dem Namen NoSQL Datenbanken, in ihren Betrieben ein.[[21]](#footnote-22)

NoSQL wurde zuerst in den 1998er Jahren von Carlo Strozzi genannt, wuchs aber erst um 2009 zu einem Konkurrenten der relationalen Datenbanken.[[22]](#footnote-23) Besonders mit dem Aufkommen von des Internets und der Vielzahl von webbasierten Anwendungen gewannen die nicht-relationalen Konzepte gegenüber den relationalen an Gewicht.[[23]](#footnote-24)

* 1. Datenbankmodelle
     1. Relationale Datenbanken

Relationale Datenbanken werden primär durch die ACID-Prinzipien repräsentiert. Daraus schließend sollten relationale Datenbanken besonders dann gewählt werden, wenn Konsistenz von hoher Relevanz ist.[[24]](#footnote-25)

[[25]](#footnote-26)Vorteile bestehen besonders in der langzeitigen Nutzung dieses Modells. Als Industriestandard können relationale Datenbanken schon viel Erfahrung abzeichnen. Die Dokumentation von Professionals, welche in ihrer Karriere die gleichen Paradigmen genutzt haben, stellt weiterhin nützliche Hilfestellung bei der Entwicklung von Datenbanken dar. Ein weiterer Vorteil von relationalen Datenbanken ist die gegebene Reife. Global Player wie International Business Maschine (IBM) und Microsoft, sowie die Open-Source Community bieten stetige Verbesserungen zu dem bestehenden Datenbankmodell und vereinfacht somit die Nutzung. Auch die hohen Levels an Support und die Möglichkeit komplexe Joins zu kreieren, damit Datenbank Administratoren tiefe und komplexe Abfragen implementieren können, stellen durchaus positive Aspekte der relationalen Datenbanken dar.

Negative Aspekte bestehen besonders in der Pflicht, Daten vor der Speicherung definieren zu müssen.[[26]](#footnote-27) Weiterhin unterstützen relationale Datenbanken nicht die hohe Skalierbarkeit, soweit nicht eine bessere Hardware zu diesem Zweck implementiert werden kann.[[27]](#footnote-28) Ein weiterer Defizit besteht in der Abfragesprache in Form von SQL, welche bei strukturierten Daten zwar effizient ist, bei unstrukturierten Daten jedoch höchst komplex wird.[[28]](#footnote-29)

* + 1. NoSQL Datenbanken

NoSQL Datenbanken existieren in dem Spektrum zwischen den ACID und BASE-Prinzipien[[29]](#footnote-30) und werden primär durch die fehlenden Relationen zwischen den verschiedenen Datensätzen charakterisiert.[[30]](#footnote-31) NoSQL Implementation baiseren auf Entitäten und unterstützen die Funktionen von relationalen Datenbankmanagementsystemen wie das Sortieren, Indexen, Projektieren und Abfragen von Daten. Zusätzlich werden Joins und ACID Garantien für eine höhere Transaktionsgeschwindigkeit eingetauscht.[[31]](#footnote-32) Popularität für diesen alternativen Ansatz wurde durch die einfache Skalierbarkeit, sowie einer verbesserten Sicherheit gewonnen.[[32]](#footnote-33)

Meier und Kaufmann erläutern in ihrem Buch „SQL- & NoSQL-Datenbanken“, dass folgende Bedingungen für eine NoSQL Datenbank vorliegen müssen[[33]](#footnote-34):

1. Das zugrundeliegende Datenbankmodell ist nicht relational
2. Das Datenbanksystem erfüllt die Anforderungen für umfangreiche Datenvolumen, flexible Datenstrukturen und Echtzeitverarbeitung, auch bekannt unter den drei V’s (Volume, Variety und Velocity)
3. Das Datenbankschema unterliegt keinem fixen Schema
4. Die Datenbankarchitektur unterstützt horizontale Skalierbarkeit und verteilte Webanwendungen
5. Das Datenbanksystem unterstützt Datenreplikation
6. Durch das CAP-Theorem ist Konsistenz nur verzögert gewährleistet, falls hohe Verfügbarkeit und Ausfalltoleranz gewährleistet angestrebt wird

Generell kann man bei NoSQL Datenbanken zwischen vier verschiedenen Arten differenzieren:

* Key-Value Store
* Document Store
* Column Store
* Graphendatenbank

Bei dem Key-Value Store werden alle gelagerten Daten durch ein „key-value pair“ je Datei repräsentiert. Damit ist jeder Schlüssel einzigartig und erlaubt es die Datensatzinformationen als Werte anzufordern. Diese Struktur ist auch unter dem Namen „Hash Table“ bekannt, in dem Datenabrufe mit Hilfe des Schlüssels durchgeführt werden.[[34]](#footnote-35)

Die Document Store Datenbanken sind dafür ausgelegt, die gespeicherten Daten, welche in Dokumenten verschiedener Formate, wie beispielsweise XML oder JSON, gespeichert werden, zu handhaben. Diese Form der Datenbank ist komplexer als der Key-Value Store.[[35]](#footnote-36)

Der Column Store besitzt eine Struktur, welche ähnlich zu der der relationalen Datenbank ist. Hier werden alle Daten in Reihen und Spalten gespeichert. Spalten, welche ähnliche Daten besitzen und vermehrt zusammen abgerufen werden, können zudem gruppiert werden.[[36]](#footnote-37)

Graphendatenbanken werden genutzt, wenn gespeicherte Daten mit verflochtenen Elementen wie Social Networking oder Road Maps, als Graph dargestellt werden sollen.[[37]](#footnote-38)

[[38]](#footnote-39)Vorteile der NoSQL Datenbanklösungen bestehen zum einem in einem einfachen Applikationsmanagement, welches die Notwendung von Applikationsänderungen oder Datenbankschema-Änderungen nimmt. Weiterhin bieten NoSQL Datenbanken die Möglichkeit, Clusters und Nodes transparent zu nutzen, ohne dass das Datenbankadministrationsmanagement oder eine manuelle Distribution von Informationen benötigt wird. Zudem sind NoSQL Datenbanken dafür ausgelegt, Daten automatisch zu administrieren und wiederherzustellen.

Wird die Literatur zu NoSQL Datenbanken betrachtet, so wird als Vorteil primär die Skalierbarkeit in den Vordergrund gezogen. Die horizontale Skalierbarkeit erlaubt es zum einen Anfragen durch niedrig-preisliche Server zufriedenzustellen, wodurch ein Unternehmen seine Kosten verringern kann.[[39]](#footnote-40) Außerdem können einzelne Dokumente oder „key-value pairs“ zwischen verschiedenen Servern verschoben werden, ohne dass die Datenbank als Ganzes an Integrität verliert. Dieser Sachverhalt vereinfacht die horizontale Skalierbarkeit und Datenredundanz.[[40]](#footnote-41)

Nachteilig zu sehen ist die unzuverlässige Natur der Datenverfügbarkeit.[[41]](#footnote-42) Die letztendliche Konsistenz bedeutet, dass zwei Personen mit verschiedenen Kopien des gleichen Dokuments eine aktualisierte und nicht-aktualisierte Version eines Dokuments sehen können, wenn kurz zuvor eine write-Operation durchgeführt worden ist.[[42]](#footnote-43) Eine weitere Schwäche der NoSQL Datenbanken besteht in der Datenredundanz und der Speicherung unstrukturierter Daten, welche eine finanzielle Belastung für das Unternehmen darstellen kann.[[43]](#footnote-44) Weiterhin ist die Technologie der NoSQL-Datenbanken noch sehr jung und dadurch teilweise fehlerhaft.[[44]](#footnote-45) Eines der größten Schwachstellen ist jedoch das CAP-Theorem, bei welchem zwischen zwei der drei Eigenschaften gewählt werden muss.[[45]](#footnote-46)

* + 1. Zusammenfassung

Mahamed, Altrafi und Ismail fassen in ihrem Zeitschriftenartikel die Unterschiede von relationalen Datenbanken und noSQL Datenbanken nach verschiedenen Kriterien zusammen. In Form einer Tabelle können diese wie folgt dargestellt werden:

|  |  |  |
| --- | --- | --- |
|  | **Relationale Datenbanken** | **NoSQL Datenbanken** |
| **Transaktionsausfallsicherheit** | Garantiert sehr hohe Transaktionsausfallsicherheit | Spanne von BASE und ACID |
| **Datenmodell** | Daten als Tupel repräsentiert und durch Relationen gruppiert  Die Relationen beinhaltet Tupel (Reihen), mit Attributen (Spalten) Attribut kann durch Domain (Menge von Werten) angesprochen werden | Nehmen Modelliertechniken wie Key-Value Stores, Graphen- oder Dokumenten Stores.  NoSQL Datenbanken kann aus einem oder mehreren Datenmodellen bestehen |
| **Skalierbarkeit** | vertikal | horizontal |
| **Cloud** | Nicht gut für den Cloud-Betrieb geeignet, da sie nicht wichtige Funktionen nicht unterstützen und die nur bis zu einem bestimmten Limit skaliert werden kann | Beste Lösung für Cloud Datenbanken, da alle Charakteristiken, welche NoSQL definieren, in der Cloud Umgebung gewünscht werden |
| **Big Data Handling** | Lösung ist Skalierbarkeit und Datenverteilung in Form von horizontaler oder vertikaler Skalierbarkeit – nur begrenzt möglich | Entworfen um Big Data zu handhaben – vorhandene Methoden erleichtern und verbessern die Performance vom Speichern und Auslesen der Daten |
| **Data Warehouse** | Über Zeit nimmt die Anzahl der gespeicherten Daten stark zu, wodurch ein Big Data Problem entsteht – führt zu Performance Abbau | Design fokussiert sich auf hohe Performance, Skalierbarkeit, Verfügbarkeit und das Speichern von Big Data – dadurch eine gute Lösung für Data Warehouses |
| **Komplexität** | Erhöht sich, da Nutzer Daten in konvertieren müssen, damit sie in die Tabellen passen | Haben die Fähigkeit unstrukturierte und semi-strukturierte Daten zu speichern |
| **Crash Recovery** | Crash Recovery durch den Recovery Manager, welcher dafür verantwortlich ist sicherzustellen, dass Transaktionen atomar und dauerhaft sind [durch log-files und ARIES algorithmus] | Benötigt Replikationen als Backup, um von einem Crash zu recovern |
| **Sicherheit** | Hat sehr sichere Mechanismen um Security Services bereitzustellen – trotzdem gibt es Sicherheitsrisiken wie SQL Injections, Cross Site Scripting, Root Kits etc. | Wurde eher für die Verbesserung von Performance und Big Data auf den Markt gebracht, nicht um Sicherheitslücken zu füllen |

Tabelle 1: Relationale vs. NoSQL Datenbanken[[46]](#footnote-47)

1. DynamoDB

Das zweite theoretisch ausgelegte Kapitel beschäftigt sich mit der Einführung in die *Amazon Web Services* und stellt im weiteren Verlauf die DynamoDB im Detail vor. Der Einstieg widmet sich den Hintergründen der *Amazon Web Services*, um ein grobes Verständnis für die Infrastruktur zu erzeugen. Anschließend sollen grundlegende Informationen zu DynamoDB hinterlegt werden. In Folge dessen gilt es, die Art der Datenverwaltung und das Datenmodell der NoSQL-Datenbank zu erläutern und zu untersuchen. Abschließend soll ein Vergleich mit alternativen NoSQL-Lösungen die Einschränkungen bzw. die besonderen Merkmale von DynamoDB ausarbeiten.

* 1. Generelle Informationen über AWS

Die DynamoDB ist ein Teil der *Amazon Web Services* und daher ist es relevant, die Merkmale und Besonderheiten der Infrastruktur zu erläutern, bevor eine Einführung in Datenbanklösung gleistet werden kann. Das folgende Unterkapitel ist auf die Einführung in AWS ausgerichtet. Im Verlauf dieses Unterkapitels werden die Ziele und Absichten der Dienstleistung aufgezeigt. Im Anschluss folgt eine Gegenüberstellung der Vorteile und Nachteile der Infrastruktur.

* + 1. Infrastructre as a Serivice (IaaS)/Kunden

Das Modell von *Infrastructure as a Service* stellt laut einem Artikel von Wolfgang Sommergut „die IT-Ressourcen, wie Rechenleistung, [Speicherplatz] oder Netzwerkkapazitäten zur Verfügung“[[47]](#footnote-48). Außerdem habe der Anwender die Chance, das Betriebssystem, sowie die Applikationen eigenständig zu steuern und die Infrastruktur nach eigenem Bedarf zusammenzustellen. Im Wesentlichen handelt es sich bei AWS also um ein *IaaS,* die die Möglichkeiten zur flexiblen Datenspeicherung und den Ausbau der verfügbaren Rechenleistung bereitstellt. Neben den genannten Ausprägungen, bietet AWS eine Vielfalt von Ressourcen an, die in Tabell 1 zusammengefast sind.

|  |  |
| --- | --- |
| **IT-Ressource** | **Beschreibung** |
| Online-Speicher | Es handelt sich hierbei um Cloud-Speicher. |
| Near-line Speicher |  |
| Server |  |
| Nachrichtendienste |  |
| Überwachung |  |
| Datenbanksystem |  |
|  |  |
|  |  |

Tabelle 2 IT-Ressourcen in AWS[[48]](#footnote-49)

* + 1. Gründe für die Verwendung von AWS/Entwicklungsgeschichte (evtl.)
    2. Ziele
    3. Vor- und Nachteile von AWS
  1. Was ist DynamoDB
     1. Database as a Service (DBaaS) Definition
     2. Geschichte der DynamoDB
     3. Verwendung der DynamoDB in der Amazon Infrastruktur
     4. Preismodell
  2. Art der Datenverwaltung/Datenmodell
     1. Tabelle, Elemente und Attribute Konzept
     2. Eigenschaften (Fully managed, Durable, Scalable, Fast, Simple Administration, Flexible, Fault Tolerance, Indexing, Secure, Cost Effective)
     3. Datenspeicherung über die SSD
     4. Key-Value Eigenschaft
     5. Datentypen
  3. Vergleich mit anderes NoSQL-Lösungen
     1. Microsoft Azure
     2. Redis
     3. Vorteile und Nachteile von DynamoDB

1. Praxis

Im Praxisteil werden folgende Themen behandelt!!!!!!!

* 1. Insgesamt zum Zurechtfinden
  2. Testdatenbank mit Testdaten
  3. Datenbanksprache
  4. GUI/API
  5. Ablageform der Daten

1. Schluss
   1. Fazit
   2. Ausblick

Anhang (bei Bedarf)

# Quellenverzeichnis

**Literaturverzeichnis**

**Preiß, N. (2007):** Entwurf und Verarbeitung relationaler Datenbanken, München/Wien: Oldenbourg

**Steger, J. (2006):** Kosten- und Leistungsrechnung, 4. Aufl., München/Wien: Oldenbourg

**Stoi, R. (2003):** Management und Controlling von Intangibles, in: Studium & Praxis, 4. Jg., Nr. 1, S. 34-46

**Verzeichnis der Internet- und Intranetquellen (bei Bedarf)**

**Gesprächsverzeichnis (bei Bedarf)**

# Ehrenwörtliche Erklärung

Wir versichern hiermit, dass wir unsere Seminararbeit mit dem Thema: [ Thema einfügen ] selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Wir versichern zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ort, Datum) (Kerstin Farke)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ort, Datum) (Dhruv Mahandru)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ort, Datum) (Henning Mohr)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ort, Datum) (Käthe Vrettos)

Literaturverzeichnis

Abramova, Veronika; Bernardino, Jorge; Furtado, Pedro (2014): Experimental Evaluation of NoSQL Databases. In: *International Journal of Database Management Systems (IJDMS)* 6 (3), S. 1–16. Online verfügbar unter http://s3.amazonaws.com/academia.edu.documents/44185663/6314ijdms01.pdf?AWSAccessKeyId=AKIAJ56TQJRTWSMTNPEA&Expires=1480620894&Signature=Xuo9y8raSPFKC7sf2q3deopev5o%3D&response-content-disposition=inline%3B%20filename%3DEXPERIMENTAL\_EVALUATION\_OF\_NOSQL\_DATABAS.pdf.

Edward, S. G.; Sabharwal, N. (2015): Practical MongoDB: Architecting, Developing, and Administering MongoDB: Apress. Online verfügbar unter https://books.google.de/books?id=7iI3CwAAQBAJ.

Hammes, Dayne; Medero, Hiram; Mitchell, Harrison (2014): Comparison of NoSQL and SQL Databases in the Cloud. In:. Information Systems Conference. Macon, GA, USA, March 21st - 22nd 2014. Proceedings of the Southern Association for Information Systems Conference. Macon, GA, USA, S. 1–8.

Jatan, Nishtha; Puri, Sahil; Ahuja, Mehak; Kathuria, Ishita; Gosain, Dishant (2012): A Survey and Comparison of Relational and Non-Relational Database. In: *International Journal of Engineering Research & Technology (IJERT)* 1 (6), S. 1–5. Online verfügbar unter http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.678.9352&rep=rep1&type=pdf.

Mahamed, Mohamed A.; Altrafi, Obay G.; Ismail, Mohammed O. (2014): Relational vs. NoSQL Databases: A Survey. In: *International Journal of Computer and Information Technology* 3 (3), S. 598–601.

Meier, Andreas; Kaufmann, Michael (2016): SQL- & NoSQL-Datenbanken. 8., überarb. u. erw. Aufl. 2016. Berlin, Heidelberg: Springer Berlin Heidelberg (eXamen.press).

Schicker, Edwin (2014): Datenbanken und SQL. Eine praxisorientierte Einführung mit Anwendungen in Oracle, SQL Server und MySQL. 4., überarb. Aufl. 2014. Wiesbaden: Springer Vieweg (SpringerLink : Bücher).

Sharma, Vatika; Dave, Meenu (2012): SQL and NoSQL Databases. In: *International Journal of Advanced Research in Computer Science and Software Engineering* 2 (8), S. 20–27. Online verfügbar unter http://s3.amazonaws.com/academia.edu.documents/33559632/V2I800154.pdf?AWSAccessKeyId=AKIAJ56TQJRTWSMTNPEA&Expires=1480620329&Signature=IgyAiXJM9JNlTsXPAjBJK3dQTUQ%3D&response-content-disposition=inline%3B%20filename%3DSQL\_and\_NoSQL\_Databases.pdf.

1. Schicker 2014, S. 18 [↑](#footnote-ref-2)
2. Sharma und Dave 2012, S. 21 [↑](#footnote-ref-3)
3. Mahamed et al. 2014, S. 598 [↑](#footnote-ref-4)
4. Schicker 2014, S. 18 [↑](#footnote-ref-5)
5. Mahamed et al. 2014, S. 598 [↑](#footnote-ref-6)
6. Mahamed et al. 2014, S. 598 [↑](#footnote-ref-7)
7. Mahamed et al. 2014, S. 598 [↑](#footnote-ref-8)
8. Sharma und Dave 2012, S. 21 [↑](#footnote-ref-9)
9. Mahamed et al. 2014, S. 598f. [↑](#footnote-ref-10)
10. Sharma, Dave 2012, S. 21, vgl. dazu auch Mahamed et. al. 2014, S.598f. [↑](#footnote-ref-11)
11. Mahamed et al. 2014, S. 598–599, vgl. dazu auch Sharma, Dave 2012, S.21 [↑](#footnote-ref-12)
12. Mahamed et al. 2014, S. 599 [↑](#footnote-ref-13)
13. Mahamed et al. 2014, S. 599 [↑](#footnote-ref-14)
14. Vgl. auch im Folgenden Sharma und Dave 2012, S. 21–22 [↑](#footnote-ref-15)
15. Edward und Sabharwal 2015, S. 16 [↑](#footnote-ref-16)
16. Jatan et al. 2012, S. 2 [↑](#footnote-ref-17)
17. Abramova et al. 2014, S. 1 [↑](#footnote-ref-18)
18. Meier und Kaufmann 2016, S. 18 [↑](#footnote-ref-19)
19. Mahamed et al. 2014, S. 598 [↑](#footnote-ref-20)
20. Abramova et al. 2014, S. 1 [↑](#footnote-ref-21)
21. Mahamed et al. 2014, S. 598 [↑](#footnote-ref-22)
22. Mahamed et al. 2014, S. 598 [↑](#footnote-ref-23)
23. Meier und Kaufmann 2016, S. 18 [↑](#footnote-ref-24)
24. Abramova et al. 2014, S. 1–2 [↑](#footnote-ref-25)
25. Hammes et al. 2014, S. 1 [↑](#footnote-ref-26)
26. Abramova et al. 2014, S. 1–2 [↑](#footnote-ref-27)
27. Jatan et al. 2012, S. 3 [↑](#footnote-ref-28)
28. Jatan et al. 2012, S. 3 [↑](#footnote-ref-29)
29. Mahamed et al. 2014, S. 599 [↑](#footnote-ref-30)
30. Abramova et al. 2014, S. 1–2 [↑](#footnote-ref-31)
31. Hammes et al. 2014, S. 1 [↑](#footnote-ref-32)
32. Hammes et al. 2014, S. 1 [↑](#footnote-ref-33)
33. Meier und Kaufmann 2016, S. 20 [↑](#footnote-ref-34)
34. Abramova et al. 2014, S. 3–4 [↑](#footnote-ref-35)
35. Abramova et al. 2014, S. 3–4 [↑](#footnote-ref-36)
36. Abramova et al. 2014, S. 3–4 [↑](#footnote-ref-37)
37. Abramova et al. 2014, S. 3–4 [↑](#footnote-ref-38)
38. Abramova et al. 2014, S. 3–4 [↑](#footnote-ref-39)
39. Abramova et al. 2014, S. 3–4 [↑](#footnote-ref-40)
40. Hammes et al. 2014, S. 3 [↑](#footnote-ref-41)
41. Jatan et al. 2012, S. 4–5 [↑](#footnote-ref-42)
42. Hammes et al. 2014, S. 3–4 [↑](#footnote-ref-43)
43. Hammes et al. 2014, S. 3–4 [↑](#footnote-ref-44)
44. Hammes et al. 2014, S. 3–4 [↑](#footnote-ref-45)
45. Hammes et al. 2014, S. 3–4 [↑](#footnote-ref-46)
46. Mahamed et al. 2014, S. 600 [↑](#footnote-ref-47)
47. Artikel aus Computerwochen von Wolfagang Sommergut

    http://www.computerwoche.de/a/was-sie-ueber-die-cloud-wissen-muessen,2504589,2 [↑](#footnote-ref-48)
48. # Entenommen aus Amazon Web Services: Migrating Your .NET Enterprise Application

    von Rob Linton, Seite 8 [↑](#footnote-ref-49)