Big Data in Datenbanken

Konzepterstellung für das Performancemonitoring von SQL und NoSQL Datenbanken

B a c h e l o r a r b e i t

an der Hochschule für Angewandte Wissenschaften Hof
Fakultät Informatik
Studiengang Wirtschaftsinformatik

Vorgelegt bei
Prof. Dr. Heym

Vorgelegt von
Andrea Reisenauer

Hof, 14.03.2019

Vorwort

Die vorliegende Praxisarbeit entstand im Rahmen meines Studiums im Bachelorstudiengang Wirtschaftsinformatik an der Hochschule für angewandte Wissenschaften in Hof.

Mein besonderer Dank gilt Herrn Prof. Dr. Heym für die Gelegenheit zur Anfertigung dieser Praxisarbeit sowie für seine umfassende Betreuung. Vielen Dank, dass Sie mir stets mit Rat und Tat zur Seite standen und sich auf meine besondere Situation des vorangegangenen Verwaltungsinformatikstudiums eingelassen haben.

Darüber hinaus bedanke ich mich bei Herrn Hettmer vom Bayerischen Landesamt für Digitalisierung, Breitband und Vermessung für die zur Verfügung gestellten Geodaten und die interessanten Einblicke in die Vermessungsverwaltung.

Ebenso möchte ich meiner Familie und meinem Freund Max danken, die mich mit viel Geduld und Nachsicht durch die Zeit der Bachelorarbeit begleitet haben.

Inhaltsverzeichnis

Vorwort II

Inhaltsverzeichnis III

Darstellungsverzeichnis V

Tabellenverzeichnis VI

Abkürzungsverzeichnis VII

1 Einleitung 1

1.1 Motivation 1

1.2 Fragestellung 2

1.3 Aufbau der Arbeit 2

2 Relationale Datenbankmanagementsysteme (RDBMS) 4

3 Nicht-Relationale Datenbankmanagementsysteme (NRDBMS) 7

3.1 Historie - von SQL zu NoSQL 7

3.2 Grundlegende Techniken und Konzepte 8

3.2.1 CAP-Theorem und BASE 8

3.2.2 Map/Reduce 10

3.2.3 REST 11

3.3 Kategorisierung von NoSQL Datenbanken 12

3.3.1 Key-Value Stores 13

3.3.2 Wide Column Stores/Column Families 15

3.3.3 Document Store 17

3.3.4 Graph Databases 18

4 Anwendungspotential von Datenbanken für die Speicherung von Geodaten 20

4.1 Definition Geodaten 20

4.2 Gauß-Krüger-Koordinatensystem 20

4.3 Geodaten in RDBMS 22

4.3.1 Erweiterungen für Geodaten 22

4.3.2 Datenformate 22

4.4 GeoDaten in NRDBMS 23

4.5 Entscheidungsgrundlage für MongoDB 25

5 Entwurf des Performancemonitorings von MySQL und MongoDB 27

5.1 technische Voraussetzungen 27

5.2 Erläuterungen zu den Verwendeten Daten 28

5.3 Performancemonitoring 29

5.3.1 Vorbereitungen 29

5.3.2 Ablauf des Performancemonitorings 30

6 Zusammenfassung und Ausblick 34

7 Literaturverzeichnis 35

8 Eidesstattliche Erklärung 39

Darstellungsverzeichnis

Abbildung 1: schematische Darstellung des Datenbankzugriffs 4
Abbildung 2: Beispiel für Daten einer Datenbank 5
Abbildung 3: Das CAP-Theorem 9
Abbildung 4: Ablauf des Map/ Reduce-Verfahrens 11
Abbildung 5: schematische Darstellung der Datenbankpartitionierung 15
Abbildung 6: Darstellung der zeilen- und spaltenorientierten Speicherung 16
Abbildung 7: Beispiel einer Dokument-Datenbank 17
Abbildung 8: Graph mit vier Knoten und ungerichteten, ungewichteten Kanten 18
Abbildung 9: Graphdatenbank 19
Abbildung 10: Raster im Gauß-Krüger Koordinatensystem 21
Abbildung 11: Typ-1-Hypervisor 28
Abbildung 12: Typ-2-Hypervisor 28
Abbildung 13: Programmablaufplan 33

Tabellenverzeichnis

Tabelle 1: Tabellarische Darstellung einer Schlüssel-Wert Datenbank 14
Tabelle 3: Daten im WKT-Format...23
Tabelle 4: Daten im GeoJSON-Format 24
Tabelle 2: Vergleich SQL und MongoDB-Abfrage 26
Tabelle 5: Gegenüberstellung der MySQL und Oracle Datentypen 30
Tabelle 6: schematische Darstellung des Wörterbuchs 32

Abkürzungsverzeichnis

BSON	*Binary JSON*
DBMS	*Datenbankmanagementsystem*
HTTP	*Hypertext Transfer Protocol*
JDBC	*Java Database Connectivity*
JSON	*JavaScript Object Notation*
KB	*Kilobyte*
LDBV	*Landesamt für Digitalisierung, Breitband und Vermessung*
MB	*Megabyte*
NRDBMS	*nicht relationales Datenbankmanagementsystem*
OGC	*Open Geospatial Consortium*
RDBMS	*Relationales Datenbankmanagementsystem*
REST	*Representational State Transfer*
SRID	*Spatial Reference Identifier*
UCS	*Universal Character Set*
UTF-8	*8-Bit UCS Transformation Format*
VM	*Virtuelle Maschine*
VMs	*Virtuelle Maschinen*
WKT	*Well Known Text*
XML	*Extensible Markup Language*

1 Einleitung

Im 19. Jahrhundert galt die Grundsteuer als wichtigste Einnahmequelle des Staates. Um eine gerechte und einheitliche Besteuerung der Bürger in Bayern sicherzustellen, gründete König Max I. im Jahr 1808 die Steuervermessungskommission mit dem Ziel der Vermessung des Grundeigentums in Bayern. Die Vermessungsverwaltung gibt es somit bereits seit über 200 Jahren. Die Technik zur Vermessung sowie die Dokumentation der Messergebnisse haben sich im Laufe der Zeit jedoch stark verändert. Mit Satellitentechnik und computergesteuerter Hochtechnologie kann die Bayerische Vermessungsverwaltung Daten über praktisch jeden Quadratmeter Bayerns zur Verfügung stellen, über das Internet sogar innerhalb von Minuten. Die Abbildung der Erdoberfläche in Form von Karten, Luftbildern oder 3D-Modellen und die Übersicht zu den Besitz- und Eigentumsverhältnissen, allesamt Geodaten, gelten als unverzichtbare Grundlage für viele gesellschaftliche und politische Entscheidungen, z. B. Bauplanungen, Umweltschutz und Standortentscheidungen. (vgl. Bayerisches Staatsministerium der Finanzen und für Heimat)

1.1 Motivation

Bisher verwendet man in der Vermessungsverwaltung relationale Datenbankmodelle zur Speicherung der Daten. Die Menge der zu speichernde Geodaten steigt jedoch stetig an. Sind in einer Datenbank große Datenmengen gespeichert, führt dies häufig zu Wartezeiten bei einer Datenbankabfrage. Um diese Wartezeiten zu vermeiden und eine größere Menge an Daten in einer Datenbank speichern zu können, entstanden in den letzten Jahren neue Datenbanken, die nicht auf dem relationalen Datenbankmodell nach Edgar F. Codd basieren. Um weiterhin auf dem aktuellsten Stand der Technik zu sein und die Performanz der Datenbankabfragen zu verbessern, zieht das Landesamt für Digitalisierung, Breitband und Vermessung (LDBV) einen Technologiewechsel in Erwägung.

Ziel dieser Bachelorarbeit ist es ein Konzept für das Performancemonitoring von relationalen Datenbankmanagementsystemen (RDBMS) und nicht relationalen Datenbankmanagementsystemen (NRDBMS) zu entwerfen. Dieses Konzept soll schließlich in der Praxisarbeit realisiert werden und dem LDBV als Entscheidungsgrundlage für das weitere Vorgehen dienen.

1.2 Fragestellung

In der vorliegenden Bachelorarbeit sollen die notwendigen Voraussetzungen für einen Performancevergleich von einem RDBMS mit NRDBMS analysiert werden. Nicht relationale Datenbankmanagementsysteme werden häufig auch als NoSQL-Datenbanken bezeichnet. NoSQL steht dabei für „Not only SQL“ und bedeutet, dass nicht nur SQL unterstützt wird, sondern auch andere Abfragemethoden wie JSON oder Skriptsprachen (vgl. Matzer und Litzel 2018). Im Anschluss an die Analyse soll ein Konzept für das Vorgehen bei der Performancemessung entworfen werden. Außerdem wird die Software, mit deren Hilfe das Performancemonitoring realisiert werden soll, spezifiziert.

Bei der Analyse und dem Entwurf soll die Beantwortung folgender Fragen im Mittelpunkt stehen:

- Wie unterscheidet sich die Datenhaltung in einer NoSQL Datenbank von der Datenhaltung in einer relationalen Datenbank?
- Müssen Daten, die bisher in einer relationalen Datenbank gespeichert wurden für eine NoSQL Datenbank verändert werden? Wenn ja, wie müssen die Daten verändert werden?
- Welche NoSQL Datenbank eignet sich zum Speichern von Geodaten?
- Welche technischen Voraussetzungen sind notwendig, um sinnvolle Messergebnisse zu erzielen?
- Wie soll das Java- Programm ablaufen mit dessen Hilfe die Performanceanalyse durchgeführt wird?

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in sechs Teile.

Zunächst werden im zweiten Kapitel kurz die Grundlagen relationaler Datenbankmodelle wiederholt und anschließend auf die Vor- und Nachteile dieser Datenbanken eingegangen.

Das dritte Kapitel beschäftigt sich mit NRDBMS. Zu Beginn wird dabei auf deren Entstehung eingegangen sowie eine mögliche Definition von NRDBMS vorgestellt. Im Anschluss daran werden die grundlegenden Techniken und Konzepte nicht relationaler Datenbanken aufgeführt. Das Kapitel endet mit der Kategorisierung der NRDBMS nach ihrer Datenhaltung.

Auf Kapitel zwei und drei aufbauend wird im vierten Kapitel auf das Anwendungspotential von Datenbanken für die Speicherung von Geodaten eingegangen. Dabei wird insbesondere die Datenhaltung von Geodaten in RDBMS und NRDBMS betrachtet.

Im Fokus des fünften Kapitels steht der Entwurf des Performancemonitorings von MySQL und MongoDB. Zunächst wird auf die technischen Voraussetzungen eingegangen, sodass sinnvolle Messergebnisse entstehen können. Nach einer kurzen Betrachtung der verwendeten Daten wird der konkrete Ablauf der Software zum Performancemonitoring vorgestellt.

Die Bachelorarbeit endet mit einer Zusammenfassung und einem Ausblick.

2 Relationale Datenbankmanagementsysteme (RDBMS)

Zur effizienten und übersichtlichen Verwaltung von Daten werden sogenannte Datenbanken verwendet. Datenbanken ermöglichen es den Benutzern gleichzeitig auf die gleichen Datensätze zuzugreifen und diese bei Bedarf zu verändern.

Definition Datenbank:
„Eine Datenbank ist eine Sammlung von Daten, die untereinander in einer logischen Beziehung stehen und von einem eigenen Datenbankverwaltungssystem (Database Management System, DBMS) verwaltet werden“ (Schicker 2014, S. 3).

Das Datenbankverwaltungssystem besitzt eine logische Schnittstelle nach außen. Jeder Benutzer und damit auch jedes Anwendungsprogramm greift ausschließlich über diese Schnittstelle auf die Daten zu. Diese logischen Datenzugriffe werden vom Datenbankverwaltungssystem in physische Zugriffe umgewandelt und so die Daten vom Speichermedium gelesen beziehungsweise auf das Speichermedium geschrieben.

Zu den wichtigsten Datenbankschnittstellen zählt SQL. Mit Hilfe dieser Programmiersprache kann der Endbenutzer auf die Datenbank zugreifen und die in der Datenbank gespeicherten Daten auslesen oder modifizieren (vgl. Schicker 2014, S. 4).

Der Zugriff auf die Daten einer Datenbank ist in Abbildung 1 dargestellt.

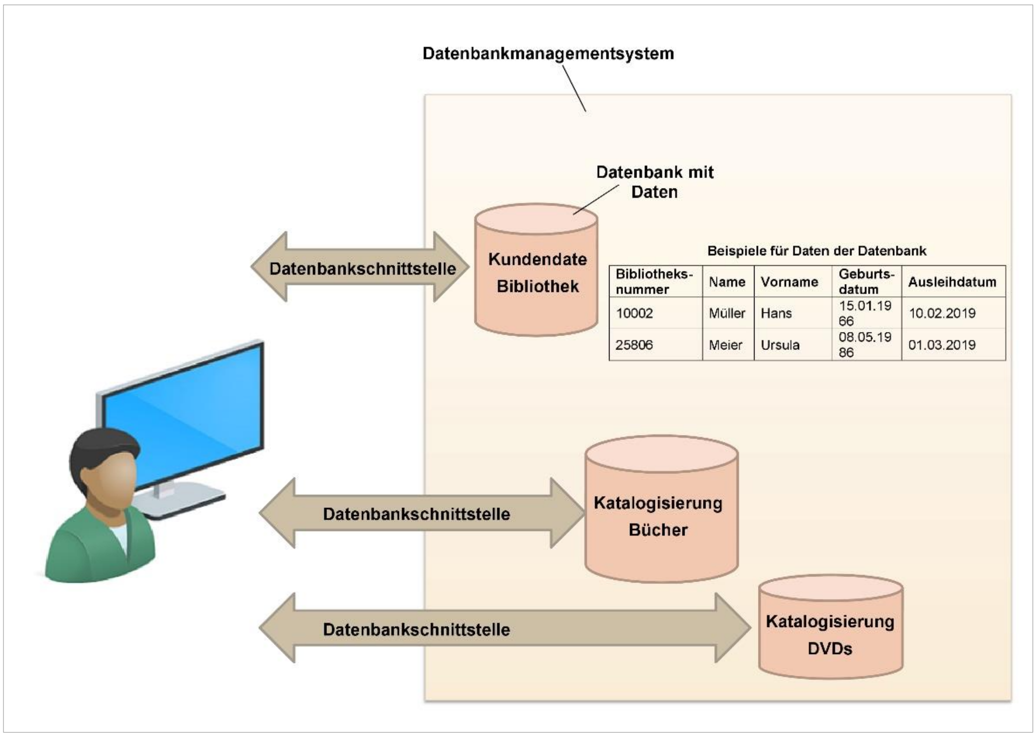


Abbildung 1: schematische Darstellung des Datenbankzugriffs | Quelle: eigene Darstellung

Bei einem Datensatz handelt es sich um eine inhaltlich zusammenhängende Gruppe von Daten einer Datenbank wie beispielsweise Autor, Titel und ISBN eines Buches. Jeder einzelne Eintrag eines Datensatzes ist ein Datenfeld.

Autor	Titel	ISBN	
Edwin Schicker	Datenbanken und SQL	978-3-8348-1732-7	Datensatz
Frank Lampe	Green IT	978-3-8348-0687-1	
Christian Baun	Betriebssysteme kompakt	978-3-662-53142-6	Datenfeld

Abbildung 2: Beispiel für Daten einer Datenbank | Quelle: eigene Darstellung

Eine relationale Datenbank besteht ausschließlich aus Tabellen, in denen die Daten gespeichert werden. Dabei muss vor dem Einfügen der Datensätze genau spezifiziert werden wie viele Datenfelder ein Datensatz enthalten darf und von welchem Datentyp die Daten eines Datenfelds sind. Darüber hinaus muss innerhalb einer relationalen Datenbank jeder Datensatz eindeutig identifizierbar sein. Die eindeutige Identifikation eines Datensatzes erfolgt über so genannte Primärschlüssel, die sich niemals verändern dürfen. Die Daten werden in den Datenbanktabellen konsistent und redundanzfrei abgelegt, weshalb die Daten jeweils nur einmal in der Datenbank gespeichert werden dürfen. Aus diesem Grund werden die Datensätze auf verschiedene Tabellen verteilt und untereinander verknüpft. Diese Zusammenhänge zwischen den einzelnen Tabellen werden über Beziehungen hergestellt, die in den Tabellen abgespeichert werden. Der Aufbau von Datenbeständen über Tabellen und ihre Beziehungen zueinander sind mathematisch fundiert. Da das Erstellen relationaler Datenbankmodelle und deren mathematische Grundlagen in der Vorlesung „Datenbanken I“ im 3. Semester intensiv behandelt wurde, wird an dieser Stelle nicht weiter auf die relationale Algebra eingegangen. Der Zugriff auf die Daten erfolgt ausschließlich über die Tabellen. Änderungen des logischen Datenbankaufbaus sind jederzeit möglich, da neue Tabellen recht einfach hinzugefügt oder gelöscht werden können. (vgl. Schicker 2014, S. 12)

Der Vorteil relationaler Datenbanken liegt in der guten Strukturierung der Daten und der Validierung der Daten beim Import. Darüber hinaus wurden die relationalen Datenbanken seit 1970 stetig weiterentwickelt und stellen ein ausgereiftes und zuverlässiges System zur Datenspeicherung dar. Hinzu kommt, dass relationale Datenbanken die standardisierte Abfragesprache SQL unterstützen, die es ermöglicht auch komplexe Datenbankabfragen zu realisieren.

Relationale Datenbanken haben jedoch auch Nachteile. Durch die Verteilung der Daten erfordern die Datenbankzugriffe „oft das Lesen und Zusammenfügen von Daten aus mehreren Tabellen, was die Laufzeit verlängert und zu vielen Ein- und Ausgaben führt“ (vgl. Schicker 2014, S. 13). Des Weiteren ist es in relationalen Datenbanken nicht möglich unstrukturierte Datenmengen zu speichern und performant abzufragen oder zu verarbeiten. „Das macht relationale Datenbankmanagementsysteme weniger geeignet für Anwendungen im Big-Data-Umfeld“ (vgl. Litzel und Tutanch 2017).

Die verbreitetsten relationalen Datenbanken sind Oracle, SQL Server von Microsoft, MySQL und PostgreSQL.

Zwar setzt das LDBV derzeit Datenbanken des Herstellers Oracle ein, dennoch wurde für das Performancemonitoring MySQL als Vertreter der RDBMS ausgewählt. Der Hauptgrund liegt darin, dass es sich bei MySQL um die populärste Open-Source-Datenbank der Welt handelt. Darüber hinaus unterstützt MySQL die Speicherung von Geodaten und ist plattformunabhängig, kann also auf jedem beliebigen Betriebssystem installiert werden. (vgl. MySQL: Warum MySQL?)

3 Nicht-Relationale Datenbankmanagementsysteme (NRDBMS)

3.1 Historie - von SQL zu NoSQL

Die Geschichte der NoSQL-Systeme begann deutlich früher als man vermuten könnte. Bereits im Jahr 1979 wurde von Ken Thompson eine Key/Hash-Datenbank namens DBM entwickelt. In den 80er Jahren entstanden dann die ersten noch heute populären Vorreiter der NoSQL-Systeme wie beispielsweise Lotus Notes, BerkeleyDB und GT.M. Der Begriff NoSQL tauchte zum ersten Mal im Jahr 1998 in Zusammenhang mit einer Datenbank von Carlo Strozzi auf. Diese Datenbank basierte zwar weiterhin auf dem relationalen Datenbankmodell, stellte aber keine SQL-API zur Verfügung. Für Strozzi selbst steht NoSQL für „noseequel", also „kein SQL" und er betont, dass sein NoSQL Konzept nichts mit der neuen NoSQL Bewegung zu tun hat. Die neue NoSQL Bewegung sollte seiner Meinung nach eher als „NoREL" bezeichnet werden, da sie nichts mit dem relationalen Datenbankmodell zu tun hat. (vgl. Strozzi 2010)

Der Aufstieg der NRDBMS begann in den 2000er-Jahren mit der Verbreitung des Internets und der damit einhergehenden Datenflut. Es bestand ein Bedarf an Datenhaltungssystemen, die mit den enorm großen Datenmengen von Web-Diensten, teilweise im Petabyte-Bereich oder größer, umgehen können. (vgl. Meier und Kaufmann 2016, S.221) Google mit seinen Ansätzen wie Map/Reduce und dem BigTable-Datenbanksystem kann man als den NoSQL-Vorreiter schlechthin bezeichnen (vgl. Edlich et al. 2011, S.1). BigTable nutzt eine Kombination aus Reihen- und Zeilenschlüsseln sowie Zeitstempeln, um Daten lose zu strukturieren und baut auf das hauseigene Dateisystem „Google File System" (GFS) (vgl. Chang et al. 2010). Andere große Suchmaschinen wie Yahoo oder der Onlinehändler Amazon und auch soziale Netzwerke wie Facebook und LinkedIn folgten ziemlich rasch Googles Vorbild (vgl. Edlich et al. 2011, S.1).

In der Zeit von 2006 bis 2009 entstanden viele der heute bekannten NoSQL-Systeme wie CouchDB, Cassandra, Redis, Riak und MonogDB. Der Begriff NoSQL im Sinne von „Not only SQL", so wie er heute verwendet wird, wurde Anfang 2009 von Johan Oskarsson für ein Treffen neu eingeführt. Dieses Treffen fand in San Francisco statt und beschäftigte sich mit verteilten strukturierten Datenspeichern. (vgl. Edlich et al. 2011, S.1).

Datenbanken, die vom klassischen RDBMS abweichen, gibt es also schon seit vielen Jahrzehnten, jedoch können sich NoSQL-Datenbanken erst seit 2009 auf dem Markt behaupten. In der vorliegenden Bachelor- und Praxisarbeit wird NoSQL-Datenbank als Synonym für NRDBMS verwendet.

Eine einheitliche Definition einer NoSQL Datenbank gibt es derzeit nicht. Der folgende Definitionsvorschlag stammt von Prof. Dr. Stefan Edlich und wurde auf seiner Internetseite http://nosql-database.org/ sowie in seinem Buch „NoSQL - Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken“ veröffentlicht:

„Unter NoSQL wird eine neue Generation von Datenbanksystemen verstanden, die meistens einige dieser Punkte berücksichtigt:

- Das zugrundeliegende Datenmodell ist nicht relational
- Die Systeme sind von Anbeginn an auf eine verteilte und horizontale Skalierbarkeit ausgerichtet
- Das NoSQL-System ist Open Source
- Das System ist schemafrei oder hat nur schwächere Schemarestriktionen
- Aufgrund der verteilten Architektur unterstützt das System eine einfache Datenreplikation
- Das System bietet eine einfache API
- Dem System liegt meistens auch ein anderes Konsistenzmodell zugrunde: Eventually Consistent und BASE, aber nicht ACID […].“ (Edlich et al. 2011, S.2)

3.2 Grundlegende Techniken und Konzepte

Bevor eine Kategorisierung der verschiedenen NRDBMS vorgenommen wird, soll im folgenden Kapitel auf die Techniken und Konzepte eingegangen werden, die häufig im NoSQL-Umfeld eingesetzt werden und die Teil der Definition nach Edlich et al. sind. Das Verständnis dieser Grundlagen ist wichtig, um den Einsatz von NoSQL Systemen zu verstehen. Einige der vorgestellten Konzepte sind bereits bekannt, da sie auch in RDBMS Anwendung finden.

3.2.1 CAP-Theorem und BASE

In Kapitel 2 ist beschrieben, dass eine der wichtigsten Vorteile von RDBMS die konsistente Datenhaltung ist. Diese Datenkonsistenz ist vor allem bei Systemen, die mit sensiblen Daten arbeiten, wie es beispielsweise in der Steuerverwaltung der Fall ist, unbedingt erforderlich. Bei großen Datenmengen stößt diese Technologie jedoch an ihre Grenzen, da eine vertikale Skalierung nicht mehr ausreicht. Unter vertikaler Skalierung versteht man eine Leistungssteigerung durch das Hinzufügen von Ressourcen, also zum Beispiel das Vergrößern des Speicherplatzes oder das Einbauen einer neuen CPU zu einem Rechner (vgl. Wikipedia 2018). Abhilfe für dieses Problem schafft häufig die horizontale Skalierung. Bei der horizontalen Skalierung erfolgt die Steigerung der Leistung eines Systems durch das Hinzufügen zusätzlicher Rechner in das Netzwerk (vgl. Wikipedia 2018). Die Gewährleistung der Konsistenz der Daten über mehrere verteilte Datenbankinstanzen hinweg ist bei RDBMS jedoch mit einem hohen Wartungs- und Entwicklungsaufwand verbunden und führt zu einer langen Antwortzeit der Datenbank.

Im Jahr 2000 hielt Eric Brewer den Vortrag „Towards Robust Distributed Systems“ über seine Forschung an verteilten Systemen und stellte im Rahmen dieses Vortrags das sogenannte CAP-Theorem vor. Er stellte die Vermutung auf, dass in verteilten Anwendungen die drei Ziele Konsistenz (Consistency), Verfügbarkeit (Availability) und Ausfalltoleranz (Partition Tolerance) nicht gleichzeitig realisiert werden können. Das CAP-Theorem sagt aus, dass in einem massiv verteilten Datenhaltungssystem maximal zwei der drei Ziele garantiert werden können und ist Abbildung 3 in dargestellt. (vgl. Brewer)

Im Jahr 2002 wurde die Theorie des CAP-Theorems durch Seth Gilbert und Nancy Lynch für verteilte Web Services bestätigt (vgl. Gilbert und Lynch 2002).

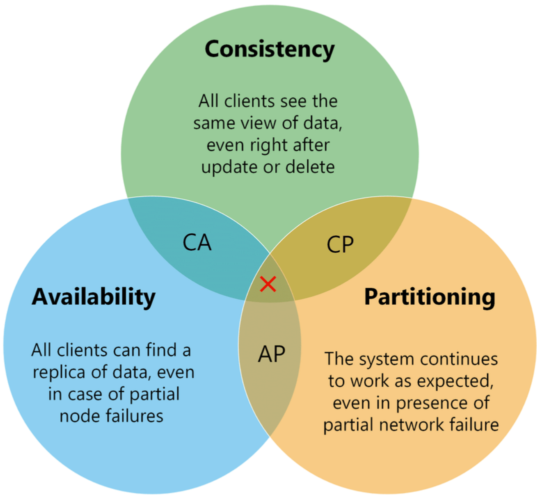


Abbildung 3: Das CAP-Theorem

Quelle: https://www.researchgate.net/figure/Visualization-of-CAP-theorem_fig2_282679529

Um eine Lösung für den aus dem CAP-Theorem resultierenden Konflikt zu finden, wurde das sogenannte BASE-Modell (Basically Available, Soft State, Eventually Consistent) entwickelt. Dieses Modell legt den Schwerpunkt auf die Verfügbarkeit und betrachtet die Konsistenz der Daten als einen Übergangsprozess. Es wird also bei NRDBMS im Gegensatz zu RDBMS erlaubt, dass zwischenzeitlich unterschiedliche Datenversionen vorhanden sind, die erst zeitlich verzögert aktualisiert werden. Systeme, die BASE verwenden, erreichen irgendwann auch den Status der Konsistenz, jedoch wird der konsistente Zu-stand der Daten nicht direkt nach Beendigung einer Transaktion gewährleistet. (vgl. Meier und Kaufmann 2016, S.148)

Wie lange es dauern kann bis die Daten konsistent sind, hängt unter anderem davon ab, wie viele replizierende Knoten das System enthält, wie schnell diese reagieren und wie hoch die Last des Systems ist. Als Softwareentwickler muss man also immer im Hinterkopf behalten, dass bei NoSQL-Datenbanken die Daten gegeben falls (noch) nicht konsistent sind. (vgl. Edlich et al. 2011, S.31)

3.2.2 Map/Reduce

Einer der Gründe warum Google, Amazon, Facebook und viele mehr auf NoSQL-Datenbanken setzten, liegt in der effizienten Verarbeitung sehr großer Datenmengen in einer Größenordnung von vielen Terabytes bis hin zu mehreren Petabytes. Beim Map/Reduce Verfahren handelt es sich um ein Datenverarbeitungsmodell zur nebenläufigen Berechnung großer Datenaufkommen in Computerclustern. Entwickelt wurde diese Technologie im Jahr 2004 von Google Inc. (vgl. Edlich et al. 2011, S.12)

Wie der Name erahnen lässt, besteht das Verarbeitungsmodell aus den zwei unterschiedlichen Phasen Map und Reduce, die nacheinander ausgeführt werden. In der ersten Phase wird die map()-Funktion auf jeden Datensatz der Datenbank angewendet. Sie erhält als Übergabeparameter ein Key-Value Paar bestehend aus einer eindeutigen ID (Key) und dem Datensatz (Value). Unter einem Key-Value Paar oder auch Schlüssel-Wert Paar versteht man eine Speicherstruktur, die aus einem Key (Schlüssel) und einem Value (Wert) besteht. Dabei dient der Schlüssel der eindeutigen Identifikation des Paares, der Value beinhaltet die eigentliche Information des Datensatzes. Die map()-Funktion transformiert die erhaltenen Daten und bildet daraus eine Liste von neuen Key-Value Paaren als Zwischenergebnis.

Liegen diese Zwischenergebnisse vor, beginnt die Reduce Phase. Die reduce()-Funktion wird für jeden Schlüssel des map()-Zwischenergebnisses ausgeführt. Die Übergabeparameter der reduce()-Funktion sind der Key sowie ein Array, das alle Elemente enthält, die dem aktuellen Key zugeordnet wurden. (vgl. Edlich et al. 2011, S.12 ff.) „Die Funktion reduce() akkumuliert die einzelnen Funktionsergebnisse der Listenpaare und reduziert sie damit auf einen Ausgabewert“ (Edlich et al. 2011, S.13). Der Datenfluss und die Phasen des Map/Reduce-Verfahrens sind in Abbildung 2 dargestellt.

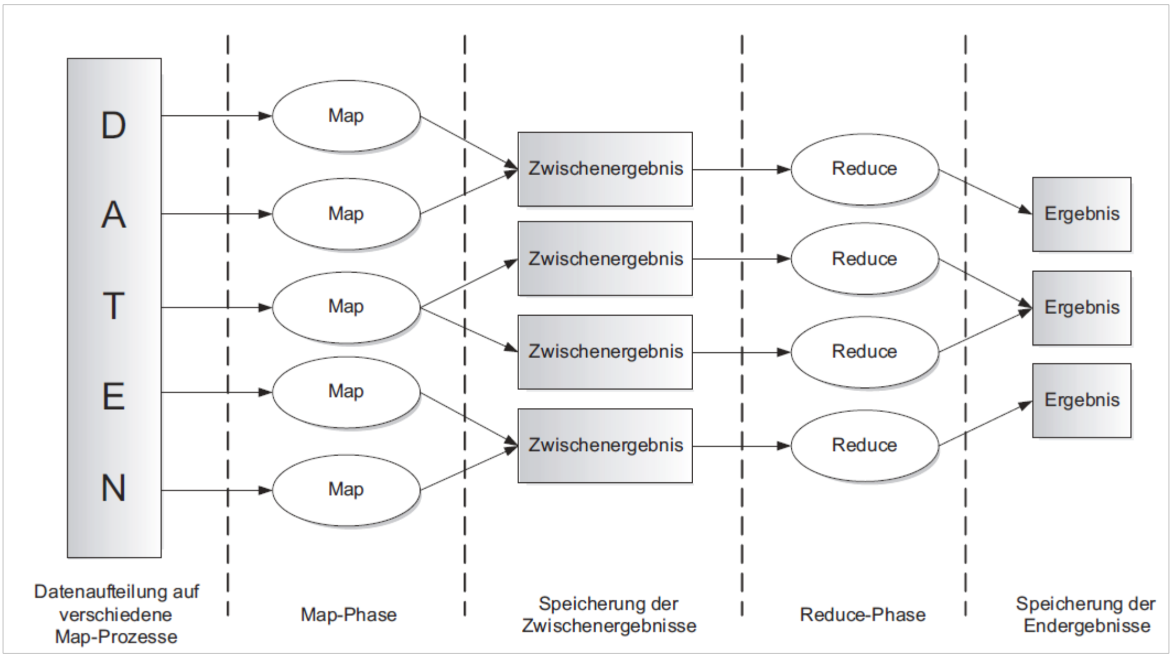


Abbildung 4: Ablauf des Map/ Reduce-Verfahrens | Quelle: Edlich et al. 2011, S.17

Die beiden Prozesse Map und Reduce können parallel durchgeführt werden, wodurch große Datenmengen performant abgearbeitet werden können. Dies bedeutet, dass in einem vernetzten System von mehreren Computern, auch Cluster genannt, jede Recheneinheit einen Teil der Daten gleichzeitig abarbeiten kann. Die Datenlast verteilt sich somit auf mehrere Instanzen. Dies ist möglich, da die beiden Funktionen map() und reduce() unabhängig vom Ergebnis einer anderen map()- oder reduce()- Funktion sind und entsteht durch die Anlehnung der beiden Funktionen an die Grundelemente map und reduce wie sie beispielsweise in Lisp und vielen anderen funktionalen Sprachen vorhanden sind. (vgl. Dean und Ghemawat 2004, S.1)

3.2.3 REST

Viele webbasierte Anwendungen verwenden eine Kombination aus verschiedenen Datenhaltungssystemen für ihre unterschiedlichen Dienste. Ein Online-Shop beispielsweise verwendet für die Session-Verwaltung und den „digitalen Einkaufswagen“ ein Schlüssel-Wert Speichersystem, um eine hohe Verfügbarkeit und Ausfalltoleranz zu garantieren. Sensible Daten, wie die Kundenkonten, werden häufig mit relationalen Datenbanksystemen verwaltet. Für die Kombination dieser heterogenen Datenbanken verwendet mal häufig REST (Representational State Transfer). (vgl. Meier und Kaufmann 2016, S.182 f.)

REST verwendet das HTTP (Hypertext Transfer Protocol) für die Kommunikation mit einer Datenquelle. Die eindeutige Identifikation der Ressourcen, also der Datensätze, erfolgt mit

Hilfe einer URI (Uniform Resource Identifier). Operationen mit den Ressourcen lassen sich über HTTP-Requests ausführen. Abhängig von den Datensätzen werden normalerweise die vier grundlegenden HTTP-Verben: GET, POST, PUT und DELET unterstützt. (vgl. Alachisoft 2019)

Ein URI ist aus fünf Teilen aufgebaut:

1) Schema (Typ des URI beziehungsweise Protokoll) z.B. http, ftp, doi
2) Autorität (Anbieter oder Server)
3) Pfad
4) Abfrage (Angaben zur Identifizierung der Ressource)
5) optional: Fragment (Referenz innerhalb einer Ressource)

Ein Beispiel wäre für eine sehr einfach URI ist https://www.hof-university.de/ueber-uns. (vgl. Meier und Kaufmann 2016, S.184)

Wird ein HTTP-Request mit der Methode GET auf eine URI durchgeführt, wird das Ergebnis dieser Abfrage als formatierte Ausgabe, häufig in Form von JSON, XML oder HTML, ausgegeben (vgl. Alachisoft 2019). Mittels der PUT-Methode können neue Datensätze angelegt werden. Das Löschen von Einträgen aus der Datenbank erfolgt mit DELETE. Um eine Änderung der Daten durchzuführen sendet man mit Hilfe eines POST-Request die neuen Daten an den Server. (vgl. Edlich et al. 2011, S.54 f.)

Die Verwendung von REST hat zwei entscheidende Vorteile. Zum einen garantiert es die horizontale Skalierung, falls das Datenvolumen stark zunimmt (vgl. Meier und Kaufmann 2016, S.184). Zum anderen ist REST sehr flexibel. Jedes System, das http verwendet, kann REST nutzen.

3.3 Kategorisierung von NoSQL Datenbanken

Seit dem Durchbruch der NRDBMS im Jahr 2006 sind zahlreiche verschiedene NoSQL-Datenbanken entstanden. Derzeit sind mehr als 255 NoSQL-Datenbanken unter http://nosql-database.org/ aufgeführt (vgl. Edlich 2019). Obwohl die meisten NoSQL-Datenbanken Open Source Projekte sind, gibt es mittlerweile auch wenige kommerzielle Anbieter. In Kapitel 3.2 hat man bereits gesehen wie schwierig es ist NRDBMS zu definieren, da NRDBMS nur einige der genannten Kriterien erfüllen müssen um als solche zu gelten. Ebenso schwierig wie das Festlegen einer allgemeingültigen Definition ist die Kategorisierung der NRDBMS, da die Eigenschaften der Systeme sehr unterschiedlich sind.

Edlich et al. 2011 unterscheidet zwischen Kern-NoSQL-Systemen (Core) und nachgelagerten (Soft-) NoSQL-Systemen. Die weitere Unterteilung der NoSQL-Kernsysteme erfolgt nach der Art ihrer Datenhaltung und wurde auf der Website http://nosql-database.org veröffentlicht. Diese Kategorisierung hat sich weitgehend durchgesetzt.

Die beiden Hauptgruppen lassen sich in folgende Untergruppen unterteilen:

- „NoSQL-Kernsysteme
    - Wide Column Stores/Column Families
    - Document Stores
    - Key/Value/Tuple Stores
    - Graphdatenbanken
- Nachgelagerte NoSQL-Systeme:
    - Objektdatenbanken
    - XML-Datenbanken
    - Grid-Datenbanken
    - und viele weitere nichtrelationale Systeme“ (Edlich et al. 2011, S.6)

Im Folgenden werden die Datenbankkategorien der NoSQL-Kernsysteme vorgestellt.

3.3.1 Key-Value Stores

Das Konzept der Key-Value Stores oder Schlüssel-Wert Datenbanken, wie sie auf Deutsch heißen, ist schon seit den 70er Jahren im Einsatz erfreut sich aufgrund der stetig weiterwachsenden Datenmengen, die durch das Web 2.0 entstehen, immer größerer Beliebtheit. Da sich Schlüssel-Wert Datenbanken sehr gut eignen um die Daten auf vielen verschiedenen Servern zu verteilen, gehören sie zu den am schnellsten wachsenden NoSQL-Datenbanksystemen. (vgl. Edlich et al. 2011, S.151)

Der Aufbau der Schlüssel-Wert Datenbanken ist recht einfach und erinnert ein bisschen an RDBMS und ihre Primärschlüssel. Im Gegensatz zu RDBMS haben Key-Value Datenbanken jedoch ein sehr einfaches Datenschema. Da alle Daten als Schlüssel-Wert Paare abgelegt werden, lässt sich eine Schlüssel-Wert Datenbank als eine Tabelle mit zwei Spalten darstellen (siehe Tabelle 1). Die erste Spalte enthält den Schlüssel, der zur eindeutigen Bestimmung eines Datensatzes dient. Die Form des Schlüssels ist dabei nicht festgelegt. Er kann aus einer Zeichenkette, einem Binärcode oder einer Zahl bestehen (vgl. Jansen 2011). Die zweite Spalte beinhaltet die Werte also die eigentlichen Informationen des Datensatzes. Abhängig vom verwendeten Datenbankmanagementsystem (DBMS) können als Werte einfache Zeichenketten und Zahlen oder auch komplexe Datentypen wie Hashes, Listen, Sets, Bitmaps oder Objekte verwendet werden (vgl. Redis). Bei Schlüssel-Wert Datenbanken ist zu beachten, dass der Zugriff auf die Daten ausschließlich über den zugehörigen Schlüssel erfolgen kann (vgl. Meier und Kaufmann 2016, S.223).

Sogenannte In-Memory-Datenbanken speichern die Schlüssel-Wert Paare im jeweiligen Hauptspeicher zwischen, während im Hintergrundspeicher permanent eine Abgleichung mit den persistenten Daten erfolgt. Durch das Caching, wie das Zwischenspeichern auch genannt wird, erhöht sich die Geschwindigkeit der Datenverarbeitung nochmal deutlich. (vgl. Meier und Kaufmann 2016, S.224)

Key	**Value**
Autor1	Eric Redmond
Autor2	Jim R. Wilson
Übersetzer	Peter Klicman
Titel	Sieben Wochen, sieben Datenbanken
Erscheinungsjahr	2012
Verlag	O'Reilly Verlag GmbH & Co. KG

Tabelle 1: Tabellarische Darstellung einer Schlüssel-Wert Datenbank

Quelle: eigene Darstellung

Der entscheidende Vorteil von Schlüssel-Wert Datenbanken ist in ihrem einfachen Datenschema begründet. Die Aufteilung des Datenbestands, auch Sharding genannt, kann beliebig ausgebaut werden und ist aufgrund des einfachen Datenschemas leicht zu realisieren. Da jeder Rechner eines Clusters, auch Shards genannt, nur einen Teil der Schlüssel abspeichert, wird die riesige Datenmenge auf eine Vielzahl an Rechnern verteilt. Durch diese Verteilung können auch große Datenmengen effizient geschrieben und ausgelesen werden. (vgl. Meier und Kaufmann 2016, S.224)

Ein Beispiel für das Partitionieren von Daten wäre die Aufteilung einer Bibliotheksdatenbank nach den Anfangsbuchstaben des Nachnamens des Autors auf vier Server: A-F, G-L, M-R, S-Z und ist in Abbildung 5 dargestellt.

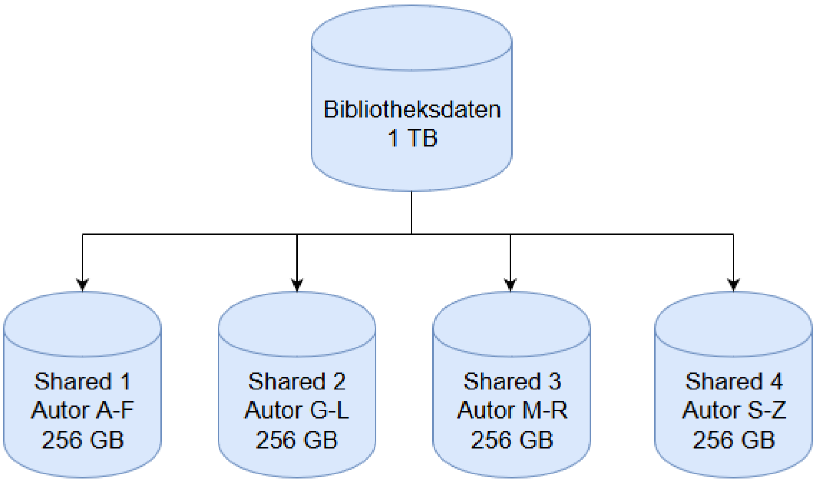


Abbildung 5: schematische Darstellung der Datenbankpartitionierung

Quelle: eigene Darstellung

Das einfache Schema einer Schlüssel-Wert Datenbank kann jedoch auch ein Nachteil sein. Durch die Struktur der Datenbank erfolgt der Zugriff ausschließlich über den Schlüssel. Dadurch ist eine Suche nach den Werten oder das Erstellen komplexer Abfragen nicht möglich. Darüber hinaus kann die Integrität der Daten kann nicht sichergestellt werden, da es nicht möglich ist Metainformationen im Datenmodell zu hinterlegen. Es gibt jedoch mittlerweile Anwendungsschnittstellen, die es ermöglichen die Datenintegrität vor dem Einfügen von Datensätzen überprüfen, sodass bestehende Key-Value Paare nicht überschrieben werden. (vgl. Datenbank Lexikon)

Die bekanntesten Schlüssel-Wert Datenbanken sind Riak, Redis und Voldemort (vgl. Edlich 2019).

3.3.2 Wide Column Stores/Column Families

Schlüssel-Wert Datenbanken verarbeiten große Datenmengen zwar sehr effizient, aber oftmals ist, je nach Anwendungsfall, darüber hinaus eine Strukturierung der Daten in der Datenbank notwendig. Aus diesem Grund gibt es die sogenannten Wide Column Stores oder auch spaltenorientierte Datenbanken. Diese sind eine Erweiterung des Schlüssel-Wert Konzepts um ein Datenschema. (vgl. Meier und Kaufmann 2016, S.226)

Die Datenspeicherung erfolgt bei spaltenorientierten Datenbanken, ähnlich wie bei relationalen Datenbanken, in Tabellen. Im Gegensatz zu den reihenorientierten RDBMS, die die Datensätze untereinander in Zeilen speichern, legen die spaltenorientierten Datenbanken die Datensätze hintereinander in Spalten oder Spaltengruppen ab. (vgl. Edlich et al. 2011, S. 63)

Der Vorteil der spaltenorientierten Datenspeicherung liegt in der Datenorganisation auf dem Datenträger. Die Datensätze werden nach ihren Spalten gruppiert abgelegt. Da beim Auslesen der Daten selten alle Spalten eines Datensatzes benötigt werden, hat sich bei der Speicherung relationaler Tabellen gezeigt, dass es effizienter ist, die Daten für die Optimierung des Lesezugriffs nicht in Form einer Zeile abzulegen. Gruppen von Spalten werden dagegen häufig zusammen gelesen um beispielsweise den maximalen Wert einer Spalte zu ermitteln oder den Durchschnitt aller Werte einer Spalte zu berechnen. Aus diesem Grund macht es Sinn, Daten, die häufig zusammen gelesen werden, in Spaltengruppen zu organisieren um den Zugriff zu optimieren. Die zeilen- und spaltenorientierte Speicherung von Daten ist in Abbildung 6 dargestellt. (vgl. Meier und Kaufmann 2016, S.226)

Autor	BuchID	Preis in €	Seitenzahl
Redmond	15	35	384
Schicker	4	30	360
Lampe	21	50	212

Daten, die gespeichert werden sollen

Redmond	15	35	384	Schicker	4	30	360	Lampe	21	50	212

zeilenorientierte Speicherung

Redmond	Schicker	Lampe	15	4	21	35	30	50	384	360	212

spaltenorientierte Speicherung

Abbildung 6: Darstellung der zeilen- und spaltenorientierten Speicherung | Quelle: eigene Darstellung

„Google hat 2008 mit BigTable ein Datenbankmodell für die verteilte Speicherung von strukturierten Daten vorgestellt und hat damit die Entwicklung von spaltenorientierten Datenbanken maßgeblich beeinflusst“ (ebd.).

Spaltenorientierte Datenbanken bieten den Vorteil, dass der Lesezugriff und die Analyse von Daten äußerst schnell erfolgt (vgl. Edlich et al. 2011, S.63). Auch das Einfügen von Daten in Spalten geschieht effizient. Darüber hinaus sind spaltenorientierte Datenbanken ebenso gut skalierbar wie Schlüssel-Wert Datenbanken. Durch die spaltenorientierte Struktur können die Daten auf mehreren Servern verteilt werden, wodurch die Last eines einzelnen Servers gering gehalten werden kann. (vgl. Meier und Kaufmann 2016, S.228)

Spaltenorientierte Datenbanken haben jedoch auch Nachteile. Das Lesen und Einfügen von Datensätzen über mehrere Spalten hinweg ist deutlich aufwendiger und dauert dadurch länger, da viel zwischen den Spalten gesucht und gesprungen werden muss (vgl. Edlich et al. 2011, S.63)

Die wichtigsten Vertreter spaltenorientierter Datenbanken sind HBase, Cassandra und Amazon SimpleDB (vgl. Edlich 2019).

3.3.3 Document Store

Eine weitere Variante von NRDBMS sind die Dokument-Datenbanken (engl. document store). Sie kombinieren die Schemafreiheit von Schlüssel-Wert Datenbanken mit der Möglichkeit zur Datenstrukturierung. Der Schlüssel ist dabei die Dokument-ID, der ein Dokument als Wert zugordnet wird. Unter einem Dokument sind dabei, anders als der Name vielleicht vermuten lassen würde, nicht beliebige Dokumente wie pdf-Dateien, Word-Dokumente oder Videos zu verstehen. Im Zusammenhang mit Dokument- Datenbanken versteht man unter einem Dokument eine Datei mit strukturierten Daten. Die Syntax der Dokumente ist frei wählbar. Häufig wird verwendet man zur Strukturierung der Daten XML, JSON (JavaScript Object Notation) oder BSON (Binary JSON). Diese Dokumentstruktur stellt eine Liste von Attribut-Wert Paaren dar, die rekursiv wiederum Listen von Attribut-Wert Paaren enthalten können. Eine Beziehung der Dokumente untereinander besteht nicht. Jedes Dokument enthält eine in sich geschlossene Sammlung von Daten. (vgl. Moniruzzaman und Hossain 2013, S.5) Abbildung 7 zeigt die Darstellung einer Dokument-Datenbank.

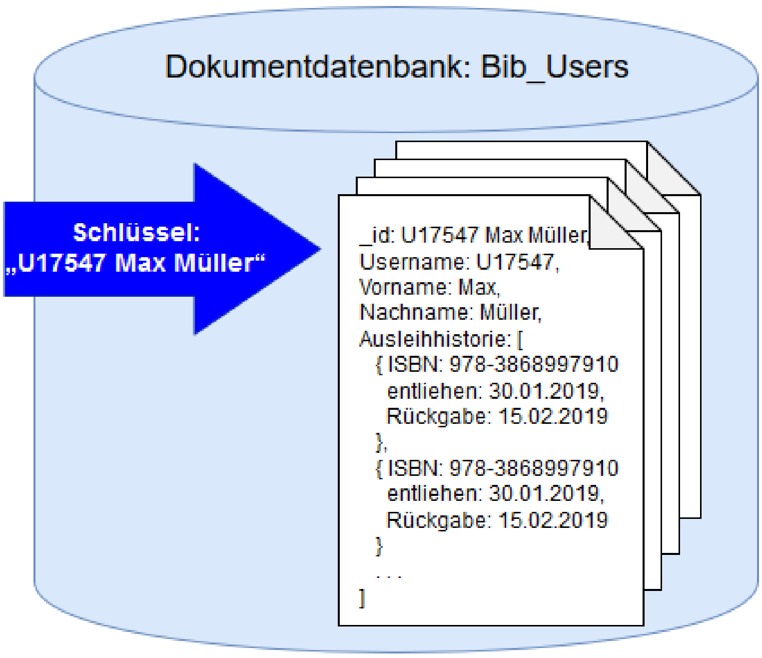


Abbildung 7: Beispiel einer Dokument-Datenbank

Quelle: eigene Darstellung in Anlehnung an Meier und Kaufmann 2016, S.230

Da Dokument-Datenbanken völlig schemafrei sind, muss vor dem Einfügen von Datenstrukturen kein Schema definiert werden. Ein Objekt kann dadurch zu jeder Zeit um beliebige Werte erweitert oder verringert werden. Darüber hinaus können die Werte beliebig lang sein. (vgl. Meier und Kaufmann 2016, S.229)

Diese Flexibilität hat ihren Preis. Durch den Verzicht auf referenzielle Integrität und Normalisierung wird dem Benutzer, der Anwendung und dem Anwendungsentwickler mehr Verantwortung übertagen. (vgl. Edlich et al. 2011, S.117)

Zwei etablierte Systeme sind MongoDB und CouchDB.

3.3.4 Graph Databases

Als vierte und letzte Kategorie der NoSQL-Kernsysteme werden in diesem Kapitel Graph Databases, also Graphdatenbanken, vorgestellt. Sie unterschieden sich deutlich von den bisher vorgestellten Datenmodellen, da sie den Fokus auf Beziehungen zwischen Daten legen, statt auf die Daten selbst. Ein Beispiel für den Einsatz von Graphdatenbanken sind moderne Webanwendungen wie die sozialen Netzwerke Facebook oder Xing. (vgl. Moniruzzaman und Hossain 2013, S.7)

Das Grundkonzept der Graphdatenbanken basiert auf der Graphentheorie der Mathematik (vgl. Edlich et al. 2011, S.209). Ein Graph besteht aus einer Menge von Knoten und Kanten, wobei eine Kante die ggf. gewichtete und/ oder gerichtete Beziehung zwischen den Knoten darstellt (vgl. Fischbach 2019, S.9). Ein Beispiel für einen Graphen zeigt Abbildung 8.

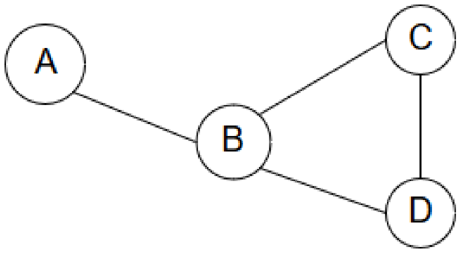


Abbildung 8: Graph mit vier Knoten und ungerichteten, ungewichteten Kanten

Quelle: eigene Darstellung

Bei Graphdatenbanken wird eine Erweiterung des Allgemeinen Graphmodells zur Speicherung der Daten verwendet. Es besteht aus Knoten, Knotenbeziehungen ("Kanten") und Eigenschaften. Sowohl Knoten als auch Kanten können Eigenschaften, die als Schlüssel-Wert-Paare gespeichert werden, tragen. (vgl. Moniruzzaman und Hossain 2013, S.7)

Graphdatenbanken haben einige Gemeinsamkeiten mit RDBMS. Dank der Struktur eines Graphen, lassen sich Beziehungen zwischen den Datensätzen leicht herstellen. Auch in einer relationalen Datenbank lassen sich diese Vernetzungen abbilden. Bei RDBMS ist es jedoch mit einigem Aufwand verbunden die einzelnen Tabellen per (rekursivem) JOIN zu durchlaufen, um komplexe Beziehungen herzustellen. Die Traversierung eines Graphen ist dagegen deutlich einfacher und performanter. (vgl. Edlich et al. 2011, S.225 f.) In Abbildung 9 ist eine Graphdatenbank dargestellt.

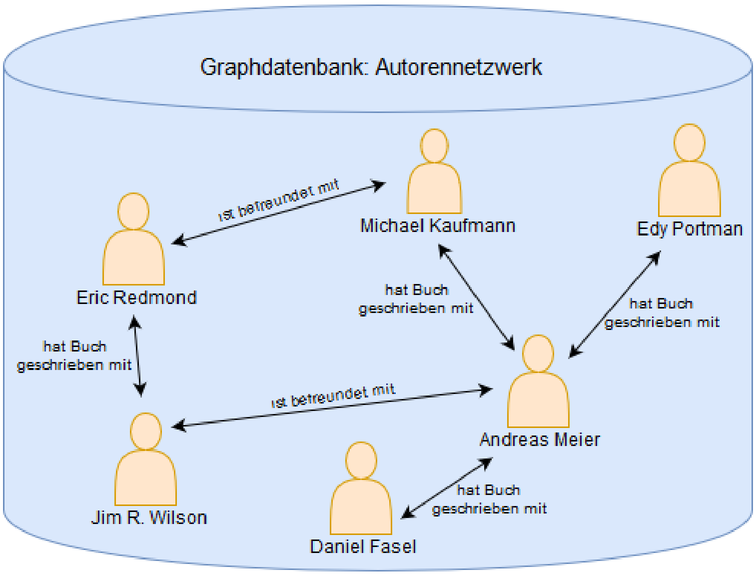


Abbildung 9: Graphdatenbank | Quelle: eigene Darstellung

Im Gegensatz zu RDBMS verwenden Graphdatenbanken aber ein sogenanntes implizites Schema. „[Das bedeutet], dass Datenobjekte zu einem bisher nicht existierenden Knoten- oder Kantentyp direkt in die Datenbank eingefügt werden können, ohne diesen Typ vorher zu definieren. Das Datenbankverwaltungssystem vollzieht aus den vorhandenen Angaben implizit die entsprechende Schemaveränderung nach, d. h. es erstellt den entsprechenden Typ“ (Meier und Kaufmann 2016, S.229).

Graphdatenbanken sind für das Erstellen und Speichern von Beziehungen optimiert. Wenn Beziehungen untersucht und die darin enthaltenen Werte abgefragt und analysiert werden sollen, ist ein für die Suche optimiertes DBMS die bessere Wahl (vgl. Moniruzzaman und Hossain 2013). Darüber hinaus erschwert die Komplexität des Datenmodells die vertikale Skalierung (vgl. Edlich et al. 2011, S.229).

Beispiele für Graphdatenbanken sind Neo4j, InfoGrid und AllegroGraph.

4 Anwendungspotential von Datenbanken für die Speicherung von Geodaten

4.1 Definition Geodaten

Der Begriff Geodaten ist eine Abkürzung für geographische Daten. Unter Geodaten versteht man Informationen über Gegenstände, Geländeformen und Infrastrukturen mit räumlichem Bezug zur Erdoberfläche, die von einem Computer lesbar sind (vgl. Schukraft und Lenz 2005, S.47). Sie bestehen aus Geometrie- und den zugehörigen Sachdaten und bilden die Grundlage für die Erstellung von Karten und Plänen.

Ein Beispiel für Geodaten ist die Lage der Stadt Neusäß:

- Breite 48 Grad 24 Minuten Nord
- Länge 10 Grad 50 Minuten Ost
- Höhe: 485 m ü.M. (Mittelwert)

4.2 Gauß-Krüger-Koordinatensystem

„Die Position eines Punktes im Raum kann in verschiedenen Koordinatensystemen dargestellt werden. Dabei wird die Position durch Koordinaten ausgedrückt. Je nach verwendetem Koordinatensystem hat derselbe Punkt unterschiedliche Koordinatenwerte“. (Wikipedia 2019b)

Das LDBV verwendet, wie die meisten Vermessungsverwaltungen in Deutschland, das Gauß-Krüger Koordinatensystem. Es wurde von Carl Friedrich Gauß entwickelt und von Johann Heinrich Louis Krüger veröffentlicht. Das am häufigsten verwendeten Gauß-Krüger-Koordinatensystem ist das 3°-Gauß-Krüger-Koordinatensystem (3GK). Dabei wird die Erde ausgehend vom Nullmeridian, der durch die Sternwarte Greenwich bei London festgelegt ist, in 3° breite Meridianstreifen unterteilt. Jeder dieser Streifen geht vom Nord- bis zum Südpol und seine begrenzenden Meridiane liegen genau 3° auseinander. In der Mitte des Meridianstreifens verläuft der Mittelmeridian.

Die Y-Achse des Koordinatensystems ist der Äquator. Der Ursprung des Koordinatensystems ist der Schnittpunkt des Meridianstreifens mit dem Äquator.

Die Angabe der Koordinaten erfolgt als x und y Werte. Der x-Wert wird auch als Hochwert bezeichnet, beginnt am Äquator mit 0 und ist nach Norden hin positiv. Dem y-Wert, der auch als Rechtswert bezeichnet wird, wird die Kennziffer des Mittelmeridians vorangestellt. Anschließend folgt die Position im Koordinatensystem, die nach Osten positiv gezählt wird. (vgl. Staatsbetrieb Geobasisinformation und Vermessung Sachsen 2017)

Der Paradeplatz in Mannheim hat folgende Gauß-Krüge-Koordinaten:

- Rechtswert 3461404 m
- Hochwert 5483498 m (vgl. Wikipedia 2019a).

Abbildung 10 zeigt ein Raster im Gauß-Krüger-Koordinatensystem für Deutschland. Die roten Linien markieren die Mittelmeridiane, die blauen Linien die Meridianstreifenränder.

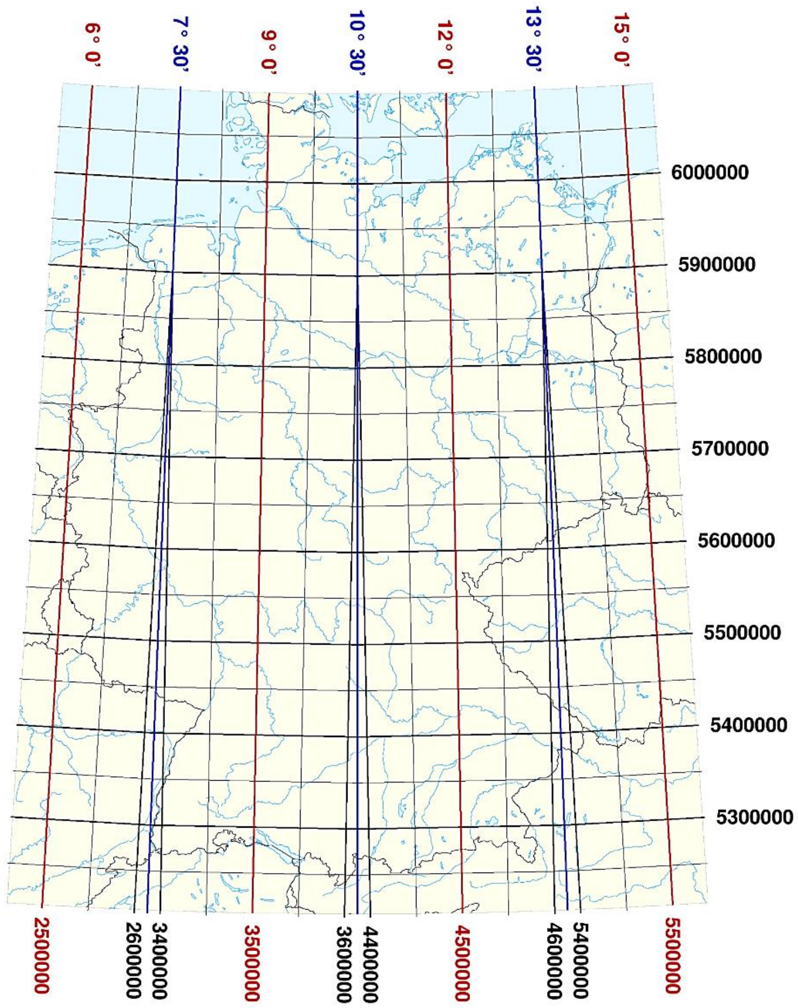


Abbildung 10: Raster im Gauß-Krüger Koordinatensystem

Quelle: https://upload.wikimedia.org/wikipedia/commons/1/12/Gau%C3%9F-Kr%C3%BCger-Raster_Deutschland.png

4.3 Geodaten in RDBMS

4.3.1 Erweiterungen für Geodaten

Um Geodaten in einer relationalen Datenbank speichern zu können, muss diese eine räumliche Erweiterung besitzen. In einer Datenbank mit räumlicher Erweiterung wird eine SQL-Spalte mit Geometriewert als Spalte mit einem Geometrietyp implementiert. Es gibt eine Reihe von SQL-Geometrietypen sowie Funktionen für diese Typen zum Erstellen und Analysieren von Geometriewerten. (vgl. MySQL 8.0 Reference Manual)

Wie bereits erwähnt soll eine MySQL Datenbank als Vertreter der RDBMS für das Performancemonitoring verwendet werden. Die räumlichen Erweiterungen von MySQL ermöglichen die Generierung, Speicherung und Analyse von geografischen Daten. MySQL implementiert verschiedene Datentypen, die die Darstellung räumlicher Werte ermöglichen. Darüber hinaus gibt es auch Funktionen mit deren Hilfe die räumlichen Daten verändert werden können. (vgl. MySQL 8.0 Reference Manual)

Folgende Datentypen sind in MySQL unter anderem implementiert:

- Point
- Linestring
- Polygon
- Multipoint
- Multipolygon

4.3.2 Datenformate

Unabhängig vom Datentyp gibt es unterschiedliche Formate, in denen die Geometriedaten in der Datenbank gespeichert werden.

Das verbreitetste Datenformat ist das Well Known Text (WKT) Format. Dieses ist aus der Simple Features Spezifikation des Open Geospatial Consortium (OGC) hervorgegangen. Das OGC ist ein internationales Konsortium aus mehr als 250 Unternehmen, Agenturen und Universitäten, die an der Entwicklung öffentlich verfügbarer konzeptioneller Lösungen beteiligt sind, die bei allen Arten von Anwendungen zur Verwaltung von Geodaten nützlich sein können. Das WKT-Format eines Geometriefeldes ermöglicht den Datenaustausch im ASCII-Format. (vgl. The Open Geospatial Consortium)

Beispiele für Daten im WKT-Format sind in Tabelle 2 gelistet.

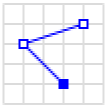
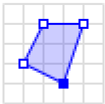
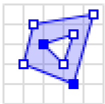
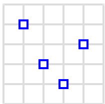
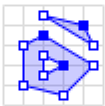
Datentyp	Visualisierung	WKT-Format
Point		POINT (30 10)
LineString		LINESTRING (30 10, 10 30, 40 40)
Polygon		POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))
		POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))
MultiPoint		MULTIPOINT ((10 40), (40 30), (20 20), (30 10))
MultiPolygon		MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)), ((20 35, 10 30, 10 10, 30 5, 45 20, 20 35), (30 20, 20 15, 20 25, 30 20)))

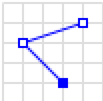
Tabelle 2: Daten im WKT-Format

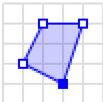
Die MySQL-Funktion `GeometryFromText(wkt[, srid])` wandelt die WKT-Darstellung in einen für MySQL geeigneten Geometriewert um. Zusätzlich zu den WKT-Daten kann dieser Funktion optional die SRID übergeben werden. Die SRID (Spatial Reference Identifier) gibt den Identifizierungsschlüssel des räumlichen Referenzsystems an. Das Gauß-Krüger Koordinatensystem hat für den Bereich Bayern die SRID 31468. (vgl. Zimmermann 2012, S.151)

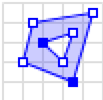
4.4 GeoDaten in NRDBMS

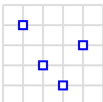
Da die Speicherung von mehrdimensionalen Daten sehr komplex ist, ist es nicht verwunderlich, dass nicht alle NRDBMS gleichermaßen für die Speicherung von Geodaten geeignet sind. Vor allem Dokument-Datenbanken eignen sich jedoch gut zur Speicherung von Geodaten. Das Konzept schemaloser Dokumente zusammen mit einer multidimensionalen Abfragemöglichkeit und der Implementierung von Map/Reduce stellt eine interessante Alternative zur Haltung von Geodaten in RDBMS dar. Darüber hinaus eignen sich Grapdatenbanken zur Speicherung von Geodaten, da sie eine ebenso komplexe Strukturierung von Daten ermöglichen wie man es von RDBMS gewohnt ist.

Die Speicherung von Geodaten wird am Beispiel der Dokument-Datenbank MongoDB betrachtet. In MongoDB können Geodaten als GeoJSON-Objekte oder als Koordinaten-paare gespeichert werden. Im Folgenden wird die Speicherung der Geodaten als GeoJSON-Objekte eingegangen. Ein GeoJSON-Dokument besteht aus zwei Feldern. Das erste Feld namens type gibt den GeoJSON-Objekttyp an. Das zweite Feld trägt den Namen coordinates und enthält die Koordinaten des Objekts. In Tabelle 3 sind einige Beispiele für Daten im GeoJSON-Format aufgeführt.

Datentyp	Visualisierung	GeoJSON
Point		{ type: "Point", coordinates: [40, 5] }
LineString		{ type: "LineString", coordinates: [[40, 5], [41, 6]] }

Polygon		{
		type: "Polygon",
		coordinates: [[[0 , 0] , [3 , 6] , [6 , 1] , [0 , 0]]]
		}

		{
		type : "Polygon",
		coordinates : [
		[[0 , 0] , [3 , 6] , [6 , 1] , [0 , 0]],
		[[2 , 2] , [3 , 3] , [4 , 2] , [2 , 2]]
		]
		}

MultiPoint		{
		type: "MultiPoint",
		coordinates: [
		[-73.9580, 40.8003],
		[-73.9498, 40.7968],
		[-73.9737, 40.7648],
		[-73.9814, 40.7681]
		]
		}

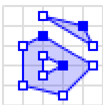
MultiPolygon		{
		type: "MultiPolygon",
		coordinates: [
		[[[-73.958, 40.8003], [-73.9498, 40.7968],
		[-73.9737, 40.7648], [-73.9814, 40.7681],
		[-73.958, 40.8003]]],
		[[[-73.958, 40.8003], [-73.9498, 40.7968],
		[-73.9737, 40.7648], [-73.958, 40.8003]]]
		]
		}

Tabelle 3: Daten im GeoJSON-Format

4.5 Entscheidungsgrundlage für MongoDB

Als Vertreter der NRDBMS wird für das Performancemonitoring MongoDB verwendet. Es zählt zu den weltweit am weitesten verbreiteten NRDBMS. Der Name mongo stammt von dem englischen Begriff "humongous" und lässt sich mit "gigantisch" oder "riesig ins Deutsche übersetzen. MongoDB basiert auf der Programmiersprache C++ und ist für Windows, Mac OS X und Linux erhältlich. (vgl. Wyllie)

Das Ziel von MongoDB ist die Kombination möglichst umfangreicher Abfragemöglichkeiten mit guter Skalierbarkeit und Performance, um die Lücke zwischen RDBMS und den Schlüssel-Wert-Datenbanken zu schließen. Ein weiterer Pluspunkt von MongoDB ist die Unterstützung aller im Web häufig verwendeten Programmiersprachen wie Javascript, PHP, Perl, Ruby, Python und Java. (vgl. Edlich et al. 2011, S.131)

„[Darüber hinaus steht] hinter MongoDB ein Team bekannter Größen aus erfolgreichen Web-Unternehmen in den USA […], die ihre langjährigen technischen Erfahrungen in der Entwicklung von MongoDB bündeln.“ (Edlich et al. 2011, S131)

MongoDB gehört zu der Kategorie der Document-Stores, die in Kapitel 3.3.3 beschrieben wurden. Es speichert die Daten in zusammenhängenden Dokumenten in einer JSON-Struktur ohne relationale Beziehungen zu anderen Datenstrukturen. Darüber hinaus implementiert MongoDB den Map/Reduce Algorithmus, sodass große Datenbankabfragen verteilt ausgeführt werden können. (vgl. Hollosi 2012, S.435)

Ein besonders großer Vorteil von MongoDB ist das sogenannte Journaling. Dieses sorgt dafür, dass nach einem Absturz des Systems beim Neustart alle unbeendeten Transaktionen automatisch reproduziert werden. (vgl. Edlich et al. 2011, S.132)

Bei RDBMS muss man vor dem Einfügen von Datensätzen das Datenbankschema mit dem Befehl CREATE TABLE anlegen. Da MongoDB zu den schemaloses Datenbanken gehört, erübrigt sich dieser Schritt. Es geht sogar so weit, dass auch das Erstellen der Datenbank nicht notwendig ist, da MongoDB diese automatisch anlegt, sobald sie erstmals verwendet wird. In der Datenbank werden die Dokumente in einer Collection (deutsch: Kollektion) zusammengefasst, die mit den Tabellen in relationalen Datenbanken verglichen werden können. Die Dokumente, die in einer Collection abgelegt werden, müssen, im Gegensatz den Daten in einer Tabelle einer RDBMS, nicht dieselbe Struktur haben. (vgl. Hollosi 2012, S.437)

RDBMS verwenden die Abfragesprache SQL. Eine so domänenspezifische Abfragesprache gibt es für NRDBMS aufgrund ihrer sehr unterschiedlichen Strukturen nicht. Bei MongoDB erfolgt die Interaktion mit JavaScript. Das zentrale Element für die Abfrage von Daten ist das Objekt db, das die aktuell verwendete Datenbank darstellt. Der Zugirff auf eine Collection erfolgt mit `db.collection.method().` Mit dem Befehl `db.helb()` kann man sich alle verfügbaren Methoden anzeigen lassen. Die Basisfunktionalitäten create, insert, update und delete stehen selbstverständlich zur Verfügung. Tabelle 4 zeigt den Vergleich zweier SQL-Befehle und die entsprechenden Abfragen in MongoDB. (vgl. Hollosi 2012, S.443 ff.)

Beschreibung	SQL	Abfrage in MongoDB
Anzeigen aller Elemente der Tabelle gebaeude	SELECT * FROM gebaeude;	db.getCollection('gebaeude').find({})
Datensatz des gebaudes mit der ID 17440394 soll angezeigt werden	SELECT * FROM gebaeude WHERE ID = 17440394	db.getCollection('gebaeude').find({ID:17440394})

Tabelle 4: Vergleich SQL und MongoDB-Abfrage

MongoDB ist aus den nachfolgenden Gründen eine geeignete Wahl für das Performancemontitoring. Die strukturierte Ablage von Dokumenten erfolgt im JSON-Format. Für die Speicherung von Geodaten stellt MongoDB die Erweiterung GeoJSON bereit. Darüber hinaus können csv-Dateien in die MongoDB eingelesen werden, wodurch sich der Datenimport leicht realisieren lässt. Zu beachten ist dabei, dass die csv-Datei die Werte durch Tabs oder Komma trennen muss. Eine Spezifikation des Trennzeichens ist (derzeit noch) nicht möglich. Da das Performancemonitoring in der Programmiersprache Java geschrieben werden soll, ist es ein weiterer Vorteil, dass MongoDB eine Java-Schnittstelle zur Verfügung stellt. Die einfache und plattformunabhängige Installation von MongoDB sind weitere Pluspunkte. Auch die grafische Oberfläche Robo3T erleichtert die Arbeit sehr und ist intuitiv bedienbar.

5 Entwurf des Performancemonitorings von MySQL und MongoDB

5.1 technische Voraussetzungen

Die Leistung einer Datenbank hängt nicht nur von der Art der Datenbank selbst ab, auch die Hardware, auf der das Datenbankmanagementsystem installiert ist, trägt entscheidet zur Geschwindigkeit der Datenbank bei.

Um sicherzustellen, dass die Rahmenbedingungen für die SQL- und die NoSQL- Datenbank identisch sind, werden zwei virtuelle Maschinen (VMs) erstellt. Unter Virtualisierung versteht Methoden zur Abstraktion von Ressourcen mit Hilfe von Software (vgl. Mandl 2014, S. 297). Mit Hilfe der Software wird ein eigenständiges System simuliert, das einen vorgegebenen Teil des Speichers, der Rechenleistung und der Festplatte des Computers verwendet. „Die Anwendungen, die in einer VM laufen, bemerken dies nicht. Anforderungen der Betriebssystem-Instanzen werden von der Virtualisierungssoftware transparent abgefangen und auf die real vorhandene oder emulierte Hardware umgesetzt“ (Baun 2017, S.231). Die einzelnen VMs sind voneinander unabhängig, sodass auf einer physischen Hardware mehrere VMs betrieben werden können.

Es gibt mehrere verschiedene Konzepte und Technologien der Virtualisierung. Für das Erstellen der VMs für das Performancemonitoring wird das Prinzip der vollständigen Virtualisierung angewendet. Wie der Name schon sagt, wird dabei einer virtuellen Maschine eine vollständige, virtuelle PC-Umgebung inklusive eigenem BIOS geboten. „Jedes Gastbetriebssystem erhält eine eigene virtuelle Maschine mit virtuellen Ressourcen wie Prozessor, Hauptspeicher, Laufwerke und Netzwerkkarten“ (Baun 2017, S.235). Basis dieser Lösung ist der Hypervisor, der häufig auch als Virtual Machine Mangager bezeichnet wird. Er ist für die Zuweisung der Hardwareressourcen an die VMs zuständig und ermöglicht das parallele Ausführen mehrerer Betriebssysteme auf einer physikalischen Hardware. Die einzelnen Betriebssysteme sind so voneinander isoliert, dass sie die Existenz des jeweils anderen nicht kennen. (vgl. Baun 2017, S.235)

Die Technologie der vollständigen Virtualisierung kann in die zwei Untertechnologien Typ-1-Hypervisor und Typ-2-Hypervisor gegliedert werden (vgl. Mandl 2014, S.306).

Ein Typ-1-Hypervisor, auch als Bare-Metal-Hypervisor bekannt, ist eine Virtualisierungssoftware, die kein Wirtssystem benötigt sondern direkt auf der Computer-Hardware installiert wird. Dieses Modell hat den Vorteil, dass der Server weniger Mehraufwand für die

Virtualisierung leisten muss und somit eine größere Anzahl virtueller Maschinen unterstützen kann. (vgl. Lampe 2010, S.75)

Beim Typ-2-Hypervisor wird die Virtualisierungsschicht wie eine Anwendung auf dem Betriebssystem des Servers installiert. Das Betriebssystem des Servers fungiert als Hostsystem und die Gastsysteme laufen wie gewohnt in vom Hypervisor erstellten virtuellen Maschinen. Durch diese dreistufige Architektur geht viel Rechenleistung verloren, wodurch die Zahl der virtuellen Maschinen, die gleichzeitig genutzt werden können, beschränkt ist. (vgl. Lampe 2010, S.75)

Die Abbildung 11 und Abbildung 12 zeigen einen Typ-1-Hypervisor bzw. einen Typ-2-Hypervisor.

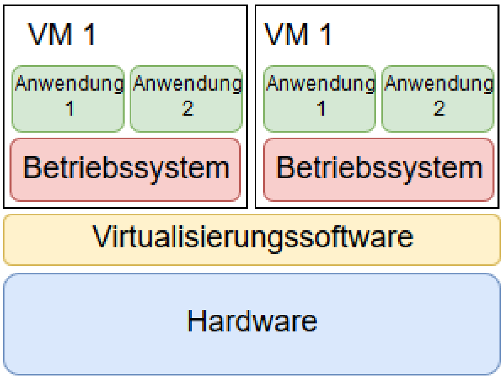


Abbildung 11: Typ-1-Hypervisor
Quelle: eigene Darstellung

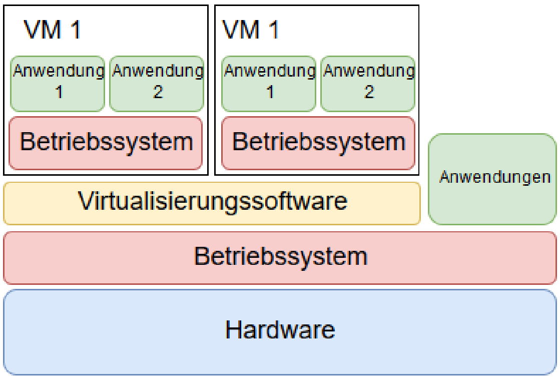


Abbildung 12: Typ-2-Hypervisor
Quelle: eigene Darstellung

Den beiden durch die Virtualisierung entstandenen VMs stehen die identischen Ressourcen zur Verfügung, sodass beide Datenbanken die gleichen Voraussetzungen haben und die Messergebnisse nicht aufgrund der Hardwareausstattung verfälscht werden.

5.2 Erläuterungen zu den Verwendeten Daten

Bei den Daten, die für das Performancemonitoring verwendet werden, handelt es sich um drei verschiedene Geodatensätze von Gebäuden eines Ortes. Diese Geodaten wurden im WKT-Format exportiert und werden dankenswerterweise vom LDBV als csv-Dateien zur Verfügung gestellt.

Der erste Datensatz enthält die Geodaten der Gebäude der Stadt Neusäß, einer kleinen Stadt im Westen Augsburgs. Die Stadt Neusäß erstreckt sich über eine Fläche von 25,2 km² und zählte am 31.12.2018 22.198 Einwohner (vgl. Stadt Neusäß). Der Datensatz umfasst 14.536 Datensätze. In der MySQL-Datenbank benötigt der Datensatz 4718,59 Kilobyte (KB) und in der MongoDB 2000 KB. Der unterschiedliche Speicherplatzbedarf in den beiden Datenbanken ist auf das völlig verschiedene Konzept zur Datenhaltung zurückzuführen.

Darüber hinaus werden die Geodaten der Stadt Augsburg verwendet. Augsburg ist 146,8 km² groß und hat 295.895 Einwohner (vgl. Stadt Augsburg 2018). Der Datensatz der Stadt Augsburg umfasst 90.773 Datensätze und ist in der MySQL-Datenbank 25,8 Megabyte (MB) groß. In der MongoDB beansprucht der Datensatz 15 MB.

Der dritte Datensatz enthält die geografischen Daten von ganz Bayern. Bayern erstreckt sich auf einer Fläche von 70 550 km² und hat 13.060.186 Einwohner (vgl. Landesamt für Statistik). Der Geodatensatz umfasst 9.127.738 Gebäudedaten. In der MySQL-Datenbank werden knapp 2,6 Gigabyte (GB) für die Speicherung der Daten benötigt, in der MongoDB etwas mehr als 1,6 GB.

5.3 Performancemonitoring

5.3.1 Vorbereitungen

Mit Hilfe der Ergebnisse der Performancemessung soll das LDBV entscheiden können, ob eine Umstellung von einer SQL-Datenbank auf eine NoSQL-Datenbank lohnend ist. Aus diesem Grund werden für die Performancemessung die drei originalen Geodatensätze des LDBV verwendet, die bereits in Kapitel 5.3 vorgestellt wurden.

Ursprünglich verwendet das LDBV zur Speicherung der Geodaten das herstellerspezifische Format SDO_GEOMETRY. Durch das Exportieren der Daten im WKT-Format durch das LDBV ist es aber möglich die Daten in MySQL und MongoDB zu importieren. Die Daten werden in Form von csv-Dateien bereitgestellt.

Um die Performance der beiden unterschiedlichen Datenbanken messen zu können, müssen zunächst die beiden Datenbanken auf dem Ubuntu 18.04 System installiert werden. Die Installation der MySQL-Datenbank mit einer geeignete Weboberfläche zur Erleichterung der Arbeit und besseren Visualisierung der Daten wird in Kapitel 2.2 der Praxisarbeit beschrieben. Das Aufsetzen einer MongoDB inklusive grafischer Oberfläche wird in Kapitel 2.3 gezeigt.

Vor dem Import der Daten muss in der MySQL-Datenbank zunächst die Datenbanktabelle der originalen Datenbank in der Testdatenbank aufgebaut werden. Das zugehörige sql-Skript zum Erstellen der Datenbanktabelle wurde ebenfalls vom LDBV bereitgestellt. Da es jedoch Unterschiede bei den Datentypen und deren Bezeichnung zwischen Oracle und

MySQL gibt, muss dieses Skript noch modifiziert werden. In Tabelle 5 werden die wichtigsten MySQL und Oracle Datentypen gegenübergestellt (vgl. OracleHelpCenter). Da es sich bei der MongoDB um eine schemafreie Datenbank handelt, erübrigt sich das Erstellen des Tabellenschemas vor dem Import der Daten.

Beschreibung	MySQL Datentyp	Oracle Datentyp
Zeichen	CHAR	CHAR
Datumsangaben ohne Zeitangabe	DATE	DATE
Datumsangaben mit Zeitangabe	DATETIME	DATE
Fließkommazahl mit einer Genauigkeit von 24	DOUBLE	FLOAT (24)
Ganzzahliger Zahlenwerte	INTEGER	NUMBER
Zeichen	CHAR	CHAR
Zeichenkette variabler Länge	VARCHAR	VARCHAR2
Multipolygon (bei Geodaten)	MULTIPOLYGON	SDO_Geometry

Tabelle 5: Gegenüberstellung der MySQL und Oracle Datentypen | Quelle: Oracle Help Center 2005

Darüber hinaus müssen die Daten im UTF-8 Format vorliegen, da dies die von MongoDB standardmäßig unterstützte Kodierung ist. Bei MySQL ist die Kodierung frei wählbar, doch sollte auch hier UTF-8 verwendet werden. Eine Konvertierung des Datenformats ist in Linux mit dem Befehl `iconv` möglich und wird in Kapitel 3 der Praxisarbeit näher beschrieben.

Die vom LDBV exportierten Daten enthalten den Datenstring „(null)“ wenn ein Datenfeld nicht befüllt ist. Um nicht (null) als Datenstring in die beiden Datenbanken zu importieren, sondern wirklich leere Felder zu erhalten, wird dieser String mittels `sed` durch „nichts“ ersetzt.

Nachdem die Daten aufbereitet wurden, werden sie in die MySQL-Datenbank `mit LOAD DATA` und in die MongoDB mit `mongoimport` importiert.

5.3.2 Ablauf des Performancemonitorings

Stehen die Daten in den Testsystemen bereit, kann das Performancemonitoring durchgeführt werden. Dieses beschränkt sich auf die Analyse des Lesezugriffs auf die Daten. Im LDBV werden die vorhandenen Geodaten nach der Vermessung von Gebäuden zwar stetig aktualisiert und auch neue, einzelne Geodatensätze in die Datenbank eingepflegt, der Schwerpunkt liegt jedoch auf dem Auslesen vieler Geodaten, wie sie beispielsweise bei der Erstellung von Einsatzplänen für Polizei und Feuerwehr oder für die Planung neuer Baugebiete benötigt werden. Sonderfunktionen für raumbezogene Daten wie die Umkreissuche oder das Ermitteln des kürzesten Weges zwischen zwei Punkten werden beim

Auslesen der Daten nicht betrachtet. Grund hierfür ist, dass die Software somit individueller für das Performancemonitoring verschiedener Datenbanken eingesetzt werden kann und nicht auf Geodaten beschränkt ist.

Es werden zwei verschiedene Arten des Lesezugriffs im Rahmen des Performancemonitorings durchgeführt.

Zum einen wird die Dauer einer Datenabfrage ermittelt, wenn ein Benutzer in kurzer Zeit viele Datensätze nacheinander aus der Datenbank ausliest. Für die Durchführung der Performancemessung wird ein Java-Programm geschrieben. Auf die MySQL Datenbank wird mittels JDBC (Java Database Connectivity) zugegriffen. JDBC ist der Industriestandard für datenbankunabhängige Verbindungen zwischen der Programmiersprache Java und einer Vielzahl von Datenbanken wie beispielsweise SQL-Datenbanken, Tabellenkalkulationen oder Flat Files (vgl. Java SE Technologies - Database). JDBC kann zwar ebenfalls für MongoDB verwendet werden, es gibt jedoch in Java auch die deutlich einfachere Variante `MongoClient mongoClient = new MongoClient();` um eine Verbindung zur MongoDB-Datenbank herzustellen. In beiden Fällen soll der Name der Datenbank, aus der die Daten gelesen werden sollen, beim Starten des Programms als Parameter an dieses übergeben werden können.

Die Abfrage einer Vielzahl von Daten (z.B. 10, 100, 250, 1000 Datensätze) von einem Benutzer wird im Java-Programm mit einer Schleife abgebildet. Mit Hilfe von Schleifen kann man festlegen, wie oft ein bestimmter Teilabschnitt eines Java Programms ausgeführt werden soll. Innerhalb dieser Schleife wird die SQL-Abfrage durchgeführt und so oft wiederholt, wie es der Schleifenzähler vorgibt.

Jedem Gebäude ist eine ID zugeordnet. Anhand dieser ID ist jedes Gebäude und somit auch der zugehörige Datensatz eindeutig identifizierbar. Beim Auslesen der Daten sollen alle Elemente eines Datensatzes ausgelesen werden. Dabei ist zu beachten, dass die Datensätze zufällig aus der Datenbank ausgelesen werden und nicht etwa nach beispielsweise der ID auf- oder absteigend sortiert. Ein geordneter Zugriff auf die Daten könnte schneller sein, was die Aussagekraft der Zeitmessung negativ beeinflusst. MySQL stellt für das zufällige Auslesen der Datensätze die Funktion rand() zur Verfügung. Das Verwenden der `rand()` Funktion verlängert die Dauer des Lesezugriffs jedoch deutlich. (vgl. MySQL 8.0 Reference Manual) Für MonogDB gibt es für das Auslesen zufälliger Datensätze die Funktion `random.sample`. Auch diese Funktion geht jedoch zu Lasten der Geschwindigkeit. (vgl. Leite 2016) Als Lösung für dieses Problem, werden bereits vor dem Beginn der Zeitmessung eine Art Wörterbuch aufgebaut, das für jede Testreihe die IDs festlegt, die für die Datenbankabfrage verwendet werden sollen. Das Wörterbuch soll dabei mit einer Schlüssel-Wert Struktur realisiert werden. Für den Aufbau des Wörterbuchs müssen zunächst alle IDs, die in der jeweiligen Datenbanktabelle vorhanden sind, in eine Liste (IDList) geladen werden. Im Wörterbuch werden anschließend jeder Testreihe (gibt die Anzahl der Datensätze an, die gelesen werden sollen) zufällige IDs der IDList zugeordnet. Die Testreihe ist dabei der Schlüssel, die zufälligen IDs die zugehörigen Werte. Tabelle 6 zeigt zwei Beispiele für den Aufbau des Wörterbuchs.

Testreihe (Schlüssel)	**Zufällige IDs (Werte)**
10	154, 5, 2589, 1425, 25694, 2, 295, 8597, 5786, 1
15	256, 2476, 26895, 2456, 2, 159, 657, 157, 9, 4, 865, 5468, 123, 9876, 2547

Tabelle 6: schematische Darstellung des Wörterbuchs

Nach der Erstellung des Wörterbuchs kann mit dem Auslesen der Datensätze begonnen werden. Um die Dauer der Datenbankabfrage ermitteln zu können, wird sowohl vor der Ausführung der Datenbankabfrage als auch nach deren Ausführung die aktuelle Systemzeit auf Millisekunden genau ermittelt. Aus der Differenz der beiden Systemzeiten ergibt sich schließlich die Dauer des Lesezugriffs. Jede Zeitmessung wird dabei 10mal wiederholt um repräsentative Messergebnisse zu erhalten.

Zum anderen wird die Antwortzeit der Datenbank gemessen, wenn mehrere Benutzer, gleichzeitig lesend auf die Datenbank zugreifen. Da diese Messung vermutlich sehr lang dauern wird, wird sie nur für den kleinsten Datensatz, also für Neusäß, ausgeführt. Zunächst müssen auf dem Ubuntu 18.04 System die gewünschte Anzahl an Benutzern angelegt werden. In jedem Benutzerverzeichnis wird anschließend das ausführbare Java-Programm, die Performance.jar, abgelegt. Mit dem Linuxbefehl `at` kann festgelegt werden, wann mit der Ausführung eines Programms oder Skripts begonnen werden soll. Um den gleichzeitigen Datenzugriff zu simulieren, wird für jeden Benutzer die Startzeit der Programmausführung auf die exakt gleiche Uhrzeit festgelegt.

Die Ergebnisse der Zeitmessung werden durch Semikolon getrennt in eine .txt-Datei geschrieben. Standardmäßig trägt die Datei den Namen PerformanceSQL_+Datenbankname+username.txt bzw. PerformanceNoSQL_+Datenbankname+username.txt. Der username ist dabei frei wählbar und wird, ebenso wie die Datenbank als Parameter, an das java-Programm übergeben. Der Username ist vor allem beim Multiuser-Monitoring relevant, da so für jeden User eine Datei mit Messergebnissen angelegt wird und nicht alle User in die gleiche Datei schreiben.

Darüber hinaus soll eine Logging-Datei erstellt werden, in die stündlich die aktuelle Uhrzeit sowie die aktuelle Testreihe geschrieben wird.

Abbildung 13 visualisiert den eben beschriebenen Ablauf des Performancemonitorings.

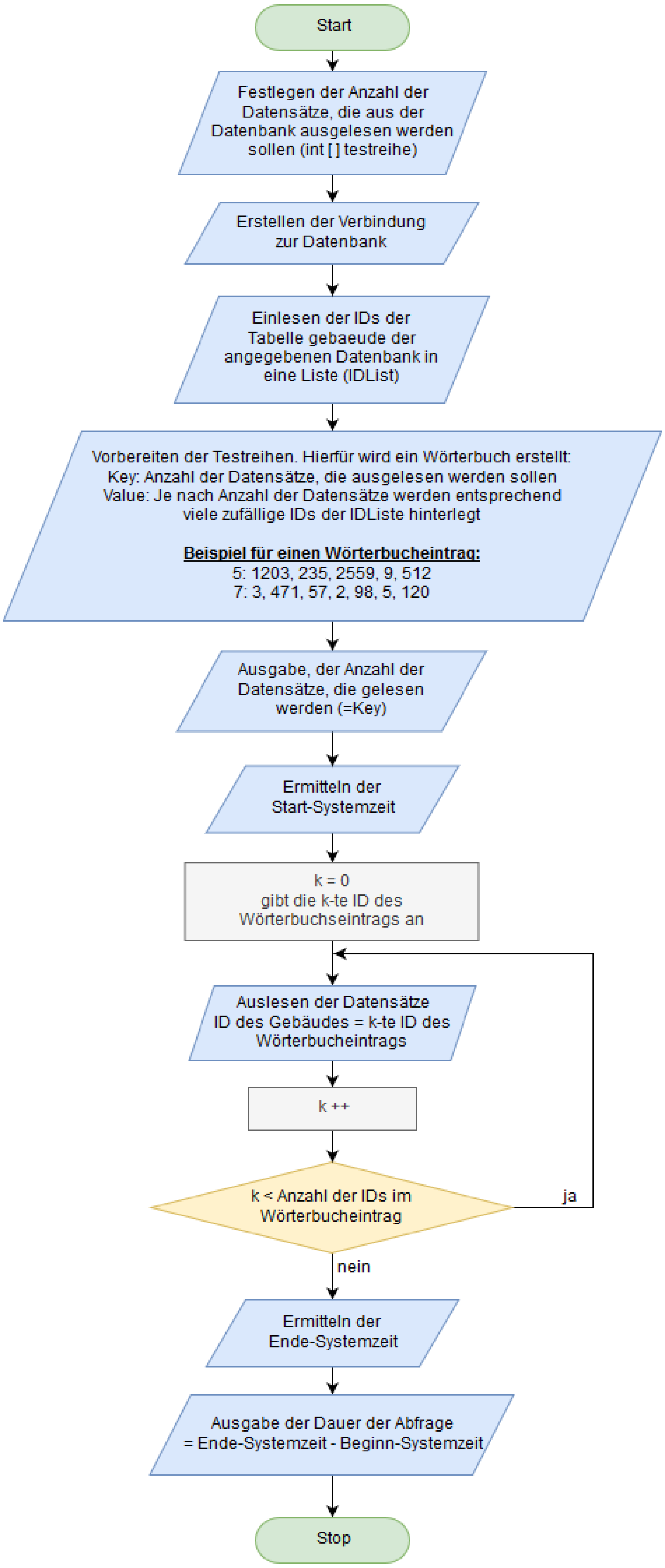


Abbildung 13: Programmablaufplan

6 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde zunächst auf die technischen Grundlagen NRDBMS eingegangen. Im Gegensatz zu RDBMS ist bei NoSQL-Datenbanken das oberste Ziel möglichst performant und leicht skalierbar zu sein. RDBMS legen dagegen Wert auf konsistente Datenhaltung und die Vermeidung von Redundanzen.

Die Wahl des NRDBMS hängt vom jeweiligen Anwendungsfall ab. Geht es rein um die Geschwindigkeit, sind Schlüssel-Wert Datenbanken zu bevorzugen. Möchte man jedoch die Zusammenhänge von Daten modellieren, sollte man Graphdatenbanken wählen. Führt man viele Berechnungen auf Spaltenwerte durch, haben sich spaltenorientierte Datenbanken bewährt. Für die Speicherung von Geodaten ist es dagegen sinnvoll eine Dokument Datenbank zu verwenden.

Für das Speichern von Geodaten gibt es, je nach verwendeter Datenbank, eine Vielzahl von Möglichkeiten. MySQL Datenbanken verwenden beispielsweise die Datentypen Point, Line oder Multipolygon. Die Datenhaltung von Geodaten in dem NRDBMS MongoDB erfolgt im GeoJSON-Format.

Im Rahmen des Performancemonitorings soll die Antwortzeit beim Auslesen einer großen Datenmenge aus einer MySQL- und einer MongoDB-Datenbank gemessen werden. Dabei wird sowohl der Single-User als auch den Multi-User Zugriff analysiert. Das Programm soll mit Java implementiert werden.

Ziel der Bachelorarbeit war es die theoretischen Grundlagen für das Erstellen des Performancemonitorings zu erarbeiten. Dieses Konzept wird in der Praxisarbeit realisiert, sodass der Vorteil eines NRDBMS beim Lesezugriff aufgezeigt werden. Mit dem Ergebnis der Performanceanalyse sollen die bayrischen Verwaltungen zu einem Umdenken in Sachen Datenbanktechnologien angeregt werden. Langfristig könnten dann auch hier NRDBMS eine echte Alternative zum klassischen RDBMS darstellen.

7 Literaturverzeichnis

7.1 Buch (Monographie)

Baun, Christian (2017): Betriebssysteme kompakt. Berlin: Springer Vieweg (IT kompakt).

Edlich, Stefan; Friedland, Achim; Hampe, Jens; Brückner, Markus; Brauer, Benjamin (2011): NoSQL. Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken. 2. aktual. u. erw. Aufl. München: Hanser.

Hollosi, Arno (2012): Von Geodaten bis NoSQL: leistungsstarke PHP-Anwendungen. Aktuelle Techniken und Methoden für Fortgeschrittene. München: Hanser.

Lampe, Frank (2010): Green-IT, Virtualisierung und Thin Clients. Mit neuen IT-Technologien Energieeffizienz erreichen, die Umwelt schonen und Kosten sparen. 1. Aufl. Wiesbaden: Vieweg+Teubner, zuletzt geprüft am 12.03.2019.

Mandl, Peter (2014): Grundkurs Betriebssysteme. Architekturen, Betriebsmittelverwaltung, Synchronisation, Prozesskommunikation, Virtualisierung. 4. Aufl. Wiesbaden: Springer Vieweg.

Meier, Andreas; Kaufmann, Michael (2016): SQL- & NoSQL-Datenbanken. 8., überarbeitete und erweiterte Auflage 2016. Berlin, Heidelberg: Springer Vieweg (eXamen.press).

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

Schukraft, Andrea; Lenz, Roman (2005): Geoinformationssysteme. Leitfaden zur Datenqualität für Planungsbüros und Behörden. 1. Aufl. München: Runder Tisch Geoinformationssysteme Techn. Univ, zuletzt geprüft am 07.03.2019.

Zimmermann, Albert (2012): Basismodelle der Geoinformatik. Strukturen, Algorithmen und Programmierbeispiele in Java ; mit 48 Tabellen und 42 Listings ; [im Internet: Java-Quelltexte der im Buch behandelten Programmbeispiele]. München: Hanser.

7.2 Zeitschriftenaufsatz

Chang, Fay; Dean, Jeffrey; Chemawat, Sanja; Hsieh, Wilson C.; Wallach, Deborah A.; Burrows, Mike et al. (2010): Bigtable: A Distributed Storage System for Structured Data, zuletzt geprüft am 08.03.2019.

Dean, Jeffrey; Ghemawat, Sanjay (2004): MapReduce: Simplified Data Processing on Large Clusters, zuletzt geprüft am 08.03.2019.

Gilbert, Seth; Lynch, Gilbert (2002): Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services, zuletzt geprüft am 08.03.2019.

Moniruzzaman, A B M; Hossain, Syed Akhter (2013): NoSQL Database: New Era of Databases for Big data Analytics -Classification, Characteristics and Comparison. In: *International Journal of Database Theory and Application* 6 (4), zuletzt geprüft am 09.03.2019.

7.3 Online-Quellen

Alachisoft (2019): Introduction to REST API in NoSQL Database - NosDB. Online verfügbar unter http://www.alachisoft.com/resources/articles/rest-api-nosql.html, zuletzt aktualisiert am 19.02.2019, zuletzt geprüft am 11.03.2019.

Bayerisches Staatsministerium der Finanzen und für Heimat: Vermessungsverwaltung. Online verfügbar unter https://www.stmflh.bayern.de/vermessung/, zuletzt geprüft am 07.03.2019.

Datenbank Lexikon. Key/Value-Datenbanksysteme, zuletzt geprüft am 09.03.2019.

Edlich, Stefan (2019): NOSQL Databases. Prof. Dr.Stefan Edlich. Online verfügbar unter http://nosql-database.org/, zuletzt aktualisiert am 07.01.2019, zuletzt geprüft am 07.03.2019.

Jansen, Rudolf (2011): NoSQL: Key-Value-Datenbank Redis im Überblick. Heise Medien. Online verfügbar unter https://www.heise.de/developer/artikel/NoSQL-Key-Value-Datenbank-Redis-im-Ueberblick-1233843.html, zuletzt aktualisiert am 29.04.2011, zuletzt geprüft am 09.03.2019.

Java SE Technologies - Database. Online verfügbar unter http://www.oracle.com/technetwork/java/javase/jdbc/index.html, zuletzt geprüft am 23.04.2018.

Landesamt für Statistik: Gebiet, Flächennutzung. Online verfügbar unter https://www.statistik.bayern.de/statistik/gebiet/, zuletzt geprüft am 07.03.2019.

Leite, Norberto (2016): How to Perform Random Queries on MongoDB. What's new in MongoDB 3.2. Online verfügbar unter https://www.mongodb.com/blog/post/how-to-perform-random-queries-on-mongodb, zuletzt aktualisiert am 16.02.2016, zuletzt geprüft am 11.03.2019.

Litzel, Nico; Tutanch (2017): Was ist RDBMS? Vogel Communications Group GmbH & Co. KG. Online verfügbar unter https://www.bigdata-insider.de/was-ist-rdbms-a-654230/, zuletzt aktualisiert am 18.10.2017, zuletzt geprüft am 07.03.2019.

Matzer, Michael; Litzel, Nico (2018): NoSQL-Datenbanken im Vergleich. Vogel Communications Group GmbH & Co. KG. Online verfügbar unter https://www.bigdata-insider.de/nosql-datenbanken-im-vergleich-a-666922/, zuletzt aktualisiert am 15.01.2018, zuletzt geprüft am 07.03.2019.

MySQL 8.0 Reference Manual. 11.5 Spatial Data Types. Online verfügbar unter https://dev.mysql.com/doc/refman/8.0/en/spatial-types.html, zuletzt geprüft am 07.03.2019.

MySQL 8.0 Reference Manual. 12.6.2 Mathematical Functions. Online verfügbar unter https://dev.mysql.com/doc/refman/8.0/en/mathematical-functions.html#function_rand, zuletzt geprüft am 11.03.2019.

MySQL: Warum MySQL? Online verfügbar unter https://www.mysql.com/de/why-mysql/?main=1&topic=12&type=22&lang=de, zuletzt geprüft am 07.03.2019.

Oracle Help Center (2005): Oracle and MySQL Compared. Oracle. Online verfügbar unter https://docs.oracle.com/cd/E12151_01/doc.150/e12155/oracle_mysql_compared.htm #CHDIIBJH, zuletzt geprüft am 08.03.2019.

Redis: Introduction to Redis. Online verfügbar unter https://redis.io/topics/introduction, zuletzt geprüft am 09.03.2019.

Staatsbetrieb Geobasisinformation und Vermessung Sachsen (Hg.) (2017): Grundlagen und Begriffe. Koordinatenreferenzsystem. Staatsbetrieb Geobasisinformation und Vermessung Sachsen (GeoSN). Online verfügbar unter http://www.landesvermessung.sachsen.de/inhalt/etrs/grund/grund.html, zuletzt aktualisiert am 29.06.2017, zuletzt geprüft am 12.03.2019.

Stadt Augsburg (2018): Augsburg in Kürze. Online verfügbar unter https://www.augsburg.de/buergerservice-rathaus/rathaus/statistik-stadtforschung/augsburg-in-kuerze/, zuletzt aktualisiert am 29.08.2018, zuletzt geprüft am 07.03.2019.

Stadt Neusäß: Zahlen und Fakten. Stadt Neusäß. Online verfügbar unter https://www.neusaess.de/de/Leben-in-Neus%C3%A4%C3%9F/Unsere-Stadt/Zahlen-und-Fakten, zuletzt geprüft am 07.03.2019.

Strozzi, Carlo (2010): NoSQL: a non-SQL RDBMS. NoSQL A Relational Database Management System. Online verfügbar unter http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page, zuletzt geprüft am 08.03.2019.

The Open Geospatial Consortium. Online verfügbar unter http://www.opengeospatial.org/, zuletzt geprüft am 13.03.2019.

Wikipedia (Hg.) (2018): Skalierbarkeit. Online verfügbar unter https://de.wikipedia.org/wiki/Skalierbarkeit, zuletzt aktualisiert am 07.12.2018, zuletzt geprüft am 08.03.2019.

Wikipedia (Hg.) (2019a): Gauß-Krüger-Koordinatensystem. Online verfügbar unter https://de.wikipedia.org/w/index.php?oldid=179750091, zuletzt aktualisiert am 24.02.2019, zuletzt geprüft am 12.03.2019.

Wikipedia (Hg.) (2019b): Koordinatensystem. Online verfügbar unter https://de.wikipedia.org/w/index.php?oldid=185978360, zuletzt aktualisiert am 24.02.2019, zuletzt geprüft am 12.03.2019.

Wyllie, Diego: MySQL vs. MongoDB: Datenbanksysteme im Vergleich. Online verfügbar unter https://www.computerwoche.de/a/mysql-vs-mongodb-datenbanksysteme-im-vergleich,1233517, zuletzt geprüft am 13.03.2019.

7.4 Vortrag

Brewer, Eric: Towards Robust Distributes Systems, zuletzt geprüft am 08.03.2019.

Fischbach, Kai (2019): Analyse sozialer Netzwerke. Representing and Measuring Networks. Bamberg, 29.01.2019, zuletzt geprüft am 10.03.2019.

8 Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde nach meiner besten Kenntnis bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Hof, den 13.03.2019 ________________________

(Andrea Reisenauer)