Datenbanken – Big Data

Ein Überblick über die Welt der großen Datenmengen

Facharbeit

Gymnasium Gerresheim, Q1

Schuljahr 2013/14

Fach: Informatik

Kurs: IF LK1

Betreuer/in: Herr Magata

vorgelegt von: Leo Bernard
40215 Düsseldorf
Q1, IF LK1
Schuljahr 2014/15

Abgabetermin: 07.03.2014

Inhaltsverzeichnis

1. Ein Überblick über die Welt der großen Datenmengen

Big Data ist ein seit 2010 in der Informatik sehr häufig verwendeter Begriff, speziell in Verbindung mit dem World Wide Web, welcher als Hype angesehen werden kann. Viele Unternehmen wie Google, Facebook und IBM werben mit “Big Data”, jedoch wird auf den ersten Blick nicht direkt klar, was damit genau gemeint ist, außer dass es sich bei diesem Begriff wohl um große Datenmengen handeln muss, da – wie bei den meisten Hype-Themen – oft nicht mehr subjektiv über das Thema berichtet wird.[1]

In dieser Facharbeit möchte ich veranschaulichen, was Big Data bedeutet, welche Vorteile durch die Verwendung von Big Data-Technologien in neuen Projekten entstehen, aber auch, welche Herausforderungen in diesem Zusammenhang bewältigt werden müssen.

Nach der Definition des Begriffes “Big Data” werde ich Anwendungsfälle anhand zweier Projekt-Beispiele aus der Vergangenheit und der Gegenwart zeigen und erläutern, inwiefern diese als Big Data-Projekte angesehen werden können.

Ich werde den Fokus dieser Facharbeit auf die Datenbanksysteme legen, welche zur Speicherung der großen Datenmengen verwendet werden können. Im Hinblick auf die Menge der frei verfügbaren Datenbanksysteme möchte ich zeigen, wie sich diese am besten unterteilen lassen und welches System für welches Big Data-Projekte am besten geeignet ist. Dafür werde ich ein Programm schreiben, welches die beiden beliebtesten Datenbanksysteme MySQL und MongoDB automatisiert hinsichtlich der Effizienz bei den beiden wichtigsten Operationen – das Einfügen und Auslesen von vielen Daten – testet und die Ergebnisse auswertet.

Zur Verdeutlichung der diesem neuen Begriff zugrundeliegenden Datenbanktechnologien werde ich einen Web-Crawler schreiben, in

[1] vgl. BRÜCHER, 2013

welchem ich alle grundlegenden Aspekte von Big Data mit einbeziehen werde und welcher die Datenstrukturen der neueren Datenbanksysteme verwenden wird.

Beide Experimente werde ich dieser Facharbeit in einer virtuellen Maschine beilegen, sodass der Leser sie selbst erneut ausführen und auswerten kann. Den Quelltext dieser Experimente werde ich unter der GPL-Lizenz online bereitstellen (siehe Anhang 2).

2. Definition des Begriffes „Big Data“

„Big Data“ – Der Begriff selbst ist nur ungenau definiert. Es gibt drei relative Kriterien, von denen mindestens ein Kriterium erfüllt werden muss, damit Daten als „Big Data“ angesehen werden können: Die Datenmenge (Volume), die Geschwindigkeit der Daten (Velocity) und die Vielfalt der Daten (Variety).[2] Diese drei Kriterien werden auch als die drei „V“ bezeichnet.

2.1. Die Datenmenge (Volume)

Unter dem Begriff „Volume“ versteht man, welche Menge an Daten in einer Datenbank gespeichert wird. Werden in einer Datenbank viele Daten gespeichert – genauer ist dies nicht definiert –, so ist dieses Kriterium erfüllt. Werden z.B. in einer Datenbank 1.000.000 Datensätze gespeichert, so könnte man bereits sagen, dass das Kriterium erfüllt ist. Vergleicht man die Menge der Datensätze jedoch beispielsweise mit den knapp 845 Mio. aktiven Benutzern, die bei Facebook registriert sind[3], sprich den mindestens 845 Mio. Datensätzen die daher bei Facebook gespeichert werden müssen, so scheinen unsere eine Millionen Datensätze nicht mehr allzu viel zu sein. Was unter „viel“ verstanden werden kann sei also der Interpretation des Lesers

[2] vgl. HORVATH, 2013
[3] vgl. FACEBOOK, Inc.

überlassen, Menge ist in diesem Fall relativ und hängt von der Ausgangssituation ab.[4]

2.2. Die Geschwindigkeit der Daten (Velocity)

Der Begriff „Velocity“ beschreibt die Geschwindigkeit in welcher Daten im Server eintreffen und verarbeitet werden müssen. Treffen Daten im Server so schnell ein, dass er mit der Verarbeitung dieser in Echtzeit an seine Grenzen kommt, ist dieses Kriterium erfüllt. Jedoch existiert auch hier keine genaue Definition, ab welcher Datengeschwindigkeit dieses Kriterium erfüllt ist. Ein handelsüblicher Computer wäre natürlich nicht dazu in der Lage, Daten in der selben Geschwindigkeit zu verarbeiten wie beispielsweise ein Supercomputer oder ein Computer-Cluster. Um auf das Beispiel Facebook zurückzugreifen, lassen sich hier die abgegebenen Likes als Beispiel nehmen. Täglich werden von den registrierten Benutzern ca. 1,2 Mrd. Likes abgegeben. [5] Das sind durchschnittlich ca. 13.889 Likes pro Sekunde, die in den Datencentern abgespeichert und verarbeitet werden müssen. Mit dieser Anzahl von Anfragen pro Sekunde wäre ein handelsüblicher Computer maßlos überlastet. Für die Computer, die bei Facebook, Inc. die Anfragen übernehmen ist diese Anzahl jedoch nicht besonders groß. Auch hier kommt es wieder auf den Anwendungsfall an, ab wann das Kriterium erfüllt ist.

2.3. Die Vielfalt der Daten (Variety)

Bei Big Data kommt es nicht unbedingt nur auf die Größe oder Geschwindigkeit der Daten an. Die Vielfalt der Quellen aus denen Daten gesammelt und verarbeitet werden und die Vielfalt der Datenformate selbst, spielt ebenfalls eine wichtige Rolle und bedingt das Kriterium „Variety“.[6] Nehmen wir als Beispiel wieder Facebook. Benutzer können hier Daten in Form von Text (Posts) speichern, jedoch auch Fotos und Videos von ihrem Smartphone aus hochladen, Links zu

[4] vgl. BRÜCHER, 2013, S. 42
[5] vgl. BRANCKAUTE, 2012
[6] vgl. DUMBILL, 2013

Websites teilen und ihren aktuellen Standort veröffentlichen. Zusätzlich dazu können sie sich mit anderen Nutzern über den integrierten Chat austauschen.[7] Alle diese Daten müssen von den Servern verarbeitet und gespeichert werden, haben jedoch unterschiedliche Formate. Die Schwierigkeit besteht nun darin, die Daten so anzupassen, dass sie strukturiert und für die spätere Verarbeitung und Analyse geeignet auf den Servern gespeichert werden können.[8]

3. Anwendungsbeispiele für Big Data

Big Data Projekte gibt es viele – sowohl online als auch offline. Im Folgenden möchte ich zwei dieser Projekte vorstellen.

3.1. Anwendungsbeispiel aus der Vergangenheit: Die amerikanische Volkszählung im Jahr 1890

Eines der ersten Projekte, die man bereits als Big Data Projekt bezeichnen kann, wurde im Jahr 1890 geplant und ausgeführt: Die amerikanische Volkszählung sollte das erste Mal durch Maschinen unterstützt und beschleunigt werden. Die Volkszählung aus dem Jahr 1880 sollte knapp 8 Jahre lang dauern, sodass Herman Hollerith, ein amerikanischer Ingenieur, versuchte, einen Weg zu finden, mit dem sich die Daten der Volkszählung einfacher verarbeiten lassen würden.[9]

Seine Idee bestand darin, die Daten in Form von Lochkarten zu erfassen, die sich durch seine Maschine später auslesen lassen würden. Der Aufbau dieser Maschine ist sowohl einfach als auch genial. Die Lochkarten werden durch kleine Metallköpfe ausgelesen. Passt ein Metallkopf durch ein Loch, so kommt er mit dem unter der Lochkarte liegenden Gegenstück in Kontakt, sodass der Strom an dieser Stelle

[7] vgl. CHANEY, 2012
[8] vgl. BRÜCHER, 2013
[9] vgl. BRUNO

fließen kann.[10] Dadurch wird ein Zählmechanismus aktiviert und die entsprechende Zahl um 1 erhöht. [11]

Mithilfe dieser Maschine konnte die Auswertung der Fragebögen von ca. 63 Millionen Bürgern, die jeweils Daten zu 55 unterschiedlichen Fragen beinhalteten[12], in knapp zwei Jahren fertiggestellt werden. Auch wenn die Kriterien „Velocity“ und „Variety“ nicht erfüllt sind, kann dieses Projekt alleine durch das riesige Datenvolumen als Big Data Projekt angesehen werden.

3.2. Anwendungsbeispiel aus der Gegenwart: Google

Google ist die weltweit mit Abstand am meisten genutzte Suchmaschine.[13] Angefangen als ein Studentenprojekt namens *BackRub* im Jahr 1996 [14] enthält die von Google verwendete und selbst entwickelte Datenbank *BigTable* [15] mittlerweile über eine Billionen Links zu unterschiedlichen Webseiten[16]. Da Googles Datenbank durchgängig durch Googles eigenen Crawler *Googlebot* aktualisiert wird, welcher mithilfe „eine[r] gewaltige[n] Anzahl von Computern“[17] täglich mehrere Milliarden Websites indiziert, ist das Kriterium „Velocity“ erfüllt.

Neben der ursprünglichen Suchmaschine bietet Google heute zusätzlich viele weitere Services, wie z.B. das Videoportal *YouTube*, das soziale Netzwerk *Google+*, den E-Mail Service *Google Mail*, den Streaming-Dienst *Google Music* und das Werbenetzwerk *AdWords* an, welche ähnlich große Datenmengen wie die Suchmaschine selbst erzeugen. So werden auf YouTube beispielsweise jede Minute ca. 100 Stunden an neuem Videomaterial hochgeladen, die verarbeitet und dem Index hinzugefügt werden müssen.[18]

---

[10] vgl. AMERICAN MEMORY HOME, 1895
[11] vgl. BRUNO
[12] vgl. ANCHESTRY, 1890
[13] vgl. STATISTA, 2013
[14] vgl. GOOGLE, INC.
[15] vgl. CHANG, DEAN, GHEMAWAT, & HSIEH, 2006
[16] vgl. GOOGLE, INC., 2008
[17] vgl. GOOGLE, INC.
[18] vgl. YOUTUBE

Zusätzlich dazu entwickelt Google aktiv das auf vielen Smartphones verwendete Betriebssystem *Android,* welches über den Google Account eines Users alle wichtigen Daten mit den Google Services synchronisieren kann.[19]

Die Kriterien „Volume“ und „Variety“ sind bei bei diesem Unternehmen durch die hohe Anzahl an unterschiedlichen Services, welche jeweils unterschiedliche Daten erzeugen und die Größe der Datenbanken, ebenfalls erfüllt.

Durch die riesigen Mengen an Daten, die Google durch ihre eigenen Services zur Verfügung stehen, wäre es für das Unternehmen theoretisch möglich, von den meisten Benutzern ein vollständiges Profil zu erstellen. Hier stößt Google teilweise auf laute Kritik. Auch wenn Googles Motto „Don’t be evil“ lautet, können diese Daten, geraten sie in falsche Hände, sehr wohl zu „bösen“ Zwecken eingesetzt werden. Durch die veröffentlichten Dokumente über die NSA und deren Kooperationsprogramm *Prism*,[20] durch das unter Anderem Google sehr viel Geld erhalten hat,[21] um den Datenverkehr seiner Benutzer der NSA zugänglich zu machen, bestätigte sich diese Kritik in den letzten Monaten bereits.

4. Nutzen und Herausforderungen von Big Data

Angenommen, ein Unternehmen hat sich dazu entschieden, ein Big Data-Projekt zu starten. Worin liegt der Nutzen dieser erzeugten Datenmengen?

Nutzt man die Vorteile der großen Datenmengen als Unternehmen zielgerichtet, so können durch Auswertungen und Verknüpfungen dieser Daten die Risiken bei Entscheidungen minimiert und Prozesse optimiert werden.[22] So kann ein Unternehmen, das Software entwickelt, beispielsweise gesammelte Nutzungsstatistiken dazu verwenden, um

[19] vgl. MAD SKILLS GMBH
[20] vgl. ERNST, GREIS, & THOMA, 2013
[21] vgl. PAKALSKI, 2013
[22] vgl. BRÜCHER, 2013

herauszufinden, mit welchen Bereichen der Software die Benutzer noch Schwierigkeiten haben, und diese gezielt optimieren.

Ein Unternehmen, das in Windräder investiert, hat z.B. durch gesammelte Wetterdaten die Möglichkeit, diese so zu platzieren, dass Sie den Wind so effizient wie möglich in Strom umwandeln. Zudem kann mithilfe dieser Daten auch im Voraus berechnet werden, wie viel Energie ein gegebenes Windrad in einem Zeitraum liefern wird, sodass eingesehen werden kann, ob Gebiete in Zukunft ausreichend mit Strom versorgt werden können.[23]

Ein Unternehmen, das Produkte verkauft, hat die Möglichkeit, anhand der gesammelten und analysierten Testberichte von Käufern, sowie von Daten aus Social Media Netzwerken seine Marketingstrategie anzupassen. Hier können beispielsweise auch gezielt einzelne Käufer angesprochen werden. Anhand des Konsumverhaltens der Kunden ist es – vorausgesetzt, das Unternehmen hat genügend Daten in diesem Bereich gesammelt – möglich, vorauszuberechnen, welche Produkte dieser Kunde als nächstes Kaufen wollen könnte und ihm dementsprechend Werbung zu schicken.[24] Dies kann, wenn das System gut entwickelt ist, komplett automatisiert geschehen.

Laut einer Analyse des Harvard Business Review arbeiten Unternehmen, welche Big Data einsetzen, um Entscheidungen zu treffen, ca. 5 Prozent produktiver und ca. 6 Prozent profitabler.[25]

Auch wenn der Nutzen von Big Data-Projekten klar sichtbar ist, stellt die Verwendung der neuen Technologien für die Unternehmen eine neue Herausforderung dar. Wie in der Definition des Begriffes „Big Data“ bereits erwähnt, muss die Software und Hardware zur Verwendung in Big Data-Projekten dazu in der Lage sein, eine große Menge an Daten in einer möglichst geringen Zeit zu speichern und zu verarbeiten. Frühere Systeme basierten meist auf wenigen, sehr leistungsstarken Servern, die diese Anfragen verarbeiten mussten. Mit der steigenden

[23] vgl. BRÜCHER, 2013
[24] vgl. RODDEN
[25] vgl. BRÜCHER, 2013

Datenmenge wurde diese Methode jedoch zu kostenintensiv. Die Verarbeitung und Speicherung der Daten sollte daher eher auf viele Server verteilt werden. Mit einem geeigneten System ließen sich sowohl die Kosten minimieren, als auch die Skalierbarkeit der Projekte sicherstellen.[26]

Bei großen Anbietern wie z.B. Amazon lassen sich Server – bezahlt pro Stunde – mieten, sodass sie je nach Bedarf zu dem System hinzugefügt werden können.[27]

Eine beliebte Lösung für diesen Ansatz ist das open-source Projekt *Hadoop*, welches von der *Apache Foundation* entwickelt wird. Mithilfe dieser Softwarelösung lassen sich Aufträge, wie beispielsweise die Verarbeitung von Datensätzen in einer Datenbank effizient auf beliebig viele Server verteilen.[28]

Zudem ist es wichtig, nur die Daten zu sammeln, welche auch wirklich später benötigt werden. Diese müssen in den Datenbanken so effizient strukturiert wie möglich gespeichert werden, sodass sie später gut weiterverarbeitet werden können.

Nun ist es nötig, den Nutzen dem Aufwand, welcher durch die Herausforderungen bei Big Data Projekten entsteht, entgegenzustellen um zu entscheiden, ob es sich lohnt, ein Big Data-Projekt zu starten.

5. Geeignete Datenbanksysteme

Um für Big Data Projekte geeignet zu sein, sollte ein Datenbanksystem dazu in der Lage sein, mit den drei bereits erwähnten Kriterien umgehen zu können. Es sollten also dazu in der Lage sein, eine große Menge an Daten so zu speichern, dass sie schnell wieder abgerufen und verarbeitet werden kann. Zudem sollten die Systeme dazu in der Lage sein, auf mehrere Server verteilt werden zu können.

---

[26] vgl. SPICHALE & WOLFF, 2013
[27] vgl. AMAZON
[28] vgl. APACHE HADOOP, 2014

Datenbanksysteme lassen sich grob in zwei Kategorien unterteilen: Relational und nicht-relational.

5.1. Relationale Datenbanksysteme (MySQL)

In relationalen Datenbanken werden die Daten in zweidimensionalen Datenstrukturen, sprich in Tabellen gespeichert. Die Spalten, oder auch Attribute, der Tabelle geben feste Eigenschaften des Datensatzes an, eine Zeile, auch Tupel genannt, repräsentiert einen Datensatz. Die erste Spalte in jeder Tabelle speichert in den meisten Projekten einen eindeutigen Index, wie z.B. eine fortlaufende Zahl. Diese Spalte wird meistens als „ID“ oder „id“ beschriftet.[29] Eine Tabelle für registrierte Kunden könnte also die Spalten ID, Name, Vorname, Mail und Passwort beinhalten (siehe Anhang 3.1).

Jedes Attribut besitzt einen festen Datentyp, wie beispielweise einen Integer oder einen String und bei den meisten Datentypen auch eine obere Grenze für die Größe. Das Attribut Name der oben gezeigten Beispieltabelle könnte z.B. als String mit einer maximalen Größe von 50 Zeichen definiert werden.

Die relationale Datenstruktur hat mehrere Vorteile. Durch die feste Struktur der Tupel kann ein Programm wissen, mit welchen Daten es arbeiten wird. Die Datensätze können zudem leicht nach einem bestimmten Attribut sortiert und gefiltert werden.

Viele relationale Datenbanken unterstützen das Erstellen von Indizes über eine oder mehrere Spalten. Angenommen, ein Programm muss häufig Benutzer-Tupel anhand des Nachnamens abfragen, so kann auf das Attribut „Name“ ein Index gesetzt werden. Ohne den Index würde das Datenbanksystem jeden Eintrag in der Tabelle durchsuchen und die Einträge, die mit der Abfrage übereinstimmen, zurückgeben. Die Komplexität der Abfrage läge im schlimmsten Falle hier bei $O(n)$ (Siehe Anhang 3.3, Violett).[30] Bei kleinen Projekten ist das kein Problem, beinhaltet die Tabelle jedoch eine sehr große Anzahl an Datensätzen,

[29] vgl. KERSKEN, 2007
[30] vgl. KERSKEN, 2007

wird diese Abfrage unter Umständen zu lange dauern. Durch den Index auf dem Attribut legt die Datenbank intern einen binären Suchbaum, auch BinarySearchTree oder BTREE genannt, über die Daten an. Die Komplexität der Abfragen, die nur auf dieses Attribut zugreifen, liegt damit auf $O(\log_2 n)$ (Anhang 3.3, Blau).

Zwei häufig genutzte relationale Datenbanksysteme sind MySQL von Orcale und PostegreSQL von der PostgreSQL Global Development Group. Sie verwenden zur Abfrage der Daten die Sprache SQL. Um beispielsweise alle Benutzer, deren Nachname mit B anfängt, nach Nachname sortiert abzurufen, lässt sich der folgende Code verwenden:

```
SELECT * FROM Kunden WHERE Nachname LIKE 'B%' ORDER BY Nachname ASC
```

Die Nachteile der relationalen Datenbanksysteme liegen wieder in der festen Struktur der Tabellen. Ein Beispiel, warum dies ein Nachteil sein kann, sind Zuordnungen. Angenommen, einem Kunden sind mehrere Bestellungen zugeordnet, welche wieder eigene Attribute besitzen, so muss für diese Bestellungen eine weitere Tabelle angelegt werden, die ein Attribut enthält, in dem die ID des Kunden gespeichert wird. So kann eine *1-zu-n* Beziehung hergestellt werden.

Soll diese Bestellung nun aus mehreren Produkten bestehen, welche ebenfalls jeweils eigene Attribute besitzen, so müssen zwei weitere Tabellen angelegt werden: Eine Tabelle, die alle Produkte mit ihren Attributen enthält und eine Tabelle, welche die Zuordnung der Produkte zu den Bestellungen enthält. So kann eine *n-zu-n* Beziehung hergestellt werden, sprich beliebig viele Produkte können beliebig vielen Bestellungen zugeordnet werden. Die fertigen Tabellen mit ihren Zuordnungen könnten so aussehen, wie die Abbildung in Anhang 3.2.

Ein weiteres Problem an den festen, tabellenartigen Strukturen besteht darin, dass sich hierarchische Strukturen nicht ohne weiteres speichern lassen. Sollen Beispielsweise Produktkategorien mit jeweils beliebig vielen Unterkategorien gespeichert werden, so müsste eine Tabelle für die Produktkategorien angelegt werden, die ein Attribut enthält, welches jeweils die ID der übergeordneten Kategorie enthält. Nun weiß die

Datenbank zwar, welche Kategorie einer anderen Kategorie übergeordnet ist, um aus diesen Daten jedoch eine echte hierarchische Struktur zu erstellen, müssen sie außerhalb der Datenbank noch einmal traversiert und dadurch in eine entsprechende Struktur umgewandelt werden. Je nachdem wie viele Kategorien es gibt, kann dies lange dauern. Probleme können zusätzlich auftreten, wenn z.B. die Wurzel der hierarchischen Struktur nicht mehr existiert.[31]

5.2. Nicht-Relationale Datenbanksysteme (MongoDB)

Bei nicht-relationalen Datenbanksystemen ist die Struktur der zu speichernden Daten nicht festgelegt. Beispiele von nicht-relationalen Datenbanksystemen sind Systeme, in denen Daten in Form von Graphen oder JSON- bzw. BSON-Dokumenten, gespeichert werden, auf welche ich später weiter eingehen werde. Beispiele dafür sind *GraphDB* und *MongoDB*. Eine Unterkategorie von nicht-relationalen Datenbanksystemen sind sogenannte NoSQL-Datenbanken. Diese Bezeichnung stammt ursprünglich von einer Bewegung gegen ältere Datenbanksysteme, welche auf die Anfragesprache SQL setzen. Ein großes Merkmal vieler NoSQL-Systeme ist der Versuch, die Verteilung der Daten auf viele Server so effizient und stabil wie möglich zu gestalten, wie in dem Kapitel „Herausforderungen von Big Data“ bereits erwähnt.

Es gibt jedoch auch noch NoSQL-Datenbanksysteme, in denen die Daten, fast wie bei relationalen Datenbanken, in Form von Tabellen gespeichert werden.[32] Ein Beispiel dafür ist das System *Hypertable*, welches auf Googles BigTable Datenstruktur basiert.[33] Der Unterschied zu relationalen Datenstrukturen besteht hier darin, dass jede Zeile zusätzlich zu den im Vorhinein festgelegten Attributen beliebig viele zusätzliche Attribute besitzen kann.[34]

---

[31] vgl. HILLYER, 2012
[32] vgl. SPICHALE & WOLFF, 2013
[33] vgl. HYPERTABLE, INC.
[34] vgl. SPICHALE & WOLFF, 2013

Da das Spektrum der nicht-relationalen Datenbanksysteme die Kapazitäten der Facharbeit überschreiten würde, werde mich hier mit einem der beliebtesten Systeme, MongoDB, auseinandersetzen.

MongoDB ist ein dokumenten-orientiertes open-source Datenbanksystem welches Daten in Form von BSON-Dokumenten speichert.

JSON ist ein Datenformat, welches dazu entwickelt wurde, Daten in einem für Menschen gut lesbaren und für Programme schnell zu verarbeitenden Format abzuspeichern. Das Format basiert in seinen Grundzügen auf der Sprache JavaScript, nutzt jedoch nur einen Teil der verfügbaren Syntax. Die Daten werden in Form von Key-Value Paaren gespeichert, davon können in einem JSON Dokument beliebig viele existieren. Der Key wird durch einen String, wie z.B. ”name“ dargestellt, der Wert kann ein Integer, String, Double, Boolean, aber auch ein Array oder wiederum ein anderes JSON-Dokument sein.[35] Die Datentypen müssen im Gegensatz zu denen in relationalen Datenbanken nicht im Vorhinein festgelegt werden. Arrays werden in eckigen Klammern gespeichert, Objekte, bzw. Dokumente, werden mit geschweiften Klammern umfasst. Einträge, sowie Attribute werden durch Kommata voneinander getrennt. [36]

BSON, oder auch Binary JSON, ist ein von MongoDB entwickeltes Datenformat, welches auf JSON aufbaut, jedoch weitere Datentypen integriert und die Daten in einem von Menschen nicht mehr lesbaren, binären Format speichert, welches von Programmen schneller verarbeitet werden kann als das JSON-Format.[37] Neue Datentypen sind beispielsweise verschiedene Datumsformate, binäre Daten, reguläre Ausdrücke und JavaScript Programmcode.[38] Keys in BSON können im Zusammenhang mit MongoDB auch als Attribute der Dokumente bezeichnet werden.

---

[35] vgl. ECMA INTERNATIONAL, 2013
[36] vgl. MONGODB, INC., 2013
[37] vgl. MONGODB, INC.
[38] vgl. MONGODB, INC.

Jedes Dokument in MongoDB erhält beim Speichern automatisch das Attribut „_id“, welches einen automatisch generierten Wert enthält, mit dem sich das jeweilige Dokument eindeutig identifizieren lässt. Dieser Wert wird bereits vor dem Speichern des Dokumentes vom Client generiert und enthält einen String, welcher sich aus dem aktuellen Zeitstempel, der ID des Computers, der ID des Prozesses und einem Zähler zusammensetzt, sodass die Einzigartigkeit sichergestellt wird, auch wenn mehrere Clients gleichzeitig Dokumente in die Datenbank einfügen.

Die BSON-Dokumente werden in MongoDB in sogenannten Collections gespeichert, eine Datenbank kann beliebig viele Collections enthalten. Collections in MongoDB sind also vergleichbar mit Tabellen in relationalen Datenbanken.[39] So könnte die Datenbank für einen online Shop die Collections „Produkte“, „Kategorien“ und „Kunden“ enthalten.

Ein BSON-Dokument für einen Kunden könnte beispielsweise, wie in dem Beispiel für relationale Datenbanken, die Attribute „Name“, „Vornamen“, „Mails“ und „Passwort“ beinhalten. Die Felder „Vornamen“ und „Mails“ können jeweils ein Array enthalten. So kann ein Kunde beliebig viele Vornamen und E-Mail Adressen eintragen. Eine E-Mail Adresse kann auch ein Objekt mit mehreren Attributen sein. So kann jede E-Mail Adresse eine Priorität enthalten, oder das Attribut, ob Newsletter an diese Adresse verschickt werden soll (siehe Anhang 3.4).

Durch den Aufbau der Dokumente lassen sich auch sehr einfach hierarchische Strukturen speichern. So könnte beispielsweise eine Liste der Kategorien als eine Collection von Hauptkategorien aufgebaut werden, wobei jede Kategorie das Attribut „Unterkategorien“ besitzt, welches ein Array mit den Unterkategorien darstellt. Löscht man nun eine Hauptkategorie, werden alle Unterkategorien entsprechend mitgelöscht. Auch die Auflistung der Kategorien ist auf diese Art einfacher, da es möglich ist, alle Kategorien inklusive Unterkategorien mithilfe von nur einem Befehl abzurufen, was bei relationalen

[39] vgl. MONGODB, INC., 2013

Datenbanksystemen aufgrund der Struktur nicht ohne Weiteres möglich ist.

Auch bei Dokumenten ist es möglich, Referenzen auf ein anderes Dokument zu erstellen. Angenommen, jedem Kunden sollen, wie in dem Beispiel für relationale Datenbanken, Bestellungen zugewiesen werden, so kann jeder Kunde ein Attribut „Bestellungen“ erhalten, welches die ID der Bestellungen enthält. MongoDB wird in diesem Fall automatisch erkennen, dass es sich hierbei um eine Referenz handelt, und die ID durch das entsprechende Objekt ersetzen.[40] So lässt sich eine 1-zu-n, aber auch eine n-zu-n Verbindung schnell und einfach realisieren.

Es ist – wie z.B. bei MySQL – möglich, in MongoDB Indizes über bestimmte Attribute anzulegen, welche man oft in Abfragen benötigt, um die Geschwindigkeit zu optimieren. Auch hier wird intern ein binärer Suchbaum erstellt, was bereits für die relationalen Datenbanken genauer erläutert wurde. Standardmäßig liegt ein Index auf dem Attribut „_id“. [41]

MongoDB wird mit einer an JavaScript angelehnten Sprache gesteuert. Will man, wie bei MySQL, also alle Kunden, sortiert nach dem Nachnamen, abrufen, deren Nachname mit „B“ anfängt, so lässt sich der folgende Aufruf verwenden:

```
db.kunden.find({"name": /B.*/}).sort({"name": 1})
```

Ein großer Vorteil von MongoDB im Hinblick auf Big Data ist die Möglichkeit, die Datenbank auf mehrere Server zu replizieren. Dafür wird ein Server als „Primary“ deklariert. Dieser Server nimmt alle Anfragen an und verarbeitet diese. Wenn sich an der Datenbank nach der Anfrage etwas geändert hat, so werden diese Änderungen auf die Datenbanken der anderen Server übertragen. Dadurch, dass nur der primäre Server Anfragen verarbeitet, wird sichergestellt, dass es keine Konflikte zwischen den Datensätzen gibt. Fällt der primäre Server nun

[40] vgl. MONGODB, INC., 2013
[41] vgl. MONGODB, INC., 2013

aus – dies ist für MongoDB der Fall, wenn er länger als 10 Sekunden nicht mit den anderen Servern kommuniziert – so wird einer der sekundären Server als neuer primärer Server ausgewählt, der nun die Anfragen verarbeitet. Die Zeitspanne, für die ein solches System nicht erreichbar ist, liegt also bei maximal 10 Sekunden.[42] Das Risiko eines Datenverlustes wird durch dieses System extrem minimiert, vor allem, wenn sich die unterschiedlichen Server an verschiedenen Orten auf der Welt befinden. Diese Replikation ist bei wenigen relationalen Datenbanken verfügbar. Bei MySQL ist so ein System beispielsweise nur mithilfe einer alternativen Version, MySQL Cluster, möglich.[43]

5.3. Leistungsvergleich zwischen MySQL und MongoDB

Um die Leistung der beiden beliebten Datenbanken MySQL und MongoDB miteinander zu vergleichen, habe ich ein abgeschirmtes Testszenario aufgebaut. Die beiden Datenbanksysteme laufen in einer virtuellen Maschine (VirtualBox) und werden über ein selbstgeschriebenes Python-Script automatisiert abgefragt.

Zuerst werden beide Systeme komplett geleert. Dadurch bleiben keine Überreste von vorherigen Tests übrig, die das Ergebnis verfälschen könnten. Dann wird eine Liste mit einer gegebenen Anzahl von zufälligen Strings mit einer Länge von jeweils 100 Zeichen generiert.

Diese Liste wird nun sowohl in MySQL, als auch in MongoDB eingefügt, wobei für jeden Eintrag eine neue Anfrage an die Datenbank gestellt wird. Dies soll die Leistung unter einer hohen Anfragefrequenz testen.

Anschließend werden alle hinzugefügten Einträge nacheinander, anhand des zufälligen Strings, in beiden Datenbanken gesucht – bei dem ersten Durchlauf, ohne dass ein Index auf dem String-Attribut liegt, bei dem zweiten Durchlauf mit einem Index auf diesem Attribut. Damit kann festgestellt werden, wie gut beide Datenbanksysteme dazu fähig sind, Einträge zurückzuliefern, auch wenn die Frequenz der Anfragen

[42] vgl. MONGODB, INC., 2013
[43] vgl. ORACLE

sehr hoch ist. Zusätzlich werden in diesem Test Anfragen, bei denen auf dem abgefragten Attribut schon ein Index liegt den Anfragen gegenübergestellt, bei denen dieser Index noch nicht existiert. Alle Tests werden danach zusätzlich mit einer neuen Verbindung pro Abfrage, also mithilfe von Threads, durchgeführt, sodass die Geschwindigkeit der Datenbanken unter hoher Auslastung noch besser getestet werden kann.

Diese Tests habe ich mit 1000, 5000, 10000 und 100000 Test-Datensätzen durchgeführt, alle Zeiten wurden in Mikrosekunden gemessen, um kleinste Unterschiede besser feststellen zu können.

Bereits bei 1000 Test-Datensätzen ließen sich deutliche Unterschiede zwischen MySQL und MongoDB feststellen. Während das Einfügen der Datensätze in MySQL ca. 1362 µs pro Datensatz gedauert hat, benötigte MongoDB nur ca. 606 µs pro Datensatz.[44] Dies kann gut daran liegen, dass MongoDB die Datensätze unabhängig von der Anfrage in der Datenbank speichert, der Client also nicht mehr auf eine Antwort warten muss.[45] Dies ist bei MySQL nicht der Fall – hier muss der Client warten, bis die Daten vollständig gespeichert wurden. Dies dauert zwar länger, hat aber den Vorteil, dass Fehler beim Einfügen der Daten sofort erkannt und behandelt werden können.

Bei den Abfragen der eingefügten Einträge aus der Datenbank werde ich mich auf die Ergebnisse bei 100000 Test-Datensätzen beziehen, denn hier waren die Unterschiede – zum einen zwischen der Leistung der beiden Datenbanken, aber auch zwischen Abfragen, bei denen auf der abgefragten Spalte bereits ein Index lag und denen ohne Index auf der abgefragten Spalte – deutlich sichtbar. So dauerte die Abfrage aller Einträge aus MySQL ohne Index ca. 1626 Sekunden, sprich ca. 27 Minuten, während sie bei den Einträgen aus MongoDB ohne Index ca. 2249 Sekunden, bzw. ca. 37 Minuten dauerte.[46] Hier ist MySQL also effektiver als MongoDB. Sieht man sich die Graphen der Abfragedauer

[44] Siehe Anhang 1 (results-1000/end_result.csv)
[45] vgl. MONGODB, INC., 2013
[46] Siehe Anhang 1 (results-100000/end_result.csv)

pro Eintrag an, so erkennt man, abgesehen von den kleinen Schwankungen, einen linearen Anstieg der Abfragedauer.[47] Dies lässt sich dadurch erklären, dass beide Datenbanken jeden Eintrag auf die in der Abfrage gegebene Bedingung überprüfen müssen, jedoch explizit nur einen Eintrag zurückgeben sollen. So wird also, wenn der erste Datensatz bereits mit den Bedingungen übereinstimmt, die Abfragedauer minimal sein, während sie maximal sein wird, wenn erst der letzte Datensatz auf die Bedingungen zutrifft.

Die selbe Abfrage, dieses Mal jedoch mit einem Index auf der jeweils abgefragten Spalte, dauerte bei MySQL insgesamt ca. 20 Sekunden, was knapp 80 Mal so schnell ist, wie bei der Abfrage ohne Index.[48] Auch bei MongoDB dauerte die Abfrage insgesamt nur ca. 32 Sekunden, was ungefähr 71 Mal so schnell ist wie die Abfrage ohne Index.[49] Hier wird bereits der Vorteil der Indizes deutlich. Sieht man sich hier die Graphen der Abfragedauer pro Eintrag an, so fällt auf, dass sich die Dauer bei den ersten abgefragten Einträgen nicht sonderlich von der Dauer bei den letzten abgefragten Einträgen unterscheidet.[50] Dies liegt daran, dass die Datensätze durch das Anlegen eines Indexes in einem binären Suchbaum gespeichert werden, was ich in dem Beispiel zu MySQL bereits genauer erläutert habe.

Zusammengefasst lässt sich sagen, dass sich MySQL in diesem Leistungstest als leistungsstärker herausgestellt hat. Da dieser Test jedoch nur die grundlegenden Operationen, Schreiben und Lesen von Datensätzen, verglichen hat, lässt sich mithilfe dieses Ergebnisses nicht absolut behaupten, dass MySQL für jedes Projekt die bessere Wahl ist. Hier kommt es auf viele unterschiedliche Faktoren an. Diese Vor- und Nachteile habe ich bereits in den Beschreibungen der beiden Datenbanken näher erörtert.

---

[47] Siehe Anhang 1 (results-100000/mysql_get_single.png und results-100000/mongodb_get_single.png)
[48] Siehe Anhang 1 (results-100000/end_result.csv)
[49] Siehe Anhang 1 (results-100000/end_result.csv)
[50] Siehe Anhang 1 (results-100000/mysql_get_single_index.png und results-100000/mongodb_get_single_index.png)

6. Fazit

Big Data-Projekte eröffnen Unternehmen viele neue Möglichkeiten im Bereich der Kostenoptimierung, sei es durch die gezielte Anpassung von Werbung an Kunden, durch die Optimierung von Standorten für die beste Erreichbarkeit oder durch die Minimierung des Risikos bei Investitionen durch Vorhersagen der Bedingungen.

Durch das stetig wachsende Bedürfnis, große Datenmengen so speichern zu können, dass sie schnell abgerufen und effizient weiterverarbeitet werden können, entstehen viele neue Datenbanksysteme, wie beispielsweise MongoDB und MySQL Cluster.

Auch die Methoden zur Erhöhung der Leistung von Servern haben sich in den letzten Jahren von dem Prinzip, eher wenige, leistungsstarke Server einzusetzen, zu dem Prinzip, die Last auf viele, schwächere Server zu verteilen, geändert. Dadurch entstanden neue Systeme, wie z.B. Apache Hadoop.

Welches der vielen Datenbanksysteme für ein spezifisches Projekt am besten eingesetzt werden soll, hängt von sehr vielen unterschiedlichen Faktoren ab, sodass es hier kein absolutes Optimum gibt.

Zusammengefasst lässt sich sagen, dass der Big Data Hype für die Informatik viele Vorteile bringt. Die Datenspeicherung wird stetig effizienter, die Möglichkeiten der Analyse dieser Daten wachsen zusammen mit der Möglichkeit, sehr viele Daten zu speichern und neue Datenbanksysteme bringen stets neue Konzepte mit sich.

Jedoch bringen diese neuen Technologien viele neue Fragen, wie beispielsweise die Frage nach der Privatsphäre der Benutzer, mit sich, die noch beantwortet werden müssen. Diese Fragen reichen teilweise in den ethischen Bereich hinein, eine klare Antwort darauf gibt es hier noch nicht.

Literaturverzeichnis

YOUTUBE. (kein Datum). *Statistics*. Von YouTube Press: https://www.youtube.com/yt/press/statistics.html abgerufen

ANCHESTRY. (07. 1890). Abgerufen am 05. 02. 2013 von 1890 United States Federal Census: http://c.ancestry.com/pdf/trees/charts/1890UsCensus.pdf

AMAZON. (kein Datum). *Amazon EC2 Pricing*. Abgerufen am 08. 02. 2014 von Amazon Web Services: http://aws.amazon.com/ec2/pricing/

AMERICAN MEMORY HOME. (19. 04. 1895). Abgerufen am 05. 02. 2014 von Hollerith's electronic tabulating machine (from Railroad Gazette): http://memory.loc.gov/mss/mcc/023/0001.jpg

APACHE HADOOP. (04. 03. 2014). *Welcome to Apache Hadoop!* Abgerufen am 06. 03. 2014 von Apache Hadoop: http://hadoop.apache.org/

BLAKE, K. (1996). *National Archives*. Abgerufen am 30. 01. 2014 von "First in the Path of the Firemen" - The Fate of the 1890 Population Census, Part 1: http://www.archives.gov/publications/prologue/1996/spring/1890-census-1.html

BRÜCHER, C. (2013). *Rethink Big Data.* mitp.

BRUNO, L. C. (kein Datum). *Plate, punch card, and instructions for Herman Hollerith's Electric Sorting and Tabulating Machine, ca. 1895. (Herman Hollerith Papers)*. Abgerufen am 30. 01. 2014 von Words and Deeds in American History: Selected Documents Celebrating the Manuscript Division's First 100 Years: http://memory.loc.gov/cgi-bin/query/r?ammem/mcc:@field(DOCID+@lit(mcc/023))

BRANCKAUTE, F. (15. 02. 2012). *Facebook 2012 [Infographic]*. Abgerufen am 24. 01. 2014 von The Blog Herald:

http://www.blogherald.com/2012/02/15/facebook-2012-infographic/

CHANEY, P. (1. 08. 2012). *Understanding the 6 Facebook Post Types*. Abgerufen am 25. 01 2014 von Practical Ecommerce: http://www.practicalecommerce.com/articles/3680-Understanding-the-6-Facebook-Post-Types

CHANG, F., DEAN, J., GHEMAWAT, S., & HSIEH, W. C. (11. 2006). *Google Research Publications*. Abgerufen am 05. 02 2013 von Bigtable: A Distributed Storage System for Structured Data: http://research.google.com/archive/bigtable.html

ECMA INTERNATIONAL. (01. 10. 2013). *The JSON Data Interchange Format.* Abgerufen am 22. 02 2014 von Ecma International: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

ERNST, N., GREIS, F., & THOMA, J. (25. 07 2013). *Glossar zur NSA-Affäre*. Abgerufen am 05. 02 2014 von Golem: http://www.golem.de/news/glossar-zur-nsa-affaere-marina-prism-noforn-scissors-pinwale-sigad-us-984xn-1307-100428.html#Prism

DUMBILL, E. (31. 12. 2013). *Big Data Variety Means That Metadata Matters*. Abgerufen am 25. 01. 2014 von Forbes: http://www.forbes.com/sites/edddumbill/2013/12/31/big-data-variety-means-that-metadata-matters/

FACEBOOK, INC. (01. 01. 2014). *Key Facts*. Von Facebook Newsroom: http://newsroom.fb.com/Key-Facts

GOOGLE, INC. (25. 07. 2008). *We knew the web was big...* Abgerufen am 05. 02. 2014 von Google Official Blog: http://googleblog.blogspot.de/2008/07/we-knew-web-was-big.html

GOOGLE, INC. (kein Datum). *Google Unternehmen*. Abgerufen am 05. 02. 2014 von Unternehmensgeschichte im Detail: http://www.google.com/about/company/history/

GOOGLE, INC. (kein Datum). *Googlebot*. Abgerufen am 05. 02. 2014 von Google Webmaster-Tools-Hilfe: https://support.google.com/webmasters/answer/182072

HYPERTABLE, INC. (kein Datum). *Architecture | Hypertable - Big Data. Big Performance*. Abgerufen am 18. 02. 2015 von Hypertable - Big Data. Big Performance: http://hypertable.com/documentation/architecture/

HILLYER, M. (04. 2012). *Managing hierarchical data in MySQL.* Abgerufen am 22. 02. 2014 von Mike Hillyer's Personal Webspace: http://mikehillyer.com/articles/managing-hierarchical-data-in-mysql/

HORVATH, S. (2013). *Aktueller Begriff - Big Data.* Abgerufen am 21. 01. 2014 von Deutscher Bundestag: http://www.bundestag.de/dokumente/analysen/2013/Big_Data.pdf

KERSKEN, S. (06. 2007). *Praktischer Einstieg in MySQL mit PHP.* Abgerufen am 06. 03 2014 von O'Reilly: http://examples.oreilly.de/openbooks/pdf_einmysql2ger.pdf

LEMKE, J. (14. 08. 2013). *Google wegen Gmail und Privatsphäre in der Kritik*. Abgerufen am 05. 02 2014 von WinFuture: http://winfuture.de/news,77441.html

MAD SKILLS GMBH. (kein Datum). *Google-Account: Der Schlüssel zu den Google-Diensten*. Abgerufen am 05. 02. 2014 von Androidnext: http://www.androidnext.de/google-account/

MONGODB, INC. (2013). *Data Modeling Introduction*. Abgerufen am 22. 02. 2014 von MongoDB: http://docs.mongodb.org/manual/core/data-modeling-introduction/

MONGODB, INC. (2013). *Database References*. Abgerufen am 22. 02. 2014 von MongoDB: http://docs.mongodb.org/manual/reference/database-references/

MONGODB, INC. (2013). *Glossary*. Abgerufen am 22. 02. 2014 von MongoDB: http://docs.mongodb.org/manual/reference/glossary/#term-collection

MONGODB, INC. (2013). *Single Indexes*. Abgerufen am 22. 02. 2014 von MongoDB: http://docs.mongodb.org/manual/core/index-single/

MONGODB, INC. (kein Datum). *Specification - BSON.* Abgerufen am 22. 02. 2014 von BSON - Binary JSON: http://bsonspec.org/#/specification

MONGODB, INC. (2013). *Replication - Integration*. Abgerufen am 22. 02. 2014 von MongoDB: http://docs.mongodb.org/manual/core/replication-introduction/

ORACLE. (kein Datum). *MySQL Cluster CGE*. Abgerufen am 22. 02 2014 von MySQL: http://www.mysql.com/products/cluster/

PAKALSKI, I. (24. 08. 2013). *Prism Skandal: NSA zahlte Facebook, Google und Microsoft Millionenbeträge*. Abgerufen am 05. 02. 2014 von Golem: http://www.golem.de/news/prism-skandal-nsa-zahlte-facebook-google-und-microsoft-millionenbetraege-1308-101177.html

SAAKE, G., SATTLER, K.-U., & HEUER, A. (2013). *Datenbanken - Konzepte und Sprachen* (5. Auflage Ausg.). mitp.

SPICHALE, K., & WOLFF, E. (01. 2013). Big Data und zeitgemäße Datenverarbeitung. *iX Developer* , S. 67-98.

STATISTA. (2013). *Marktanteile der Suchmaschinen weltweit im September 2013*. Abgerufen am 05. 02. 2014 von Statista: http://de.statista.com/statistik/daten/studie/222849/umfrage/marktanteile-der-suchmaschinen-weltweit/

RODDEN, T. (11. 02 2014). Big Data and T's & C's. *Computerphile*. (S. Riley, Interviewer)

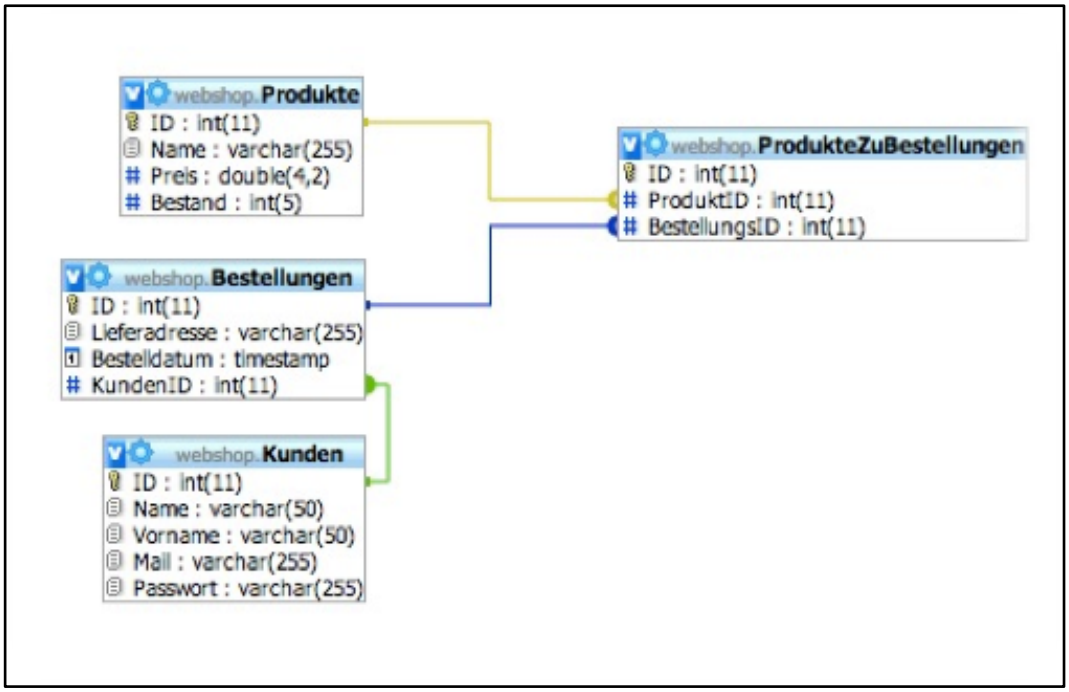
Anhang

1. und 2. Siehe USB-Stick

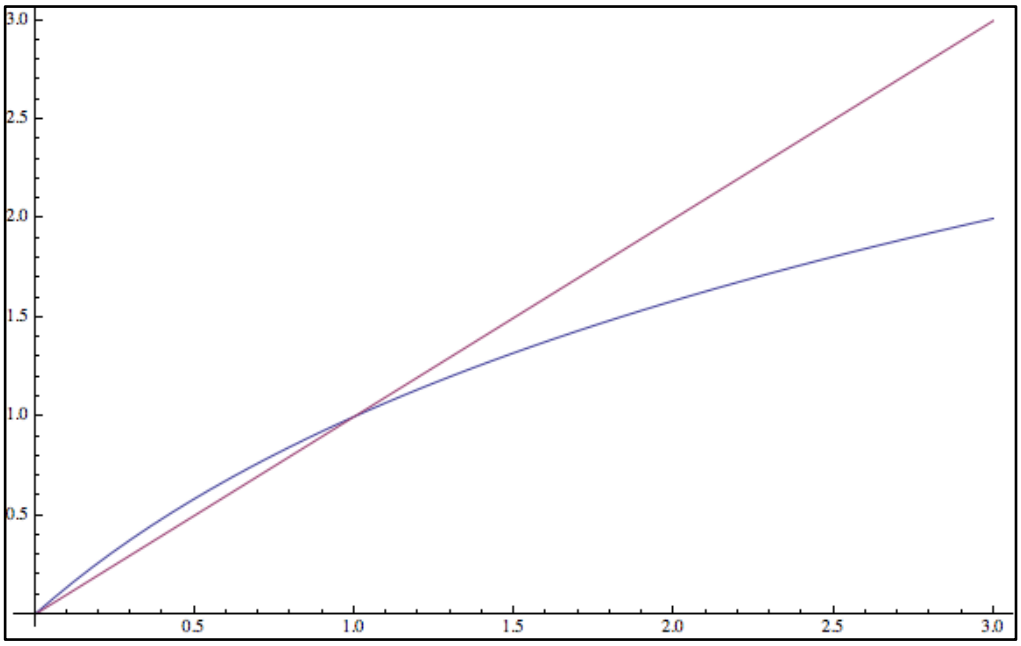
3.1. Mögliches Schema für eine Kundentabelle

ID	Name	Vorname	Mail	Passwort
1	Bernard	Leo	leo@mail.de	Test123
2	Rosescu	Paul	paul@mail.net	Passwort234
3	Gunz	Moritz	moritz@mail.com	Sicherheit456

3.2. Mögliches Schema der Verbindungen in einer Datenbank



3.3. Vergleich der Ordnungszahlen bei Abfragen mit und ohne Index



3.4. Mögliches Dokument in MongoDB

```
{
    "_id": ObjectId("..."),
    "name": "Bernard",
    "vornamen": [Leo", Phillip"],
    "mails": [
        {
            "mail": "mail1@provider.de",
            "newsletter": true,
            "priority": 10
        },
        {
            "mail": "mail2@provider.de",
            "newsletter": false,
            "priority": 5
        }
    ]
}
```