Assignment zum Modul DBA20

Thema: NoSQL-Datenbanken - Aufbau, Möglichkeiten und Grenzen

[redacted]

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

1.1 Problemstellung

Das World Wide Web ist ein Bereich in dem Entwicklungen schnell überholt sind. Das Internet ist einer der rasantesten Bereiche was die Veränderung betrifft. Tagtäglich entstehen neue Probleme für die schnellstmöglich eine Lösung gefunden werden muss, um den Wachstum und die Weiterentwicklung nicht zu bremsen.

Ein besonders wichtiges Thema, welches auch zukünftig immer mehr in den Vordergrund rückt, ist das drastisch ansteigende Datenvolumen. Das Analyseunternehmen IDC prognostiziert, dass der voraussichtliche Umfang der weltweiten Datensphäre im Jahr 2025 auf kolossale 163 Zettabyte (ZB) anwachsen wird. Das bedeutet eine Verzehnfachung der Datenmenge gegenüber dem Jahr 2016. [1]

Ein grundlegendes Problem von relationalen Datenbanken, die große Datenmengen indizieren liegt darin, dass sie bei steigendem Mengenaufkommen zwangsläufig immer langsamer werden, was für Firmen ein größeres Problem darstellt, da sie eine schnelle Verfügbarkeit von Daten benötigen. Dadurch, dass Daten in unterschiedlichsten Formaten vorliegen, passen diese auf den ersten Blick nicht in das relationale Raster der SQL-Datenbanken. Eine Umstellung auf NoSQL-Datenbanken kann hier Abhilfe schaffen, da diese eine höhere Geschwindigkeit aufweisen und keinerlei Struktur unterliegen.

1.2 Ziel und Aufgabe der Arbeit

Die Datenbankwelt reorganisiert sich derzeit stark und dass trotz der damaligen Datenexplosion durch das WEB2.0. In der Informatik ist NoSQL einer der Bereiche, der sich derzeit am rasantesten entwickelt. Dadurch ist es besonders schwer Bücher zu diesem Thema zu finden, da sich die API des Produktes höchstwahrscheinlich bis zum Abschluss eines Kapitels verändert hat. Aus diesem Grund wird in dieser Projektarbeit auch auf die allgemeinen Grundlagen Theorien und Techniken von NoSQL-Datenbanken eingegangen.

Ziel dieser Arbeit ist es, den praktischen Einsatz von NoSQL-Systemen zu erläutern, deren grundlegenden Aufbau darzustellen und die Möglichkeiten sowie Grenzen anhand eines jeweiligen Konzeptes aufzuzeigen.

[1] http://www.seagate.com/de/de/our-story/data-age-2025/

Zu Beginn der Projektarbeit soll eine Einführung in die Grundlagen von NoSQL Datenbanken gegeben werden, um die für die anschließende Ausarbeitung notwendigen Begriffe verständlich zu machen. Nachdem der Begriff NoSQL genauer erklärt wurde, werden grundlegende Konzepte vorgestellt, welche im NoSQL-Umfeld eingesetzt werden. Im weiteren Verlauf werden die vier verbreitetsten Datenbanktypen erläutert und jeweils ein Vertreter erörtert.[2]

2 Grundlagen Techniken und Theorien

2.1 Was versteht man unter NoSQL-Datenbanken

Unter dem Begriff NoSQL werden Datenverwaltungssysteme mit unterschiedlichen Funktionen zusammengefasst. Dort wo SQL-Datenbanken an ihre Grenzen stoßen oder nur mit großem Aufwand betrieben werden, können SQL- und NoSQL-Datenbanken gemeinsam zum Einsatz kommen. Wenn von NoSQL die Rede ist bedeutet das nicht, dass keine SQL-Datenbanken mehr eingesetzt werden dürfen, sondern sich nicht ausschließlich auf diese zu beschränken.

2.2 Begriffsdefinition / Eigenschaften

NoSQL-Datenbanken fehlt es jedoch nicht nur an einer klaren Bezeichnung, sondern auch an einer eindeutigen Definition. „Die Definition aus dem NoSQL-Archiv lautet in deutscher Übersetzung: Unter NoSQL wird eine neue Generation von Datenbanksystemen verstanden, die meistens einige der nachfolgenden Punkte berücksichtigen:

1. Das zugrundeliegende Datenmodell ist nicht relational.
2. Die Systeme sind von Anbeginn an auf verteilte und horizontale Skalierbarkeit ausgerichtet.
3. Das NoSQL-System ist Open Source.
4. Das System ist schemafrei oder hat nur schwächere Schemarestriktionen.
5. Aufgrund der verteilten Architektur unterstützt das System eine einfache Datenreplikation.
6. Das System bietet eine einfache API.
7. Dem System liegt meistens auch ein anderes Konsistenzmodell zugrunde: Eventually Consistent und BASE, aber nicht ACID.“ [3]

[2] Vgl. Edlich/Friedland/Hampe/Brauer/Brückner (2011) S.XV
[3] Vgl. Edlich/Friedland/Hampe/Brauer/Brückner (2011) S.2

2.3 Skalierung

Skalierbarkeit meint die Eigenschaft eines Software- oder Datenbanksystems mit den steigenden Anforderungen linear zu wachsen.[4]

2.3.1 Vertikale Skalierung

Das Aufrüsten vorhandener Server durch ersetzen veralteter Hardwarekomponenten wird als vertikale Skalierung ("Scale-up") verstanden. Durch das Aufrüsten von neuen Komponenten wird nicht nur die Leistung verbessert, sondern auch die Qualität und somit die Ausfallsicherheit gewährleistet. Eine wichtige Rolle spielen die Kosten der neuen Komponenten, da diese linear verlaufen und im Hochpreissegment exponentiell steigen können. Beim Ausfall von Komponenten oder dem Umrüsten des Servers entstehen Ausfallzeiten, was bei einer geringen Anzahl an Servern zu Problemen führen kann.[5]

2.3.2 Horizontale Skalierung

Durch die Erhöhung der verfügbaren Server und einer Vernetzung der Datenbanken wird eine horizontale Skalierung ("Scale-out") verstanden. Die Anzahl der Server kann beliebig hoch sein und muss auch keine Ausfallsicherheit aufweisen. Im Gegensatz zu den vorhandenen Knotenpunkten müssen die zur Erweiterung dienenden Server nicht leistungsfähig sein. Der Vorteil von horizontal skalierten Systemen liegt in der Verfügbarkeit und Ausfalltoleranz. Der Ausfall eines einzelnen Serverknotens kann durch die Redundanz des gesamten Systems kompensiert werden.[6]

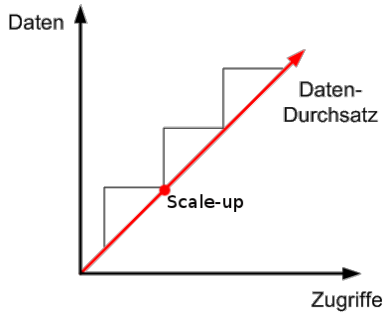

Abb. 1: Vertikale Skalierung

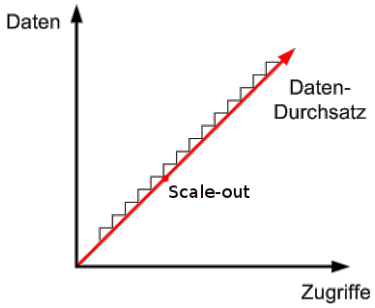

Abb. 2: Horizontale Skalierung

[4] Vgl. Kurowski (2012) S.20
[5] http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/Skalierbarkeit
[6] http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/Skalierbarkeit

2.4 Wichtige Konzepte

Zu den Grundlagen, die zur Entwicklung von NoSQL-Systemen beigetragen haben, gehören einige wichtige Konzepte wie Cap Theorem, Base oder MVCC.

2.4.1 Cap Theorem

Das CAP Theorem beschreibt eine Eigenheit von verteilten Datenbanksystemen, in dem maximal zwei von drei Anforderungen erreichbar sind. Dabei muss individuell entschieden werden, ob die Datenbank als eine **CA**-, **CP**-, **AP**-Applikation zu realisieren ist. Mittlerweile hat sich die Erkenntnis durchgesetzt, dass die im WEB2.0 Zeitalter zu verarbeitenden Datenmengen sich nur noch auf der Basis dynamisch skalierbarer Servernetze umsetzen lassen.[7]

C: Consistency (Konsistenz):
Die Konsistenz des Systems ist im Sinne des CAP Theorem gegeben, sobald alle Knoten des verteilten Systems zu einem beliebigen Zeitpunkt die gleichen Daten sehen.
A: Availability (Verfügbarkeit):
Wenn das System auf alle Anfragen antwortet ist es verfügbar.
P: Partition tolerance (Partitionstoleranz):
Wenn bei einer Unterbrechung der Kommunikation mit einem Teil der Knoten das System noch funktionstüchtig bleibt, spricht man von Partitionstoleranz.

2.4.2 BASE

BASE steht als Kurzwort für Basically Available, Soft State, Eventually Consistent und ist weit verbreitet in NoSQL-Systemen. Beim Designprinzip von BASE wird die absolute Konsistenz aufgegeben und dafür die Verfügbarkeit des Systems erhöht. Eine solche Begegnung zwischen Konsistenz und Verfügbarkeit spiegelt sich auch im CAP Theorem wieder.[8]

2.4.3 MVCC

Multiversion concurrency control (MVCC) ist eine Technik zur parallelen Verarbeitung von Datenzugriffen bei Datenbankmanagementsystemen.

[7] http://lwibs01.gm.fh-koeln.de/wikis/wiki_db/index.php?n=Datenbanken.CAP
[8] https://db-engines.com/de/article/BASE

Bei MVCC wird im Falle einer Datenänderung der frühere Datenstand aufgehoben und mit einem Timestamp oder einer Transaktions-ID versehen. Aufgrund dessen ist es möglich gültige und konsistente Daten zu liefern ohne Datenänderungen dabei zu blockieren. Durch die notwendige Verwaltung von mehreren Versionen wird ein größeres Datenvolumen erzeugt und das Löschen alter Versionen erfordert einen Aufwand. Bei vielen DBMS Systemen wird als Standardmethode MVCC verwendet oder als Option angeboten.[9]

3 Die wichtigsten NoSQL-Datenbanktypen

Vier der wichtigsten Datenbanktypen, darunter Key-Value-Datenbanken, dokumentorientierte Datenbanken, graphenorientierte Datenbanken und spaltenorientierte Datenbanken, werden im Anschluss erläutert sowie Vor- und Nachteile aufgezeigt. Anhand eines Beispiels soll das Grundprinzip des Datenbanktyps vereinfacht dargestellt werden. Anhand der Grafik erkennt man schnell die „Core NoSQL" Datenbanktypen, welche sich von den relationalen SQL Datenbanken deutlich unterscheiden.

3.1 Key-Value-Systeme

3.1.1 Allgemein

Die seit den 70er Jahren zum Einsatz kommenden KeyValue-Datenbanksysteme zählen zu den ältesten NoSQL-Datenbanksystemen. Weitaus erstaunlicher ist daher, dass sie erst im Web2.0 und Cloud-Zeitalter ihren richtigen Aufschwung erleben. Das liegt mitunter daran, dass in den 90er Jahren fast alle Daten in relationalen Systemen abgelegt wurden. Mit Firmen wie Amazon, Facebook oder Twitter ist das Datenvolumen angestiegen und wurde somit auf viele unterschiedliche Rechnerinstanzen verteilt, wofür sich KeyValue-Systeme besonders eignen.
An relationale Datenbanken ist bei einem solchen Datenvolumen nicht mehr zu denken.[10]
Key-Value-Systeme sind nach einem einfachen Schlüssel- und Wertschema aufgebaut. Dabei ist einem Wert ein bestimmter Schlüssel zugeordnet, der aus geordneten oder willkürlichen Zeichenabfolge bestehen kann. Eine Aufteilung der Schlüssel in Namensräume oder Datenbanken ist möglich. Das System kann Werte (Values) wie Strings, Listen, Sets oder Hashes beinhalten.[11]

[9] https://db-engines.com/de/article/MVCC
[10] Vgl. Edlich/Friedland/Hampe/Brauer/Brückner (2011) S.151
[11] http://lwibs01.gm.fh-koeln.de/wikis/wiki_db/index.php?n=Datenbanken.KeyValueSysteme

3.1.2 Vor und Nachteile

+ Ein Vorteil des Systems besteht in der Skalierbarkeit, sowie der schnellen und effizienten Datenverwaltung, aufgrund des einfachen Schemas.
+ Datenbanken müssen bei Änderungen nicht für eine längere Zeit gesperrt werden, stattdessen wird der Ansatz der Datenversionierung verfolgt.
+ Durch verzichten auf komplexe Suchalgorithmen und Indizes, können so bessere Laufzeit für Lese- und Schreiboperationen ermöglicht werden.

- Je nach Projekt ist es umständlich bestimmte Werte zu suchen.
- Benutzer kann darüber bestimmen wie die Daten abgespeichert werden. Dadurch kann es zu einer Datenüberschreibung und somit zum Datenverlust kommen.
- Aufgrund der einfachen Handhabung des Schemas, bieten die Systeme oft nur eingeschränkte Abfragemöglichkeiten. [12]

3.1.3 Beispiel anhand der Datenbank „Redis“

Redis speichert alle Daten standardmäßig im RAM, dadurch zählt sie zu den sogenannten Hauptspeicherdatenbanken. Nach erfüllen bestimmter Kriterien können die Daten auf die Festplatte verschoben und persistent geschrieben werden. Dadurch zählt Redis zu einer der schnellsten Key-Value-Datenbanken.

Folgende Typen werden als Values unterstützt:

- Strings
- Hashes
- Listen mit möglichen Duplikaten
- Sets ohne Duplikate
- Sortierte Sets

3.1.3.1 Installation

Auf allen POSIX-kompatiblen Unix-Systemen gestaltet sich die Installation von Redis schnell und einfach. Folgende Befehle müssen zur Installation unter Ubuntu ausgeführt werden:

[12] http://wikis.gm.fh-koeln.de/wiki_db/index.php?n=Datenbanken.KeyValueSysteme

`$ wget http://redis.googlecode.com/files/redis-1.2.2.tar.gz` // Download Software

`$ tar xvzf redis-1.2.2.tar.gz` // Entpacken des Archivs in das Verzeichnis redis-1.2.2

`$ cd redis-1.2.2` // Selektieren des Verzeichnises redis-1.2.2

`$ make` // Erzeugen der Executable durch kompilieren

`$ ./redis-server` // Server starten

`$ sudo cp redis-server redis-cli /usr/local/bin` // Kommandos global machen[13]

3.1.3.2 Arbeiten mit Redis

Nach der Installation von Redis werden beispielhaft einige einfache Befehle zum Arbeiten beschrieben. Zu beachten ist, dass die Keys von Redis case-sensitive sind und somit „value“ ein anderer Schlüssel als „Value“ ist.

`INFO` // Ausgabe von Informationen zur Datenbank

`SET Value1 10001` // Speicher den Value '10001' mit dem Key 'Value1'

`SET Value2 10002` // Speicher den Value '10002' mit dem Key 'Value2'

`SET Vakue3 10003` // Speicher den Value '10003' mit dem Key 'Value3'

`KEYS '*'` // Ausgabe aller Keys

`EXIST Value1` // Prüft ob der Key ' Value1' vorhanden ist

`DBSIZE` // Gibt die Größe der Datenbank zurück

`MOVE Value3 2` // Verschiebt den Key in die Datenbank Nummer 2

`DEL Value3` // Löscht den Key 'Value3'

`FLUSHALL` // Löscht alle Keys in allen Datenbanken

`SAVE` // Speichert die im RAM befindlichen Daten auf der Festplatte

`LASTSAVE` // Gibt den integer Zeitwert des letzten Speicherns zurück

`MSET Value3 10003 Value2 10001 Value2 10002` // Multi-Set

`MGET Value3 Value1 Value2` // Multi-Get

`SHUTDOWN` // Fährt den Server herunter

[13] http://wikis.gm.fh-koeln.de/wiki_db/index.php?n=Datenbanken.Redis

3.2 Spaltenorientierte Datenbanken

3.2.1 Allgemein

Das Speicherprinzip bei spaltenorientierten Datenbanken ist vom Grundprinzip anders als bei relationalen Datenbanken. Dabei werden die Werte nicht in Zeilen, sondern in Spalten gespeichert. Anstelle von Tabellen werden sogenannte „Column Families“ erstellt, welche an keine Struktur gebunden sind und beliebig viele Spalten enthalten können (Abb.3).

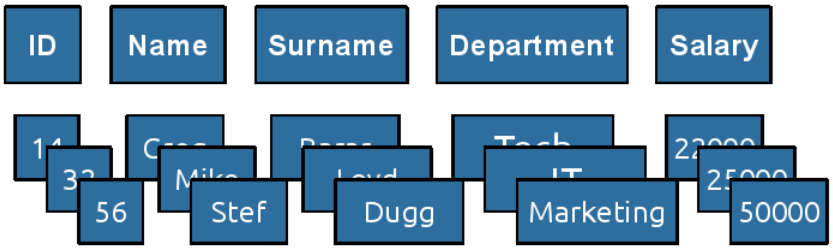

Abb. 3: Column Familie

3.2.2 Vor und Nachteile

+ Keine unnötigen Daten werden während des Leseprozesses gelesen.
+ Nur ausgewählte Daten werden gelesen
+ Schneller Schreibeprozess bei einzelnen Spalten (in der Regel disk-seek)

- Schreiben einer Vielzahl von Daten aus mehreren Spalten
- Bei Zugriff auf mehrere Spalten wird der Schreibvorgang verlangsamt[14]

3.2.3 Beispiel anhand der Datenbank „Cassandra“

Cassandra zählt zu den spaltenorientierten Datenbanken, welche zur Optimierung der Suche im Posteingang von den Software-Ingenieuren Facebook’s entwickelt wurde. 2008 wurde Cassandra als Open Source freigegeben und 2010 von der Apache Software foundation als Top-Level-Projekt aufgenommen.

Cassandra unterstützt die Erstellung und Entfernung von Spalten, ohne die Struktur einer Column Family zu verändern. Das Datenmodell hat kein Schema oder logische Struktur.

[14] http://wikis.gm.fh-koeln.de/wiki_db/index.php?n=Datenbanken.SpaltenorientierteDatenbank

3.2.3.1 Installation

Dadurch, dass Cassandra in Java implementiert ist wird eine Java Virtual Machine vorausgesetzt. Sobald die Umgebungsvariable JAVA_HOME gesetzt und die JVM unter Path eingetragen ist, beschränkt sich die Installation von Cassandra auf das Herunterladen des zuletzt veröffentlichten Releases:

aktuell 3.11, http://cassandra.apache.org/download/ // Downloaden des Archivs

$ tar -xzf apache-cassandra-0.7.5-bin.tar.gz // Entpacken des Archivs

3.2.3.2 Arbeiten mit Cassandra

Seit Juni 2011 ist die Sprache CQL verfügbar (Cassandra Query Language), die den Befehlen von SQL ähnelt. CQL unterstützt alle grundlegenden Befehle zur Verwaltung, Abfragung und Definierung von Daten. Die wichtigsten Befehle lauten:

```
cqlsh> help

Documented shell commands:
===========================
CAPTURE COPY DESCRIBE EXPAND PAGING SOURCE
CONSISTENCY DESC EXIT HELP SHOW TRACING.

CQL help topics:
================
ALTER           CREATE_TABLE_OPTIONS       SELECT
ALTER_ADD       CREATE_TABLE_TYPES         SELECT_COLUMNFAMILY
ALTER_ALTER     CREATE_USER                SELECT_EXPR
ALTER_DROP      DELETE                     SELECT_LIMIT
ALTER_RENAME    DELETE_COLUMNS             SELECT_TABLE
```

Abb. 4: Übersicht und Beschreibung der cqlsh Befehle[15]

3.3 Dokumentenbasierte Datenbanken

3.3.1 Allgemein

Im Gegensatz zu relationalen Datenbanken werden die Daten nicht in Spalten und Tabellen gespeichert, sondern in sogenannte Dokumente. In diesem Zusammenhang versteht man unter einem Dokument eine strukturierte Zusammenstellung bestimmter Daten. Nicht zu verwechseln mit textverarbeitungsbasierten erstellten Dateien, die ebenfalls als Dokumente bezeichnet werden. Eine Definition dieser Dokumente in der Datenbank gibt es nicht. XML oder JSon sind die

[15] http://www.w3ii.com/de/cassandra/cassandra_shell_commands.html

gängiger Dokumentenstrukturen. Die bestehenden Beziehungen werden in der Softwar definiert, da es keine Verbindung der Daten untereinander gibt. Die verbreitetsten Implementierungen sind „CouchDB“ und „MongoDB“. [16]

3.3.2 Vor und Nachteile

+ Datenbanktyp ist schemafrei
+ Große Freiheit für den Anwender
+ JSON-Format erleichtert die Nutzung
+ Möglichkeit zusammenhängende und große Daten geschlossen abzuspeichern
+ Abrufen zusammenhängender Daten

- Anwender muss Struktur der Dokumente selbst festlegen und kontrollieren
- Keine Überprüfung der eingegebenen Werte[17]

3.3.3 Beispiel anhand der Datenbank „MongoDB“

MongoDB zählt zu den beliebtesten Datenbanken der NoSQL-Vertreter. Dies liegt ziemlich wahrscheinlich daran, dass MongoDB mit Features überzeugen kann, die bei anderen Vertretern vermisst werden. Das Ziel von MongoDB besteht darin die Lücke zwischen klassischen relationalen Datenbanken und den Key-Value-Stores zu schließen. MongoDB ist in C++ programmiert und kann an viele bekannte Programmiersprachen angebunden werden. [18]

3.3.3.1 Installation

In meinem Beispiel wird die Installation unter Windows erklärt. Da MongoDB keine Installations- oder Setuproutine hat, müssen alle Schritte per Hand durchgeführt werden.

`https://www.mongodb.com/download-center#community` // Downloaden der Software

`C:\MongoDB\TestDB` // Erzeugen eines Verzeichnisses

`C:\data\db` // Zweites Verzeichnis zur Speicherung

[16] Vgl. Edlich/Friedland/Hampe/Brauer/Brückner (2011) S.117
[17] Vgl. Kurowski (2012) S.70
[18] Vgl. Edlich/Friedland/Hampe/Brauer/Brückner (2011) S.131

3.3.3.2 Arbeiten mit MongoDB

Bei MongoDB werden die Dokumente intern gespeichert, im Speicher gehalten und mit dem Client Driver in BSON-Format getauscht. Voraussetzung dafür ist, dass die Client Driver die BSON-fähigkeit besitzen. Auf der Grafik ist zu erkennen, dass die Datenbank Struktur in MongoDB eine andere ist als in relationalen Datenbanken. Die Datenbank enthält beliebig viele Collections, in denen sich wiederum beliebig viele Dokumente befinden.

<table>
<tr><th colspan="3">Datenbank Name: I-net</th><th>Datenbank Name: I-net</th></tr>
<tr><td colspan="3">Tabelle „users“</td><td>Collection „users“</td></tr>
<tr><td colspan="2">id</td><td>name</td><td rowspan="3">{_id:1, name: „Bea“}

{_id:2, name: „Tanja“}</td></tr>
<tr><td colspan="2">1</td><td>Bea</td></tr>
<tr><td colspan="2">2</td><td>Tanja</td></tr>
<tr><td colspan="3">Tabelle „posts“</td><td>Collection“posts“</td></tr>
<tr><td>Id</td><td>User_ID</td><td>comment</td><td rowspan="4">{ _id:1;
User_id:1,
Comment: „Heute ist Vortrag“}
{ _id:1;
User_id:1,
Comment: „Kann es losgehen?“}
{ _id:1;
User_id:1,
Comment: „immer doch“}</td></tr>
<tr><td>1</td><td>1</td><td>Heute ist Vortrag</td></tr>
<tr><td>2</td><td>1</td><td>Kann es losgehen?</td></tr>
<tr><td>3</td><td>2</td><td>Immer doch</td></tr>
</table>

Abb. 5: Datenbankvergleich: relationale vs. dokumentenorientiert [19]

3.4 Graphendatenbanken

3.4.1 Allgemein

Bereits in den 60er Jahren entstanden die ersten graphenorientierte Datenbanken, bevor die ersten relationalen Datenbanken entwickelt wurden. In den frühen 90er Jahren wurden sie fast vollständig durch spezialisierte Ansätze verdrängt. In der heutigen Zeit steigt jedoch das Interesse für graphenorientierte Datenbanken an, aufgrund des Wachstum der Graphenstrukturen im Internet und in Anwendungen. In sozialen Netzwerken wie Twitter können durch diese Datenbanken komplexe Beziehungen abgespeichert und performant wiedergegeben werden.

Die Graphendatenbank besteht aus Knoten und Kanten, dabei stellen die Knoten immer eine Entität dar und die Kanten die Beziehung zwischen den Entitäten. Für eine detaillierte Beschreibung können den Knoten und Kanten Attribute hinzugefügt werden.

[19] http://wikis.gm.fh-koeln.de/wiki_db/index.php?n=Datenbanken.MongoDB

3.4.2 Vor und Nachteile

+ Vereinfachung des Verständnisses durch Abbildung realer Netzwerkstrukturen
+ Hohe Performance aufgrund des Verzichts von JOIN-Operationen
+ Graphendatenbanken weisen eine übersichtliche Struktur auf
+ Optimierung der Datenbank durch bereits entworfene Algorithmen

- Komplizierte Nutzbarkeit gegenüber relationaler Datenbanken
- Keine verständliche Abfragesprache im Vergleich zur SQL Sprache[20]

3.4.3 Beispiel anhand der Datenbank „Neo4j“

Unter Neo4j versteht man eine voll ACID-transaktionale Graphendatenbank, welche die gesamten Daten auf Datenträger in einem selbst entwickelten Format abspeichert. Die Datenbank kann sowohl in eine Java-Anwendung eingebettet werden oder als eigenständiger Datenbankserver laufen, wodurch eine sehr hohe Verarbeitungsgeschwindigkeit entsteht. Neo4j ist mit ihrer Open Source Community einer der bekanntesten und beliebtesten Graphendatenbanken, aufgrund der hohen Performance und der vielen Schnittstellen für die unterschiedlichsten Programmiersprachen. [21]

3.4.3.1 Installation

Auf GitHub können drei verschiedenen Varianten von Neo4j heruntergeladen werden (Community Edition, Advanced Edition, Enterprise Edition). In meinem Beispiel wird die Installation unter Windows erklärt.[22]

`http://neo4j.com/download/` // Downloaden der Software

`C:\bin\neo4j` // Entpacken in gewünschtes Verzeichniss

`\bin\neo4j console` // Ausführen des Programms in der Konsole

`CTRL-C` // Stoppen des Programms in der Konsole

[20] Vgl. Edlich/Friedland/Hampe/Brauer/Brückner (2011) S.207 ff
[21] Vgl. Edlich/Friedland/Hampe/Brauer/Brückner (2011) S.290 ff
[22] https://neo4j.com/docs/operations-manual/current/installation/windows/

3.4.3.2 Arbeiten mit Neo4j

Anhand eines einfachen Abfrage-Beispiels kann man das Grundprinzip von Neo4j erklären. Dabei würde die Abfrage in Cypher wie folgt aussehen:

```
START film=node:node_auto_index(titel="Matrix
MATCH (schauspieler)-[rolle:SPIELT_IN]->(film
WHERE schauspieler.alter > 32
RETURN schauspieler.name, COLLECT(rolle.rolle) AS rollen
```

Zum Erzeugen von Knoten und Beziehungen wird bei Cypher die CREATE Operation verwendet. Anschließend der Film „Matrix“, die Rolle „Neo“ des Schauspielers „Kenau Reeves“:

```
CREATE film={titel:"Matrix"},
schauspieler={name:"Keanu Reeves", alter:48},
actor-[:SPIELT_IN {rolle:"Neo"}]->film;
```

Zum Hinzufügen von neuen Schauspielern ("Carrie-Anne Moss" als "Trinity") ermitteln wir den Knoten für den Film „Matrix“ aus dem Index, welcher wieder mit dem Befehl „CREATE“ verbunden wird.[23]

```
START film=node:node_auto_index(titel="Matrix")
CREATE schauspieler={name:"Carrie-Anne Moss", alter:45},
schauspieler-[:SPIELT_IN {rolle:"Trinity"}]->film
RETURN schauspieler.name;
```

4 Fazit und Ausblick

Die Entdeckung und der Grundgedanke von NoSQL-Datenbanken liegt bereits weit zurück. Da sie von Beginn an im Schatten von relationalen Datenbanken gestanden haben, hat sich bis zur Entstehung des WEB2.0 keine große Entwicklung von NoSQL-Datenbanken ergeben. Aufgrund des rasanten Anstiegs von Datenmengen die im Zeitalter des WEB2.0 angefallen sind, spielt die NoSQL-Datenbank eine wichtige Rolle, da relationale Datenbanken nicht mit solch großen Datenmengen umgehen können. Anhand der aufgeführten Vor- und Nachteile von NoSQL-

[23] http://www.chip.de/artikel/Graphdatenbanken-Mehr-Intelligenz-fuer-Big-Data-5_59448531.html

Datenbanken zeigt sich deutlich, dass die reine NoSQL-Datenbank keine Allzwecklösung ist. In Kombination mit anderen Datenbanktypen können sie jedoch eine gute Verfügbarkeit, Ausfalltoleranz und horizontale Skalierung zur Verfügung stellen. Aufgrund dessen wird in der Fachliteratur auch von „Not only SQL“ gesprochen.
Das CAP-Theorem besagt jedoch, dass Ausfalltoleranz, Verfügbarkeit und Konsistenz nicht zur gleichen Zeit erreicht werden können. Aus diesem Grund muss bei der Implementierung von NoSQL-Datenbanken auf die permanente Konsistenz verzichtet werden. Dadurch wurde ein neues Konsistenzmodell (BASE) eingeführt, wodurch die Daten zwar nicht permanent, jedoch zeitweise Konsistent sind. Das Verwenden des MVCC-Verfahren stellt eine Versionierung der Datensätze bereit, zum Erreichen dieser zeitweisen Konsistenz.

Was viele Firmen von der Verwendung von NoSQL-Datenbanken abhält ist das nicht standardisierte Verfahren, was relationale Datenbanken mitbringen. Jedoch haben sich die ersten namhaften Unternehmen der Internetbranche an NoSQL-Datenbanken herangetraut und anhand von Auswertungen gezeigt, dass sie eine höhere Performance aufweisen und weniger Datenmengen produzieren. Mit diesen Vorteilen gegenüber relationale Datenbanken gewinnen NoSQL-Datenbanken immer mehr an Bedeutung, was mit zunehmender Digitalisierung weiterhin der Fall sein wird. Ein plötzlicher Umstieg auf NoSQL-Datenbanken würde bei den meisten Unternehmen keinen Sinn machen, jedoch experimentieren einige Unternehmen an Kombilösungen zwischen relationalen und NoSQL-Datenbanken (Not only SQL) um das Optimum an Performance herauszuholen. Dadurch wird das Thema NoSQL-Datenbanken weiterhin viel Aufmerksamkeit auf sich ziehen.

Literaturverzeichnis

EDLICH, S; FRIEDLAND, A; HAMPE, J; BRAUER, B; BRÜCKNER, M.
NoSQL Einstieg in die Welt nichtrelationaler WEB2.0 Datenbanken, 2., aktualisierte und erweiterte Auflage 2011.

KUROWSKI, O.
NoSQL Einführung CouchDB, MongoDB und Redis, E-Book 2012.

DB-ENGINES.
Internet: https://db-engines.com/de/articles (letzter Zugriff am 22. August 2017).

FH KÖLN WIKI.
http://wikis.gm.fh-koeln.de/wiki_db/index.php?n=Category.NoSQL (letzter Zugriff am 22. August 2017).

CHIP.DE.
http://www.chip.de/artikel/Graphdatenbanken-Mehr-Intelligenz-fuer-Big-Data-5_59448531.html (letzter Zugriff am 18. August 2017).

DATENBANKEN VERSTEHEN.
http://www.datenbanken-verstehen.de/lexikon/nosql/ (letzter Zugriff am 01. August 2017).

BIGDATA-INSIDER.
http://www.bigdata-insider.de/was-ist-nosql-a-615718/ (letzter Zugriff am 01. August 2017).

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]