Migration zwischen relationalen und dokumentenorientierten Datenbanken

Laura Kuperjans

HAW Hamburg, Berliner Tor 7, 20099 Hamburg, Deutschland
laura.kuperjans@haw-hamburg.de

Zusammenfassung. Die Motivation hinter einer Datenbankmigration kann vielfältig sein. Mit der Etablierung der NoSQL Datenbanken entstand die Herausforderung zwischen verschiedenen Arten von Datenbanken zu migrieren, welche nicht mehr dem relationalen Ansatz entsprechen. Diese Arbeit befasst sich mit der Migration von einer relationalen zu einer dokumentenorientierten, sowie zwischen dokumentenorientierten Datenbanken. Des Weiteren wird das Grundkonzept der Masterarbeit zu dieser Thematik vorgestellt.

Schlüsselwörter: Datenbankmigration · NoSQL · relationale Datenbanken · dokumentenorientierte Datenbanken.

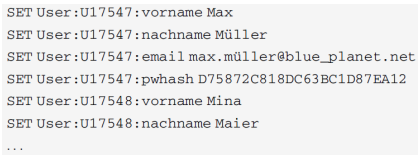
1 Einführung

Die Entscheidung für eine Datenbankmigration kann vielfältige Gründe haben, wie beispielsweise eine Kosteneinsparung durch den Wechsel von einer proprietären oder kommerziellen Datenbank zu einer Open Source. Ein anderer Grund kann sein, dass der zugrundeliegende Datensatz und die Anforderungen an diesen in einem anderen Datenbankmodell optimaler umgesetzt werden können.

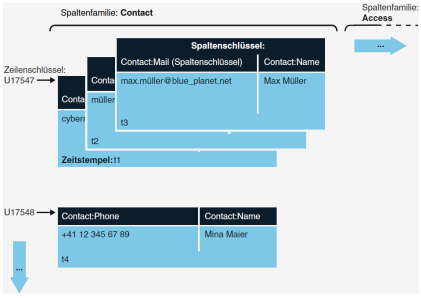
Bei einer Datenbankmigration werden der Datensatz, sowie die dazugehörigen Funktionalitäten von der Quell- in die Zieldatenbank transferiert. Unter Funktionalitäten werden in diesem Kontext sämtliche datenbankspezifische, unterstützte Funktionen verstanden, wie z.B. Views, Trigger, Stored Procedues, Abfragen etc. Erfolgt eine Migration zwischen verschiedenen Arten von Datenbanken mit unterschiedlichen Modellen, z.B. zwischen einer relationalen und einer NoSQL, muss zusätzlich noch eine Vorgehensweise bei der Umwandlung der Struktur der Daten erstellt werden. Für eine erfolgreiche Umsetzung des Migrationsprozesses sollten zuvor alle Aspekte der Quelldatenbank analysiert und die Machbarkeit für die Zieldatenbank berücksichtigt werden, sowie auf eine strukturierte und dokumentierte Arbeitsweise geachtet werden [24].

Unter NoSQL (Not Only SQL) Datenbanken versteht man jene, welche nicht dem relationalen Modell zuzuordnen sind. Diese werden typischerweise in vier Kategorien unterteilt, den Schlüssel-Wert-, Spaltenfamilien, dokumentenorientierten und Graphdatenbanken, welche für spezielle Fälle angepasste Lösungen

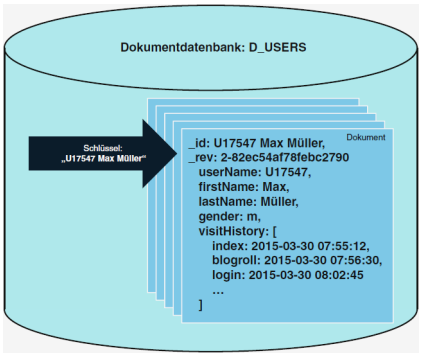
bieten. In Abbildung 1 wird die Speicherung der Benutzerdaten von Websites im Bezug auf die unterschiedliche Hinterlegung der Daten bei den vier Basiskategorien dargestellt. (vgl. [16])



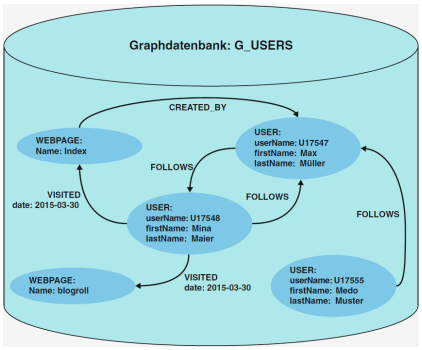
(a) Schlüssel-Wert-Datenbank



(b) Spaltenfamilien Datenbank



(c) Dokumentenorientierte Datenbank



(d) Graphdatenbank

Abb. 1: Vier Basiskategorien der NoSQL Datenbanken (Abbildungen entnommen aus: [16])

Grundsätzlich ergeben sich vier Möglichkeiten zwischen welchen Datenbanktypen migriert werden kann, der Migration zwischen zwei relationalen, relational und NoSQL, NoSQL der gleichen Art und zwischen verschiedenen Arten von NoSQL Datenbanken.

Die Migration bei relationalen Datenbanken ist bereits sehr gut erforscht, sodass eine Vielzahl an Tools für den Prozess zur Verfügung gestellt werden. Beispielsweise gibt es für PostgreSQL[1] ein inoffizielles Wiki [21], welches für verschiedenste Quelldatenbanken eine Liste an Tools zur Verfügung stellt. Darunter sind kommerzielle oder Open Source Tools, Skripte auf Github oder Tools zu Cloud Services wie AWS[2]. Die Migration zwischen verschiedenen NoSQL Datenbanken stellt einen komplexen Fall dar, da es sich um schemafreie Daten-

[1] https://www.postgresql.org/
[2] https://aws.amazon.com/de/

banken handelt und dies erschwert es eine allgemeingültige Vorgehensweise für die Migration zu definieren. In der Literatur wird dieser Fall z.B. von Wijaya et al. [24] oder Bansel et al. [1] erprobt, jedoch an simplen Datensätzen wie der von Twitter, welcher dies aufgrund der einfachen Struktur gut ermöglicht. Bei komplexeren Beispielen, welche nicht eine Struktur aufweisen, können zwar Lösungen für das Problem gefunden werden, dies setzt jedoch nicht voraus, dass es genau so auch bei einem anderen Beispielfall funktioniert. So wurden für diese Arbeit die beiden übrigen Migrationsmöglichkeiten ausgewählt, mit dem Schwerpunkt auf dokumentenorientierten Datenbanken. Grundsätzlich wird literarisch die Migration von allen vier Kategorien von NoSQL Datenbanken behandelt. Die dokumentenorientierten wurden deshalb ausgewählt, da es in der Literatur eine breite Menge an Ansätzen zu dem Thema gibt, jene Datenbanken zu den populärsten gehören und somit am häufigsten in der Verwendung sind. Dies kann anhand des Rankings von DB-Engines [2] oder unter Statista [23] nachvollzogen werden. Ebenso kann der Wechsel von einer relationalen zu einer dokumentenorientierten Datenbank eine Performanceverbesserung bewirken. Dazu wird die Zeit gemessen, die das jeweilige Datenbanksystem für die Ausführung der vier Basisoperationen (Insert, Update, Delete und Select) benötigt. Zumeist wird die MongoDB[3] als Vertreter für das dokumentenorientierte Modell ausgewählt und z.B. MySQL[4] [7], PostgreSQL [9] oder Microsoft SQL Server[5] [19] für das relationale. Bei der Messung der benötigten Zeit für die Basisoperationen kann die MongoDB besser abschneiden als der Vertreter der relationalen Datenbank. Diese Vergleiche beschränken sich nicht nur auf die MongoDB, sondern können auch mit verschiedenen NoSQL Datenbanken [13] durchgeführt werden.

Erfolgt eine Migration von einer relationalen zu einer dokumentenorientierten Datenbank, so ist die Tabellenstruktur mit den Beziehungen untereinander aufzulösen. Dargestellt werden kann diese Struktur mit dem ER-Modell (Entity-Relationship Modell). Nach Geisler [6] wird dies wie folgt definiert. Eine Entität ist in dem Modell eine Tabelle, welche verschiedene Beziehungen zu anderen besitzt. Man unterscheidet zwischen drei verschiedenen Beziehungen, der 1:1-, 1:n- und m:n-Beziehung zwischen zwei Tabellen. Bei der 1:1-Beziehung wird jedem Datensatz aus Tabelle_1 einer aus Tabelle_2 zugeordnet und umgekehrt. Bei der 1:n-Beziehung sind einem Datensatz aus Tabelle_1 beliebig viele aus Tabelle_2 zugeordnet, umgekehrt ist einem Datensatz aus Tabelle 2 ein Datensatz aus Tabelle 1 zugeordnet. Bei der n:m-Beziehung sind beliebig vielen Datensätzen aus Tabelle_1 beliebig viele aus Tabelle_2 zugeordnet. Dargestellt werden kann dieser Fall im relationalen Modell nur über eine Zwischentabelle, welche die Kombinationen von Primärschlüsseln aus den beiden Tabellen speichert. Somit ergeben sich zwei 1:n-Beziehungen zu der Zwischentabelle.

[3] https://www.mongodb.com/de
[4] https://www.mysql.com/de/
[5] https://www.microsoft.com/de-de/sql-server/sql-server-2019

2 Vergleichbare wissenschaftliche Arbeiten

Die Literatur befasst sich bei der Migration von einer relationalen zu einer NoSQL Datenbank mit allen vier Basiskategorien. Schlüssel-Wert-Datenbanken [5] stellen dabei den am wenigsten komplexen Fall dar, da hier die Daten in Schlüssel-Wert-Paaren hinterlegt werden und dementsprechend für die Datensätze eindeutige Schlüssel definiert werden.

Stellen Spaltenfamilien, dokumentenorientierte und Graphdatenbanken die Zieldatenbank, so muss eine grundlegende Änderung der Struktur der Daten erfolgen. Dazu müssen, unabhängig von der Kategorie, aus der relationalen Datenbank alle relevanten Metadaten erhalten werden, um basierend darauf, die neue Struktur für die Zieldatenbank zu definieren.

Bei Spaltenfamilien ist das Ziel, zusammengehörige Daten aus den verschiedenen Tabellen zu fusionieren und in einer Tabelle darzustellen. Dabei entsteht eine verschachtelte Struktur, da Tabellen in Tabellen dargestellt werden. Li et al. [12] definiert drei Regeln, mittels denen das neue Schema für die Spaltenfamilien Datenbank HBase definiert wird. Soll eine komplette Verschachtelung aller Tabellen erzielt werden, so kann, mittels einer Abfrage, auf alle zugehörigen Informationen zugegriffen werden, dazu stellen Zhao et al. [27] und Lee et al. [11] jeweils ein Ansatz vor.

Graphdatenbanken bilden die Daten in einer Graphenstruktur ab, indem Tabellen zumeist in Knoten und Beziehungen in Kanten dargestellt werden. Zwischentabellen aus n:m-Beziehungen sollten dabei keinen eigenständigen Knoten bilden, sondern über Beziehungen realisiert werden. Orel et al. [18] definiert eine Reihe von Regeln, mittels denen entschieden wird, wie Tabellen und Beziehungen in Knoten und Kanten aufzulösen sind. Singh et al. [22] stellt einen Ansatz vor, bei dem alle Spalten als ein eigener Knoten dargestellt werden. Knoten mit Fremdschlüsseln, also dem Primärschlüssel einer anderen Tabelle, bilden Kanten zu diesen. Dabei entstehen Knoten mit eingehenden, ausgehenden Kanten oder ohne solche. Diese werden über fünf verschiedene Fälle zu Eigenschaften, Knoten oder Kanten definiert und somit die Graphenstruktur gebildet. Virgilio et al. [3] stellt ein Konzept vor, welches die Performance von Abfragen verbessern soll, indem Informationen, welche oftmals zusammen aufgerufen werden, in einem Knoten zusammengefasst werden. In einer weiterführenden Arbeit wird dazu ein Tool realisiert, dem R2G [4].

Im Folgenden werden die Konzepte für dokumentenorientierte Datenbanken detaillierter betrachtet, da der Schwerpunkt auf diesen liegen soll. Generell soll dabei die Tabellenstruktur aufgelöst werden und ähnlich den Spaltenfamilien Datenbanken eine Verschachtelung der Daten erzielt werden. Dazu ergeben sich nur zwei Möglichkeiten dies zu realisieren, dem Embedding und der Referenz. Beim Embedding werden Inhalte aus verschiedenen Tabellen zusammengefasst und die Inhalte der kleineren in das Dokument gezogen, womit ein Dokument in einem Dokument entsteht. Die Referenz hingegen hinterlegt in einem Dokument einen Verweis auf das Dokument, welches die Inhalte enthält. Dies ähnelt der relationalen Struktur mit den Beziehungen zwischen Tabellen. Welche der

beiden Optionen besser ist, hängt von verschiedenen Faktoren ab. Soll mit einer Abfrage auf möglichst alle zugehörigen Informationen zugegriffen werden, so ist dies mit Embedding zu erreichen, jedoch entstehen dabei Redundanzen, welche den benötigten Speicherplatz erhöhen. Erfolgen Abfragen auf unterschiedliche Bereiche und müsste somit auf stark verschachtelte Informationen zugegriffen werden, so kann dies mit der Referenz performant erzielt werden.

Die hier vorgestellten Konzepte sind exemplarisch und keine vollständige Literaturauswertung, da hierbei der Fokus auf der Varianz mit dem Umgang der Beziehungen und der Hinterlegung der Daten liegt.

Mit der prinzipiellen Realisierbarkeit einer Migration von einem relationalen zu einem dokumentenorientierten Modell hat sich Zhao et al.[26] befasst. Anhand der MongoDB wurde das relationale Modell nachgewiesen, sodass zumindest die MongoDB fähig ist, das relationale Modell abzubilden und somit eine Migration grundsätzlich realistisch ist. In der Folgearbeit [28] wird auf den Umgang mit dem Join bei Abfragen auf relationalen Datenbanken eingegangen. Mittels einer Graphenstruktur werden die Informationen aus den verschiedenen Tabellen in ein Dokument verschachtelt und auf dieser verschachtelten Struktur Abfragen formuliert. Den Erwartungen entsprechend, ist durch die verschachtelte Struktur die Abfragezeit kürzer, da nur ein Dokument für alle Informationen gefunden werden muss, wobei der Platzbedarf größer ist, da durch die Verschachtelung Redundanzen entstehen.

Bei Liang [14] werden die Daten in einem Zwischenmodell in Objekten hinterlegt. Bei Zwischenmodellen handelt es sich um Ansätze, welche explizit mittels einer Zwischenspeicherung der Daten, eine Erweiterung der Quell- und Zieldatenbanken vereinfachen sollen. Das Objekt ist dabei eine Entität, aus dem relationalen Modell, welches neben Eigenschaften, sowie Beziehungen der Entität, Informationen zu Daten und Abfragen enthält. Dieser Ansatz ist für alle NoSQL Datenbanken geeignet, da er sich darauf fokussiert, wie alle relevanten Informationen zu den Daten aus der Quelldatenbank hinterlegt werden. Der eigentliche Prozess der Umwandlung der Daten wird in sogenannten Strategien hinterlegt, welche jedoch nicht in dem Paper erläutert werden, obwohl exemplarisch gezeigt wird, wie die Struktur von relational zu Spaltenfamilien oder dokumentenorientierten Datenbanken aussehen könnte.

Ebenfalls kann über direkte, azyklische Graphen (DAGs) (vgl. [10]) die Struktur der relationalen Datenbank dargestellt werden, indem die Tabellen Knoten und die Beziehungen Kanten bilden. Jede Tabelle bildet einen eigenen DAG mit den Beziehungen, die diese Tabelle besitzt. Die DAGs werden für die Migration aufgelöst, bis die Wurzel alle DAGs enthält. Ob ein Embedding oder eine Referenz erfolgen soll, ist hierbei vorgegeben.

Bei dem Konzept von Hamouda [8] erfolgt eine Migration ausgehend vom ER-Modell. Dazu wird ein neues Schema definiert, das Document-Oriented Data Schema (DODS). Erfasst werden unter diesem sämtliche Informationen aus der relationalen Datenbank, sowie eine Zuweisung der Konzepte aus dem relationalen Modell zum dokumentenorientierten. Abschließend kann dann eine Migration

erfolgen, bei der der Umgang mit den verschieden Beziehungstypen, sowie die Entscheidung, ob Embedding oder Referenz erfolgen soll, vorher festgelegt ist.

Die Entscheidung zwischen Embedded und Referenz muss nicht zwingend durch die Autoren des Ansatzes festgelegt werden. So wird bei der MigDB [15] mittels Neuronaler Netzwerke eine Empfehlung gegeben, ob ein Embedding oder eine Referenz zu bevorzugen ist. Der Benutzer kann von der Empfehlung abweichen und die andere Methode verwenden. Wird eine andere Entscheidung getroffen, kann das Neuronale Netzwerk davon lernen und bessere Prognosen erstellen.

Die Arbeit von Yassine et al. [25] verdeutlicht, dass eine sinnvolle Verwendung von Embedded und Referenz nötig ist, um eine gute Performance zu erzielen. Dazu wurden basierend auf einem Beispieldatensatz drei verschiedene Strategien für die Verwendung von Embedding und Referenz erprobt. Im ersten Fall werden alle Tabellen in ein Dokument Embedded. Der zweite Fall kombiniert sinnvoll Embedding und Referenz und der dritte Fall arbeitet nur mit Referenzen. Anschließend wurde die Performance der drei Strategien mit einer relationalen Datenbank verglichen, indem vordefinierte Abfragen auf die Daten erfolgten. Zusammenfassend kann die dritte Strategie, welche das Schema einer relationalen Datenbank nachbildet, nicht mit dieser konkurrieren. Die Verwendung von Embedded und Referenz in Kombination erzielt im Schnitt die beste Performance. Ebenso ist nur die Verwendung von Embedding nicht so performant wie die Kombination beider Methoden.

Insgesamt vereint alle Ansätze, unabhängig von der Kategorie der NoSQL Datenbank, dass zunächst die wichtigen Informationen zu Struktur und Aufbau der Daten aus dem relationalen Modell erfasst werden. Basierend auf diesen Informationen kann das neue Schema für die Daten gebildet werden, um sie optimal in der Zieldatenbank zu hinterlegen. Dadurch, dass die Gewinnung der wichtigen Informationen unabhängig der Zieldatenbank erfolgt, ist dieser Schritt bei allen Ansätzen zu den NoSQL Datenbanken sehr ähnlich. Auf Grund dessen verweisen manche Paper darauf, dass der Ansatz ebenfalls für andere NoSQL Datenbanken verwendet werden kann. Wird von Anfang an auf einen speziellen Zieldatenbanktypen hingearbeitet, so können direkt die Besonderheiten des Typs berücksichtigt werden und somit eine gezieltere Migration der Daten ermöglichen.

Fast alle Ansätze verweisen darauf, dass der Benutzer während des Prozesses direkt mitbestimmen kann, ob ein Embedding oder eine Referenz erfolgen soll. Andernfalls kann die vorgeschlagene Struktur abschließend beurteilt werden. Dies ist ein sinnvoller Prozess, da der Benutzer dadurch nicht darauf angewiesen ist, ob der zugrundeliegende Algorithmus für die Migration optimal agiert und individuelle Veränderungen vornehmen kann.

3 Ausblick auf die Masterarbeit

In diesem Kapitel wird ein Ausblick auf das grundlegende Konzept der Masterarbeit gegeben. Das Thema der Arbeit befasst sich mit der Migration von

einer relationalen zu einer dokumentenorientierten, bzw. zwischen zwei dokumentenorientierten Datenbanken. Dabei sollen neben den Daten auch sämtliche Funktionalitäten migriert werden, welche sich in der Zieldatenbank realisieren lassen. Um eine Evaluation des eigenen Ansatzes zu ermöglichen, soll ein Tool erstellt werden, welches für ausgewählte Quell- und Zieldatenbanken funktioniert. Daraus ergibt sich folgende Forschungsfrage:

Wie kann unter Berücksichtigung der Funktionalitäten eine evaluierbare Migration zwischen relationalen und dokumentenorientierten Datenbanken erfolgen?

Im Weiteren wird auf die für die Masterarbeit relevanten Ergebnisse aus dem Hauptprojekt eingegangen (vgl. Kapitel 3.1). Anschließend wird ein möglicher Lösungsansatz vorgestellt (vgl. Kapitel 3.2), sowie in Kapitel 3.3 eine Abgrenzung gegeben.

3.1 Ergebnisse des Hauptprojekts

Im Rahmen des Hauptprojekts wurde ein Überblick über die Literaturlage zum Thema Migration bei NoSQL Datenbanken gegeben. Da keine Arbeiten zum Thema Migration zwischen NoSQL Datenbanken des gleichen Typs gefunden werden konnten, wurde dies exemplarisch im Hauptprojekt untersucht und dabei auf die Datentypen, Indexe, grundlegende Syntax der Abfragesprachen, sowie die Funktionalitäten eingegangen.

Berücksichtigt wurden fünf dokumentenorientierte Datenbanken (MongoDB, CouchDB[6], OrientDB[7], ArangoDB[8] und RavenDB[9]), welche zum Teil Multi-Modell Datenbanken sind, d.h. neben dem dokumentenorientierten Modell werden weitere unterstützt, z.B. unterstützt die OrientDB das dokumentenorientierte, Schlüssel-Wert-, Graph- und Objekt-Modell. Für einen direkten Datentransfer stehen zumeist JSON oder CSV als Import/Export Format zur Verfügung, welche jedoch nicht direkt zwischen den Datenbanken migriert werden können, denn unterschiedliche Datentypen oder erwartete Formatierungen der Daten in der Datei müssen berücksichtigt werden. Zum einen können sich die unterstützten Datentypen unterscheiden und zu Komplikationen führen. Beispielsweise ist die _id bei MongoDB eine ObjectID, welche nicht von CouchDB verarbeitet werden kann, da diese den Datentyp nicht unterstützt. D.h. ohne entsprechende Vorverarbeitung der Daten ist keine Migration über Import/Export möglich. Zum anderen unterstützt nicht jede Datenbank Datum und Zeit mit einem eigenen Datentyp. Teilweise liegt ein entsprechender Datentyp vor und erlaubt somit spezielle Operationen über diesen. Alternativ kann zwar kein Datentyp dafür vorliegen, aber trotzdem Datum und Zeit nach einem bestimmten Schema hinterlegt werden, um somit dennoch Operationen darauf anbieten zu können.

[6] https://couchdb.apache.org/
[7] https://orientdb.com/
[8] https://www.arangodb.com/
[9] https://ravendb.net/

Zum anderen müssen die Daten in einer anderen Struktur hinterlegt sein. So erwartet z.B. die MongoDB eine JSON-Datei, in der in jeder Zeile ein Dokument steht, die CouchDB hingegen erwartet ein Array von Dokumenten. Bei dem Transfer der Daten muss auf solche Besonderheiten geachtet werden.

Des Weiteren ergeben sich bei den Abfragen große Unterschiede. Prinzipiell unterteilt sich die Charakteristik in zwei Bereiche, den SQL-ähnlichen (OrientDB, ArangoDB und RavenDB) und denen mit einer eigenen Sprache, wie MongoDB oder CouchDB. Aufgrund der Vielfältigkeit der möglichen Operatoren wurden hier nur die Basisoperatoren verglichen. Zum Vergleich bietet die MongoDB, nach Anwendungsart kategorisiert, weit über 100 Operatoren [17] und die Anzahl und Art variiert bei den Datenbanken. Das Gleiche gilt für die Funktionalitäten, die eine Datenbank zur Verfügung stellt. Das bedeutet für die Masterarbeit, dass es keine allgemeine Erfassung aller Abfrageoperatoren oder Funktionalitäten gibt, so dass sich immer auf die ausgewählten Datenbanken beschränkt werden muss. Im Gegensatz zu den Funktionalitäten, gibt es eine breite Unterstützung von Indexen, welche prinzipiell eine Migration dieser ermöglichen.

3.2 Lösungsansatz

Die Basis für die Masterarbeit bilden die Erfahrungen aus dem Hauptprojekt, welches sich mit der Migration zwischen dokumentenorientierten Datenbanken befasst, sowie den Ergebnissen aus der Literaturrecherche, welche die Migration von einer relationalen zu einer dokumentenorientierten Datenbank thematisiert.

Wie Kapitel 2 entnommen werden kann, gibt es verschiedene Konzepte, wie die Daten aus dem relationalen Modell ausgewertet werden können und wie eine Entscheidung bezüglich Referenz oder Embedded erfolgen kann. Hierbei fehlt jedoch ein Vergleich der verschiedenen Konzepte, welcher in der Masterarbeit erarbeitet werden soll. Basierend auf diesen Kenntnissen kann ein eigener Ansatz für eine Migration erstellt werden, welcher dem Benutzer verschiedene Optimierungsziele zur Verfügung stellen soll, wie z.B. Platz- oder Zeitoptimierung.

Evaluiert werden soll der Migrationsprozess mit einem Tool. Für dieses Tool wird ein Datensatz benötigt, der eine Komplexität aufweist, welche einem realitätsnahen Szenario entspricht. D.h. der Datensatz muss eine angemessene Menge an Daten enthalten, unterschiedlichste Datentypen, die verschiedenen Beziehungstypen sollten vorhanden sein, sowie die entsprechenden Funktionalitäten, welche migriert werden können.

Nach einer Migration muss festgestellt werden, ob die Daten korrekt migriert worden sind und Abfragen das gleiche Ergebnis liefern. Das kann bei großen Datensätzen stichprobenartig mittels vordefinierter Abfragen erfolgen.

Das Tool sollte dann neben der reinen Migration der Daten ebenfalls die Funktionalitäten übertragen können, so wie es bereits bei der Migration zwischen relationalen Datenbanken der Fall ist.

3.3 Abgrenzung

Bei der Konzeption und dem Vorgehen der Migration von relational zu dokumentenorientiert, sowie zwischen dokumentenorientierten Datenbanken, handelt es sich lediglich um eine weitere Möglichkeit, wie dies zu realisieren wäre. Der Fokus des eigenen Ansatzes liegt darauf, dem Nutzer verschiedene Optimierungsziele zur Verfügung zu stellen und dahingehend das Konzept anzupassen. Das bedeutet jedoch nicht, dass dieses Konzept generell für jeden Datensatz bzw. für jede Migrationsaufgabe der einzige oder der optimale Weg ist. Wie der Datensatz im Idealfall in der neuen Struktur darzustellen ist, kann beispielsweise von der Nutzung, bzw. den Abfragen der Daten abhängig sein.

Die Migration wird nicht als eine Live-Migration durchgeführt, sondern es wird eine Momentaufnahme der Datenbank verwendet, welche dann migriert wird.

Das Tool, welches für die Evaluation erstellt wird, bietet nur für eine stark begrenzte Auswahl an Datenbanken eine Unterstützung an. Das Konzept zur Datenbankmigration ist grundlegend unabhängig von den verwendeten Datenbanken und somit Typ unabhängig, jedoch wird für das Tool eine genaue Analyse zu beispielsweise den Datentypen oder Funktionalitäten benötigt. Somit ist es ausschließlich für die ausgewählten Datenbanken der Masterarbeit verfügbar. Ebenso muss zu jeder Datenbank entsprechend die Schnittstelle zum Tool implementiert werden. Des Weiteren kann eine Versionsänderung bei einer der unterstützten Datenbanken zu Komplikationen bei dem Tool führen. So wäre z.B. bei einer Änderung der Schnittstelle ebenfalls eine Anpassung am Tool notwendig. Ein weiterer Faktor können Änderungen bei den Funktionalitäten sein, werden welche hinzugefügt oder entfernt, so können diese Änderungen nur mit einem durchgängigen Support des Tools weiterhin berücksichtigt werden. So wurde z.b. bei PostgreSQL erst mit Version 11 im Oktober 2018 die Funktion der Stored Procedures hinzugefügt [20]. Da die unterstützten Funktionalitäten von Datenbank zu Datenbank stark variieren können, werden lediglich jene berücksichtigt, welche von den ausgewählten Datenbanken realisiert wurden und zum Zeitpunkt der Bearbeitung der Masterarbeit zugänglich sind.

Da der zeitliche Rahmen bei der Bearbeitung der Masterarbeit begrenzt ist, wird bei die Realisierung des Tools der Support nur für eine kleine, begrenzte Anzahl an Datenbanken möglich sein.

4 Fazit

Bei der Literaturrecherche ist es nicht immer ersichtlich, für welche NoSQL Datenbank der entsprechende Ansatz geeignet ist. Wird beispielsweise in Titel, Abstrakt und Einleitung noch generell von NoSQL Datenbanken gesprochen, so wird erst bei der Vorstellung des Modells für die Migration auf den konkreten Datenbanktyp eingegangen. Dies erweckt zunächst den Eindruck, dass diese Ansätze generell für alle NoSQL Typen geeignet sind und erschwert es, die Ansätze für den gesuchten Typ zu finden.

Obwohl eine breite Varianz an Konzepten zur Migration von relational zu dokumentenorientiert vorhanden ist, gibt es keinen Vergleich zwischen diesen in Bezug auf ihre Performanz. Dies erschwert es, einzuschätzen, wie gut die Ansätze im Vergleich performen, welche Stärken oder Schwächen diese besitzen.

Auch bei der Migration zwischen den selben NoSQL Datenbankmodellen gilt es, alle Aspekte und Funktionen der Datenbank zu untersuchen, da zwar der Typ gleich ist und somit das Schema der Daten nicht grundlegend geändert werden muss, aber die Architektur und Funktionalitäten von Datenbank zu Datenbank variieren.

Ein Ausblick auf die Masterarbeit wurde gegeben, welcher ein erstes Konzept zur Realisierung bietet und in Zukunft an der HAW als ein Masterthema durchgeführt werden soll.

Literatur

1. Bansel, A., González-Vélez, H., Chis, A.E.: Cloud-Based NoSQL Data Migration. In: 2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP). pp. 224–231. IEEE (2016)
2. DB-Engines: DB-Engines Ranking, https://db-engines.com/de/ranking, (Zuletzt aufgerufen am: 26.02.2020)
3. De Virgilio, R., Maccioni, A., Torlone, R.: Converting Relational to Graph Databases. In: First International Workshop on Graph Data Management Experiences and Systems. pp. 1–6. ACM (2013)
4. De Virgilio, R., Maccioni, A., Torlone, R.: R2G: a Tool for Migrating Relations to Graphs. In: EDBT. pp. 640–643 (2014)
5. El Alami, A., Bahaj, M., Khourdifi, Y.: Supply of a key value database redis in-memory by data from a relational database. In: 2018 19th IEEE Mediterranean Electrotechnical Conference (MELECON). pp. 46–51. IEEE (2018)
6. Geisler, F.: Datenbanken: Grundlagen und Design. mitp Verlags GmbH & Co. KG (2014)
7. Győrödi, C., Győrödi, R., Pecherle, G., Olah, A.: A comparative study: MongoDB vs. MySQL. In: 2015 13th International Conference on Engineering of Modern Electric Systems (EMES). pp. 1–6. IEEE (2015)
8. Hamouda, S., Zainol, Z.: Document-Oriented Data Schema for Relational Database Migration to NoSQL. In: 2017 International Conference on Big Data Innovations and Applications (Innovate-Data). pp. 43–50. IEEE (2017)
9. Jung, M.G., Youn, S.A., Bae, J., Choi, Y.L.: A Study on Data Input and Output Performance Comparison of MongoDB and PostgreSQL in the Big Data Environment. In: 2015 8th International Conference on Database Theory and Application (DTA). pp. 14–17. IEEE (2015)
10. Kuszera, E.M., Peres, L.M., Fabro, M.D.D.: Toward RDB to NoSQL: Transforming Data with Metamorfose Framework. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. pp. 456–463. ACM (2019)
11. Lee, C.H., Zheng, Y.L.: SQL-to-NoSQL Schema Denormalization and Migration: A Study on Content Management Systems. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics. pp. 2022–2026. IEEE (2015)
12. Li, C.: Transforming relational database into HBase: A case study. In: 2010 IEEE international conference on software engineering and service sciences. pp. 683–687. IEEE (2010)

13. Li, Y., Manoharan, S.: A performance comparison of SQL and NoSQL databases. In: 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM). pp. 15–19. IEEE (2013)
14. Liang, D., Lin, Y., Ding, G.: Mid-model Design Used in Model Transition and Data Migration between Relational Databases and NoSQL Databases. In: 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity). pp. 866–869. IEEE (2015)
15. Liyanaarachchi, G., Kasun, L., Nimesha, M., Lahiru, K., Karunasena, A.: MigDB-relational to NoSQL mapper. In: 2016 IEEE International Conference on Information and Automation for Sustainability (ICIAfS). pp. 1–6. IEEE (2016)
16. Meier, A., Kaufmann, M.: SQL-& NoSQL-Datenbanken. Springer (2016)
17. MongoDB: Operators, https://docs.mongodb.com/manual/reference/operator/, (Zuletzt aufgerufen am: 26.02.2020)
18. Orel, O., Zakošek, S., Baranovič, M.: Property Oriented Relational-To-Graph Database Conversion. automatika **57**(3), 836–845 (2016)
19. Parker, Z., Poe, S., Vrbsky, S.V.: Comparing NoSQL MongoDB to an SQL DB. In: Proceedings of the 51st ACM Southeast Conference. p. 5. ACM (2013)
20. PostgreSQL Global Development Group: PostgreSQL 11 Released!, https://www.postgresql.org/about/news/1894/, (Zuletzt aufgerufen am: 26.02.2020)
21. PostgreSQL Wiki: Converting from other Databases to PostgreSQL, https://wiki.postgresql.org/wiki/Converting_from_other_Databases_to_PostgreSQL, (Zuletzt aufgerufen am: 26.02.2020)
22. Singh, M., Kaur, K.: SQL2Neo: Moving health-care data from relational to graph databases. In: 2015 IEEE International Advance Computing Conference (IACC). pp. 721–725. IEEE (2015)
23. Statista: Ranking of the most popular database management systems worldwide, as of September 2019*, https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/, (Zuletzt aufgerufen am: 26.02.2020)
24. Wijaya, Y.S., Arman, A.A.: A Framework for Data Migration Between Different Datastore of NoSQL Database. In: 2018 International Conference on ICT for Smart Society (ICISS). pp. 1–6. IEEE (2018)
25. Yassine, F., Awad, M.A.: Migrating from SQL to NoSQL Database: Practices and Analysis. In: 2018 International Conference on Innovations in Information Technology (IIT). pp. 58–62. IEEE (2018)
26. Zhao, G., Huang, W., Liang, S., Tang, Y.: Modeling MongoDB with Relational Model. In: 2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies. pp. 115–121. IEEE (2013)
27. Zhao, G., Li, L., Li, Z., Lin, Q.: Multiple Nested Schema of HBase for Migration from SQL. In: 2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. pp. 338–343. IEEE (2014)
28. Zhao, G., Lin, Q., Li, L., Li, Z.: Schema Conversion Model of SQL Database to NoSQL. In: 2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. pp. 355–362. IEEE (2014)