Projekt NoSQL

Author:	Lukas Lieb
	Daniel Schmid
	Matthias Eiholzer
E-Mail:	lukas.lieb@stud.hslu.ch
	daniel.schmid@stud.hslu.ch
	matthias.eiholzer@stud.hslu.ch

Erstellungsdatum:	20. Dezember 2016

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

1.1 Aufgabenstellung

Ziel dieser Arbeit ist es, die Uni-DB, welche auf einer SQL Server liegt, in eine NoSQL-Datenbank zu implementieren. Dazu Wird das ER-Schema der Uni-DB so umgezeichnet werden, dass es in der gewählten NoSQL-Datenbank abgebildet werden kann. Dieses Schema wird dann in der entsprechend gewählten Datenbank umgesetzt. Dazu migriert man die Daten aus der SQL- in die NoSQL-Datenbank. Anschliessend setzt man das vorgegebene Querry (siehe Aufgabenstellung) so um, dass es in der entsprechend gewählten Datenbank benutzt werden kann. Dazu schreibt man eine Abfrage oder erstellt ein einfaches GUI.

1.2 Datenbankauswahl

Bei der Auswahl der Datenbank haben wir uns für Die MongoDB entschieden. Entscheidend waren dabei folgende Gründe:

- Wird in der Praxis eingesetzt
- Bekannt
- Gute Dokumentation
- Kostenlos
- Unterstützung durch Forenuser
- Erste Erfahrung vorhanden

Die MongoDB ist eine Datenbank, welche sich grundlegend im CAP Theorem in der spalte CP aufhält. Dies bedeutet, dass die Konsitenz (Consistency) gewährleistet ist. Dies wird realisiert, indem in einem System ein Knoten als primäres Mitglied fungiert, und alle Anfragen über diesen Knoten abgearbeitet werden. Zusätzlich wird die Partition-Tolerance gewährleistet. Dies Funktioniert laut Vertreiber, indem wenn der Primäre Knoten ausfällt, automatisch ein sekundärer Knoten als neuer Primärknoten Definiert wird und somit das System wieder funktioniert. Ebenfalls soll es möglich sein, auf Kosten der Konsistenz (Consistency) die Verfügbarkeit (Availaility) zu erhöhen, indem man in den Einstellungen die Konfiguration vornimmt, dass Daten nicht bloss über den Primär-Knoten, sondern auch über Sekundärknoten bezogen werden können.

2 Datenmanagement

Zwei Aktoren wirken auf die Datenbank ein. Einerseits der Benutzer, welcher die Informationen abfragen kann. Andererseits der DB-Administrator, welcher die Daten abfragen, aber auch verändern und neue hinzufügen kann. Diese Anwenungsfälle sind in der Abbildung **??** abgebildet.

Welche Daten werden integriert migriert und wie werden sie aufbereitet? Wie interagiert der Benutzer mit der Datenbank?

3 Datenmodellierung

MongoDB ist eine dokumentorientierte Datenbank. Dabei werden die Daten in JSON ähnlichen Dokumenten verwaltet. Das bedeutet, dass die Daten nicht relational verwaltet werden. So kann zum Beispiel in Tupel einer relationalen Datenbank als Dokument in der dokumentorienterten Datenbank abgebildet werden. Die Attribute und die dazugehörigen Werte werden dabei in Schlüssel-Wert Paare abgebildet.

Unserem Projekt liegt das ER-Schema aus 1 zugrunde. Da MongoDB eine NoSQL und keine relationale Datenbank ist, kann das ER-Schema nicht direkt in dieser Form in der Datenbank abgebildet werden.

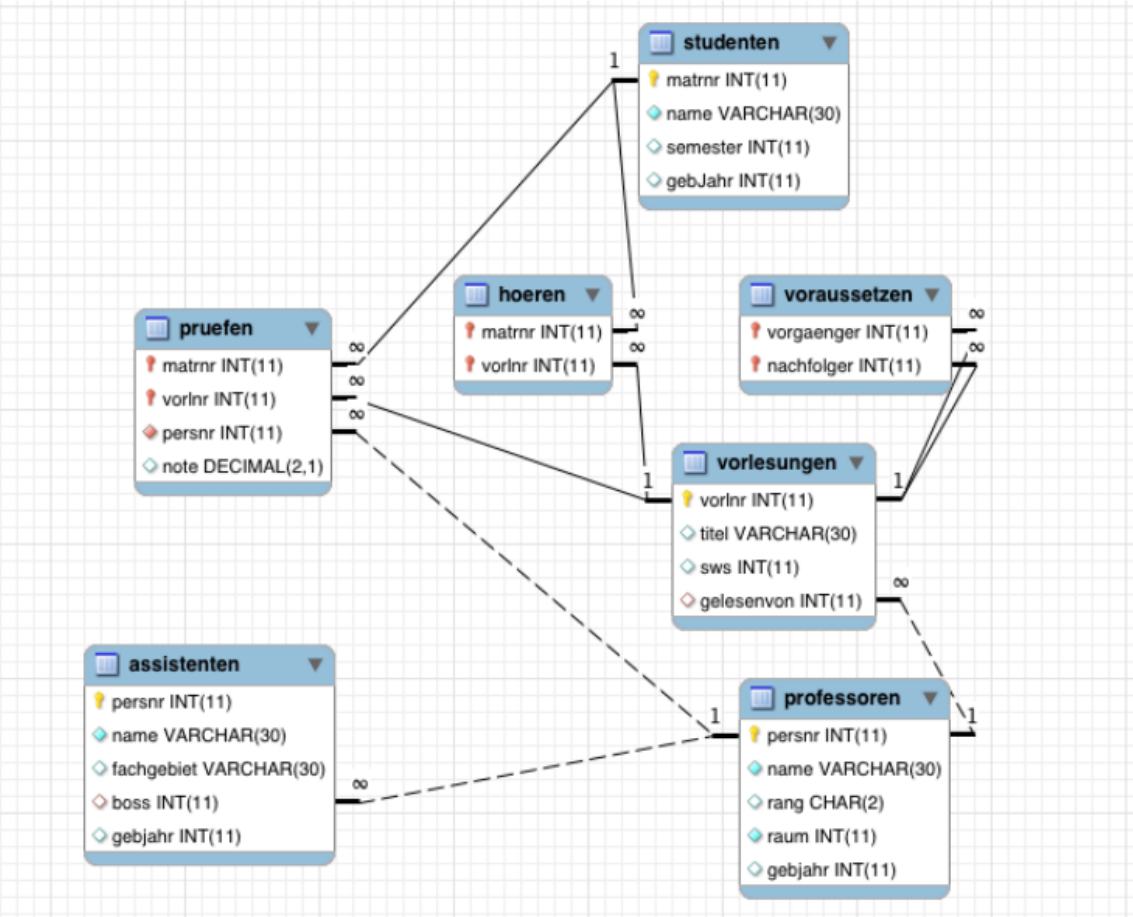

Abbildung 1: ER-Diagramm zur Uni-DB Kaufmann (2016)

Eine Komposition bedeutet, dass z.B in der NoSQL Datenbank als “teil von” abgebildet wird. Eine Aggregation bedeutet, dass diese in Beziehung als Referenz abgebildet wird.

```
{
"Legi": 25403,
"Name" : "Jonas",
"Semester" : 12,
"Hören" : [5032, 1910],
}
```

Im ER-Diagramm wird zum Beispiel die Komplex-Komplexe-Beziehung zwischen Vorlesungen und Studenten als eigene Tabelle abgebildet. In unserer Datenbank wird diese Beziehung in zwei einfach-komplexe Beziehung abgebildet. Somit besitzen Studenten eine Einfach-Komplexe Beziehung zu den Vorlesungen. Diese Beziehung wird im JSON -Format im Feld Hören ersichtlich. Die vom Studenten besuchten Vorlesungen werden als Vorlesungsnummer in einem Array referenziert.

4 Datenbanksprachen

5 Konsistenzsicherung

Bei relationalen Datenbanken bietet die Datenbank Mechanismen zur Konsistenzsicherung an. Dies wird durch verschiedene Eigenschaften im RDBMS erreicht. Jeder Tabelle liegt ein Schema zu Grunde. In diesem wird defniniert, wie die Daten strukturiert sein müssen. Zum Beispiel wird festgelegt, welches der Primärschlüssel ist, ob Felder "null" sein dürfen, oder ob Tupel gelöscht werden dürfen, wenn noch Referenzen darauf zeigen. MongoDB ist Schema frei. Das Bedeutet, dass sich je zwei Dokumente in einer Sammplung komplett in ihrer Struktur voneinander unterscheiden können. Desweitern garantiert MongoDB keine integrität der Referenzen. Dies wird klar, wenn man verlgeicht, welchen Regeln die relationale Datenbank und die MognoDB unterliegen. RDBMS unterliegen den ACID regeln.

- Atomar: Die gesamte Transaktion wird ausgeführt oder die Transaktion wird rückgängig gemacht.
- Consistent: Nach jeder Transaktion muss die Datenbank widerspruchsfrei sein.
- Isolatet: Bei einer Transaktion dürfen keine Seiteneffekte auftretreten. Dies garantiert, dass eim Mehrbenutzerbetrieb möglich ist.
- Dauerhaft: Die Daten werden sicher gespeichert, auch bei Systemabstürzen. Bei einem Systemabsturz wird ein recovery durchgeführt, so dass danach die Datenbank wieder in einem konsistenten Zustand ist.

Die in unserem Projekt eingesetzte MongoDB unterliegt den BASE Regeln.

- Basically Available: Die Datenbank sollte meistens laufen.
- Eventually Consistent: Die Konsistenz der Daten wird nicht unmittelbar nach der Operation gewährleistet. Sie kann verzögert eintreten.

Da MongoDB keine Transaktionen(eine Folge von Operationen, die Atomar ausgeführt werden)) unterstützt, muss diese Funktionalität auf der Anwendungsebe implementiert werden. Dies ist beim Einsatz eines RDBMS nicht notwendig, da dieses Transaktionen unterstützt. In unserem Fall werden keine Transaktionen verwendet,

da nur lesend auf die Daten zugegriffen werden. Da nur gelesen wird, und keine Daten geändert oder hinzugegügt/entfernt werden, kann die Datenbank durch Operationen nicht in einen inkonsistenen Zustand überführt werden. Deswegen kann auch parallel auf die Datenzugegriffen werden ohne die Konsistzen zu verlieren. Sollen zu einem späteren Zeitpunkt zur Laufzeit der Anwendung Daten geändert oder hinzugefügt/entfernt werden, so muss eine Konsistenzsicherung auf der Anwendungsebene implementiert werden.

6 Systemarchitektur

MongoDB besteht aus drei verschieden Servern. Den Routern, Conig Servern, und den Replica Sets. Jeder dieser drei Servertypen hat eine bestimmte Aufgabe. Der Cleint sendet seine Operation an den Router. Der Router wiederum leitet die Operation an die zuständigen Replica Sets weiter. Die Replica Sets wiederum speichern di zu verarbeitenden Dokumenten. Damit der Router weiss, an welche Replica Sets er die Anfrage weiterleiten muss, stellt er wiederum eine Anfrage an die Config Server. Diese Antworten entsprechen. Die Config Server enthalten also die Metadaten über die Dokumentenverteilung in den Replica Sets. In der Abbildung **??** ist die Architektur von MongoDB visualisiert. In der in diesem Projekt eingesetzte MongoDB Instanz, laufen alle drei Server auf dem selben physischen Rechner. Dies wurde so gewählt, da Sammlung an Dokumenten klein ist. Desweiteren wird auch nur sporadisch auf die Daten zugegriffen. Sollte sich in Zukunft einer dieser Parameter ändern, kann darüber nachgedacht werden, die Server auf verschiedene physischen Server zu verteilen. Dies hätte auch den weiteren Vorteil, dass Parallel auf die Dokumente zugegriffen werden kann. Ist die Replica Set 1 mit der Verarbeitung einer Anfrage beschläftigt, kann währenddem die Replica Set 2 die nächste Anfrage bearbeiten, sofern es sich bei der Anfrage um die in ihr gespeicherten Dokumente handelt. In RDBMS liegen die Tabellen in normalisierter Form vor. Daten die nicht funktional vom Primärschlüssel abhängen, werden als Fremdschlüssel referenziert. Die Referenzierung macht es sehr schwierig, dass die Tabelle, in welcher das referenzierte Tupel abgelegt ist, auf einen anderen Server auszulagern. Denn bei jedem Zugriff auf die ausgelagerte Tabelle, wird die Abfrage der Daten verlangsamt, da zwischen den beiden Servern Kommunikation stattfindet. Dieses Problem hat MongoDB nicht, da die Daten nicht normalisiert vorliegen müssen. Bei MongoDB dürfen die bei RDBMS referenzierten Tupel als Aggretationen in die Dokumente umgesetzt werden. Dies vermindert die Anzahl der Referenzierungen und ermöglicht es damit, das eine horizontale Skalierung viel einfacher zu bewerkstelligen ist, als bei einer relationalen Datenbank. Dies führt natürlich auch Nachteile mit sich. Jedes mal wenn ein referenziertes Tupel als Aggregation in ein Dokument gespeichert wird, erhöhrt sich die

Datenmenge. Desweitern bedeutet es mehr Aufwand ein solches aggregiertes Tupel zu ändern, da die Änderung bei allen Dokumenten gemacht werden muss, bei denen das referenzierte Tupel als Unterteil gespeichert wird. Dies im Gegensatz zur relatiolen Datenbank. Bei dieser muss die Änderung nur an einem Ort stattfinden. MongoDB kennt keine Funktionlität zum Auflösen von Referenzen. Diese muss jeweils in der Applikation implementiert werden. Da wie bereits erwähnt, alle Server auf einem physischen Rechner laufen, ist die Datenbank bei einem Systemausfall nicht mehr erreichbar. Bei einem Festplattenausfall kann es sogar passieren, dass die komplette Datenbank verloren geht, da es in der Replica Set nur einen Server gibt, und somit ein weiter Server fehlt, der ein Replikat der Datenbank enthält.

7 Schlussfolgerungen

Literatur

Kaufmann, Michael., "Übung S1: SQL Grundlagen," 2016.