GYMNASIUM
Ursulaschule

Relationale und nicht relationale Datenbanken

ein Vergleich

Seminararbeit

im Seminarfach Webentwicklung

an der Ursulaschule Osnabrück

vorgelegt am: 22.11.2021

von: Moritz Mithöfer

Ursulaschule Osnabrück

Kleine Domsfreiheit 11, 49074 Osnabrück, Deutschland

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abkürzungen

DBMS Datenbankmanagmentsystem

DCL Data Control Language

DDB Dokument-Datenbank

DDL Data Definition Language

DML Data Manipulation Language

JSON JavaScript Object Notation

kvDB key-value-Datenbank

nRDBMS nicht relationales Datenbankmanagmentsystem

RDBMS relationales Datenbankmanagmentsystem

SQL Structured Query Language

1 Einführung

1.1 Datenmengen in unserer Welt

Mittlerweile erhebt fast jedes Gerät Daten. Sei es die Waschmaschine, die sich über Smart Home Dienste aus der Cloud steuern lässt, die Maschinen in der Industrie, die penibel sämtliche Daten, wie Produktionsmengen, Temperaturen oder Ausschussmengen festhalten oder unser Smartphone, das unsere Suchanfragen oder Bewegungsprofile speichert. Dazu kommen Daten die, die Menschen bewusst erstellen, wie zum Beispiel Filme, Fotos oder Dokumente wie Schriftstücke oder aber auch Präsentationen. So hat ein durchschnittlicher Deutscher 1070 Fotos auf seinem Smartphone. Damit ist Deutschland auf Platz 8 weltweit. [21] Dadurch das immer mehr Daten gespeichert werden verdoppelt sich der Datenbestand alle zwei Jahre. 2020 lag er bei etwa 40 Zettabytes. Bis 2025 sollen es schätzungsweise 163 Zettabytes werden.

> „40 Zettabytes – das entspricht nach Schätzungen der Forscher 57-mal der Menge an Sandkörnern aller Strände der Erde." [14]

Zudem basiert unsere Gesellschaft mehr und mehr auf den erhobenen Daten. Unternehmen treffen wichtige Geschäftsentscheidungen auf Basis der erhobenen Daten ihrer Produktion, Amazon schlägt Artikel basierend auf dem eigenen Kaufverhalten oder dem anderer Kunden vor und Google Maps berücksichtigt bei der Routenführung den voraussichtlichen Verkehr auf der Strecke. [7]

1.2 Was sind Datenbanken und warum brauchen wir sie?

Datenbanksysteme sind Programme zur digitalen Verwaltung von Daten. Ein Datenbanksystem sorgt für eine effizient, dauerhafte und fehlerfreie Speicherung der Daten und stellt sie bei Bedarf zur Verfügung.
Datenbanken werden benötigt, wenn ein oder mehrere Programme Zugriff auf einen zentralen Datensatz brauchen.

2 Varianten von Datenbanken

Für verschiedene Anwendungsbereiche bedarf es unterschiedlicher Strukturierung der Daten.
So gibt es zum Beispiel Anwendungsfälle bei denen eine feste Struktur der Daten vorauszusehen ist. So erfasst eine Maschine zum Beispiel für jede Schicht die Anzahl der verpackten Gutteile, den Ausschuss und das verarbeitete Material. In einem anderen Anwendungsfall weisen die Datenpunkte beispielsweise eine sehr hohe oder sehr niedrige bis nicht vorhandene Anzahl an Verknüpfungen untereinander auf. Deshalb unterscheidet man zwei grundlegende Kategorien von Datenbanksystemen. Die **relationalen** und die **nicht relationalen** Datenbanken.

2.1 SQL - relationale Datenbanken

Bei den weitverbreiteten relationalen Datenbanken werden Daten nach einem tabellarischen Schema abgelegt, das heißt es werden Daten basierend auf Schlüsseln in Verbindung gebracht. Durch ein sorgfältig gewähltes Schema werden Redundanzen vermieden und eine effiziente Speicherung ermöglicht. SQL-Datenbanken eignen sich folglich besonders gut für Daten, die eine logische Verknüpfung zueinander aufweisen. Die Datenbank läuft auf einem einzigen Server, sprich, um die Leistung der Datenbank zu erhöhen, muss die Leistung des Servers erhöht werden. Man spricht hierbei von vertikalem Skaling. [3] [2]

2.2 NoSQL - nicht relationale Datenbanken

NoSQL*, zu Deutsch „Nicht nur SQL“,* beschreibt eine nicht relationale Speicherung von Daten. Die Speicherung beruht dabei nicht auf einem festen Tabellenschemata. Unterkategorien der nicht realtionalen Datenbankmanagmentsysteme (nRDBMSs) sind dokumentbasierte, key-value basierte oder aber auch graphbasierend Datenbanken. Im Gegensatz zu SQL-Datenbanksystemen können NoSQL sehr gut mit großen Datenmengen und häufigen Zugriffen umgehen. NoSQL Datenbanken sind dabei horizontal skalierbar, das heißt die Datenbank läuft auf mehreren Servern, auch Nodes genannt. So lässt sich die Last verteilen, eine Ausfallsicherheit gewährleisten oder die Leistung der Datenbank durch einfaches hinzufügen einer

Node verbessern. [3] [2]

3 SQL-Datenbanken

3.1 Relationales Modell

SQL-/relationale Datenbanken basieren auf dem dem Menschen geläufigen Prinzip von Tabellen. Zum Speichern von Daten wird ein entsprechendes Tabellenschema erstellt. Aus mathematischer Sicht ist eine Relation eine Teilmenge aus dem kartesischen Produkt der Wertebereiche der Attribute:
$R \subseteq D_1 \times D_2 \times D_3 \times ... \times D_n$, wobei D_m der Wertebereich (*engl. domain*) des m-ten Attributs ist. Folglich ist ein Tupel in mathematischer Sicht eine Menge an Werten aus den Wertebereichen der Attribute $t = (d_1, d_2, d_3, ..., d_n)$. [2] In dem folgenden Beispiel sollen Informationen über Schüler gespeichert werden. Folgende Informationen sollen gespeichert werden: Name, Vorname und die Klasse. Jeder Datensatz muss im relationalen Modell eindeutig, mittels eines Schlüssels, referenzierbar sein. Da es aber sein kann, dass zwei identisch heißende Schüler die selbe Klasse besuchen, wird ein zusätzliches, fiktives Attribut einem jeden Schüler zugeordnet, um in eindeutig referenzieren zu können, die Schüler-Nr.

Schüler-Nr.	Name	Vorname	Klasse
1	Müller	Liara	10f
2	Hausmann	Heinrich	5a
...	...	...	...

Tabelle 1: SCHÜLER

[2]

3.1.1 Schlüssel

Ein Schlüssel ist ein Merkmal oder die kleinstmögliche Merkmalskombination, die es erlaubt, einen jeden Datensatz einer Tabelle eindeutig zu referenzieren. Häufig werden hierfür fiktive Attribute wie eine laufende Nummer genutzt.

3.1.2 Attribut

Ein Attribut ordnet jedem Datensatz, auch Tupel genannt, einen Wert zu. Dieser Wert stammt dabei aus dem vorher definierten Wertebereich.

3.1.3 Tupel

Ein Tupel enthält alle Datenpunkte eines Datensatzes. Ein Beispiel eines Tupels aus Tabelle 1 ist:
1 Müller Liara 10f

3.1.4 Relation vs. Tabelle

Im relationalen Modell ist es wichtig zwischen Tabelle und Relationen zu unterscheiden. So ist jede Relation als Tabelle darstellbar, aber nicht jede Tabelle stellt eine Relation da. Zudem können mehrere Tabellen die gleiche Relation darstellen. [2]

Klasse	Name	Vorname	Schüler-Nr.
5a	Hausmann	Heinrich	2
10f	Müller	Liara	1
...	...	...	...

Tabelle 2: SCHÜLER

56	86	4	78
1	45	56	456
2	7	56	45 ; 2
...	...	...	...

Tabelle 3: Zahlen

So stellen die Tabelle 1 und die Tabelle 2 die selbe Relation da, obwohl es sich um zwei verschiedene Tabellen handelt. Die Reihenfolge der Spalten und Reihen ist irrelevant. Tabelle 3 stellt allerdings keine Relation da. [6]
Folgende Anforderungen müssen von einer Relation erfüllt sein:

- Jede Relation hat eine Kopfzeile, bestehend aus den Attributen und eine Body, eine Menge an Tupeln.
- In einer Relation kann eine Tupel nur maximal einmal vorkommen.
- Die Tupel und Attribute einer Relation sind nicht geordnet.

- Relationen liegen immer in der ersten Normalenform vor. Das heißt, dass in jeder Attributsposition sich genau ein Wert befindet.
- Über ein Schlüsselattribut oder eine Schlüsselattributskombination lässt sich jeder Tupel eindeutig referenzieren.

3.1.5 ACID Kriterien

Ein relationales Datenbankmanagmentsystem (RDBMS) verfolget die ACID Kriterien. Diese stellen bestimmte Anforderungen an Änderungsprozesse in Datenbanken.

- **Atomicity:** Jede Transaktion/Änderungsprozess ist atomar. Das heißt jede Transaktion muss vollständig ausgeführt werden, damit das System weiter funktioniert.
- **Consistency:** Wenn durch eine Transaktion kein gültiger Zustand erreicht wird, wird der alte wiederhergestellt.
- **Isolation:** Jede Transaktion läuft autark ab. Somit sind Transaktionen verschiedener Anwender voneinander isoliert.
- **Durability:** Die Daten sind persistent, das heißt sie bleiben dauerhaft gespeichert.

[15] [1]

3.2 SQL

Structured Query Language (SQL) ist die Standardsprache für relationale Datenbanken. Mit ihr lassen sich Informationen abfragen, erstellen, verändern und löschen. Die SQL-Befehle lassen sich in drei Kategorien aufspalten:

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)

3.2.1 Syntax

- Alle SQL Befehle sind alpha-numerisch, berücksichtigen keine Groß- und Kleinschreibung und beginnen mit einem Buchstaben. Sollten zum Beispiel Tabellennamen andere Zeichen enthalten, so sind sie in doppelten Anführungszeichen zu schreiben.
- Reihen-, Spalten- , Tabellen- und Datenbanknamen werden getrennt betrachtet. Eine Spalte kann folglich den selben Namen haben wie die ihr zugehörige Tabelle.
- Kommentare werden durch /* ... */ gekennzeichnet. Mit '–' lässt sich eine ganze Zeile kommentieren.
- SQL Befehle werden in der Regel in Großbuchstaben geschrieben.
- SQL-Statements enden mit einem ';'.

SQL untersützt einfache Datentypen wie: INT2 (2 bytes), INT4(4 bytes), Float4 (4 bytes), Float8 (8 bytes) CHAR(n), wobei n die fixe Länge des Strings ist, bei dem die verbleibenden Zeichen mit Leerzeichen aufgefüllt werden, VARCHAR(n), Strings mit variabler Länge mit einem Maximum von n Zeichen. Der Inhalt eines Strings ist in einfachen Anführungszeichen anzugeben. Zudem gibt es Datentypen für Datum, Zeit und Datum und Zeit.

Häufig genutzte SQL Befehle sind:

CREATE TABLE	erstellen einer leeren Tabelle
DROP TABLE	löschen einer Tabelle und allen zugehörigen Tupeln
ALTER TABLE	ändern der Struktur einer Tabelle
INSERT	Zeilen oder Ausschnitte einer Zeile zu einer Tabelle hinzufügen
SELECT	Abfrage bestehender Daten
UPDATE	Änderung bestehender Daten
DELETE	Löschen bestehender Daten
ROLLBACK	rückgängigmachen einer Änderung
COMMIT	permanentes Speichern aller angestoßenen Änderungen
COPY	kopieren von Daten zwischen Tabellen

Eine mögliche Beispieloperation wäre somit:

```
SELECT name FROM schueler
WHERE jahrgang > 5;
```

4 NoSQL-Datenbanken

4.1 Historischer Hintergrund

Der Begriff NoSQL wurde 1998 zuerst von Carlo Strozzi für sein RDBMS Strozzi NoSQL verwendet. Der Begriff sollte sein Datenbankmanagmentsystem (DBMS) von anderen Systemen abgrenzen, die SQL als Interfacesprache nutzten. Im Gegensatz zu den DBMS, die wir heute als No-SQL bezeichen, beruhte Strozzis DBMS jedoch auf dem relationalen Modell [20].

4.2 Einführung

Das große Problem von RDBMS ist die Skalierbarkeit. Die vielen Sicherheits- und Konsistenzfunktionen von SQL sind zwar für einige Anwendungen nötig, stellen aber auch hohe Leistungsanforderungen an das System. Das wird vor allem kritisch wenn man, wie bei z.B. bei Social Media Plattformen oder großen Websiten mit sehr vielen Usern, Datenmengen im Petabyte-Bereich oder größer verarbeiten muss. Im folgenden werde zwei häufig genutze Unterformen der NoSQL-Datenbanken vorgestellt.

4.3 key-value-Datenbanken

Eine einfache und fast allen Entwicklern geläufige Möglichkeit zur Datenspeicherung ist eine Variable. In key-value-Datenbank (kvDB) wird einem key, vergleichbar mit dem Variablennamen in einer Programmiersprache, ein value/Wert zugeordnet. Dabei ist eine kvDB vollkommen frei von festgelegten Strukturen. Eine Vernestung/Verschachtelung oder Referenzen werden allerdings nicht unterstützt. Dadurch können Daten völlig frei gespeichert werden, was eine performante Abfrage und höchste Flexibilität bietet. Allerdings birgt dies, bei schlechter Organisation, auch das Risiko, dass die Datenbank unübersichtlich wird. Im folgenden Beispiel werden die Daten zweier Schüler gespeichert:

```
db.schueler.save( { schueler_no: "123456", abiturdatum: "12/12/2123",
jahrgang:7 } );
db.schueler.save( { schueler_no: "789", abiturdatum: "12/12/1989",
name:"Simon" } );
```

Die Daten lassen sich mittels einfacher Abfragen abrufen. In dem folgenden Beispiel werden alle Datensätze abgerufen, dessen name Simon ist:

```
db.restaurants.find( {"name" : "Simon" } );
```

4.3.1 Eigenschaften

Eine kvDB efüllt folgende Eigenschaften:

- Eine Menge von referenzierbaren Datenobjekten, den Schlüsseln, ist festgelegt.
- Zu jedem Schlüssel gibt es immer genau ein Datenobjekt, das den dem Schlüssel zugeordneten Wert repräsentiert.
- Mittels des Schlüssels lässt sich der zugehörige Wert abrufen.

kvDB sind sehr beliebte NoSQL-Datenbanken. Das liegt vorallem daran, dass sie sehr gut skalierbar sind. Zudem müssen keine Integritätsbedingungen überprüft werden, was eine höchst performante Datenspeicherung und Abfrage ermöglicht. Um kvDB noch performanter zu gestalten, wird auf ein Verbund von Caching und Sharding gesetzt.
Beim Caching werden regelmäßig oder kürzlich abgefragte bzw. gespeicherte key-value-Paare im RAM gespeichert bevor sie in den langfristigen und persistenten Hintergrundspeicher abgelegt werden.
Um die Datenbank horizontal, das heißt über mehrere Servernodes, zu skalieren wird die Datenbank in einzelne Teile aufgeteilt. Dieser Prozess wird Sharding genannt. Eine Node, auch Shard genannt, ist dabei für einen festgelegten Raum der Schlüssel zuständig.

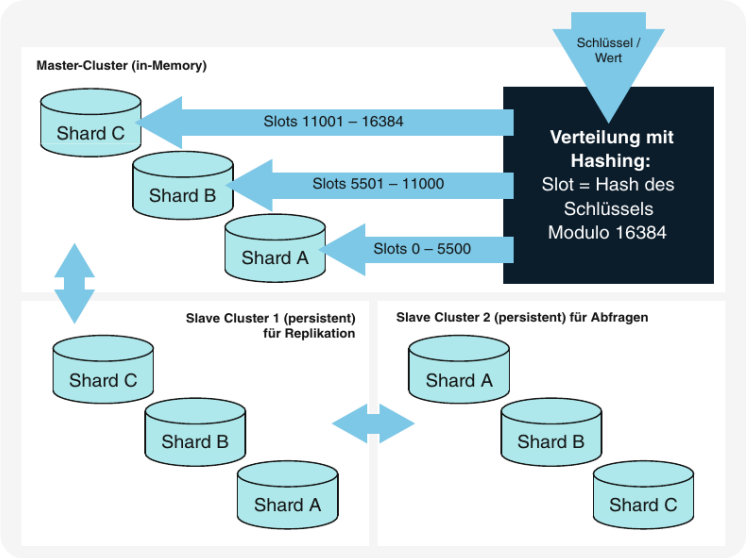

Abbildung 1: verteilte key-value-Datenbank auf Basis von Key Hashing und Sharding

Die Datenbank in Abbildung 1 beruht auf drei Clustern. In dem Master-Cluster übernimmt eine Node das Hashing und das Verteilen der Schlüssel auf die anderen Nodes, während die anderen Nodes die key-value-Paare In-Memory, sprich im RAM, speichern. Nebei spiegeln sie die key-value-Paare auf das erste Slave-Cluster, zur persistenten Speicherung. Das zweite Slave-Cluster ist ein Kopie des ersten, das für komplexe Abfragen genutzt werden kann ohne die Performance der Datenbank negativ zu beeinflussen.

4.4 Dokument-Datenbanken

Eine weitere NoSQL Datenbank sind die sogenannten Dokument-Datenbanken (DDBs). Sie verbinden die kvDBs mit der Möglichkeit eine Strukturierung der Daten vorzunehmen. Dafür basieren sie auf zwei Stufen. Die erste Stufe ist praktisch eine kvDB, die allerdings keine Datenobjekte, sondern Dokumente speichert. Diese Dokumente sind Datensätze mit strukturieden Daten, zum Beispiel im JavaScript Object Notation (JSON)-Format.

4.4.1 Eigenschaften

Als Dokument-Datenbank (DDB) wird eine Datenbank bezeichnet die folgende Eigenschaften erfüllt:

- Sie ist eine kvDB.
- Die Werte der Schlüssel sind Datenobjekte und werden Dokumente genannt.
- In den Dokumenten stehen rekursiv-verschachtelte key-value-Paare
- Die Datenstrukturen sind schemafrei.

4.4.2 MongoDB Intregration in Laravel

Um MongoDB mit Laravel nutzen zu können, ist nur ein geringer Aufwand nötig. Im folgenden wird das hinzufügen einer MongoDB Datenbank, die in der Cloud gehostet wird, erklärt. Als erstes muss das PHP MongoDB Modul installiert werden. Unter Arch beispielsweise geht das wie folgt: `sudo pacman -S php-mongodb`. Danach muss sichergestellt werden, dass das neue Modul geladen wird. Dazu muss folgende Zeile in der Datei php.ini auskommentiert sein: `extension="mongodb.so"`. Der Speicherort der Datei php.ini variiert je nach Betriebssystem. Bei Arch und vielen anderen Linux Distributionen liegt sie in `/etc/php/php.ini` . Schlussendlich muss noch das entsprechende MongoDB Paket zum Laravelprojekt hinzugefügt werden. Das geschieht mit dem PHP Dependency Manager composer wie folgt: `composer require jenssegers/mongodb` . Damit Laravel eine Verbindung zur MongoDB Datenbank aufbauen kann, muss noch eine Anweisung in `/config/database.php` ergänzt werden. Folgender Code muss dem connections Objekt ergänzt werden:

```
'mongodb' => [
        'driver' => 'mongodb',
        'dsn' => env('MDB_DSN'),
        'database' => env('MDB_DATABASE', 'homestead'),
],
```

In der .env Datei wird folgendes ergänzt:

```
MDB_CONNECTION=mongodb
MDB_DATABASE=<Datenbankname>
MDB_DSN=mongodb+srv://<username>:<password>@<Clustername>.mongodb.net/
```

Nun lässt sich noch die Standarddatenbank von Laravel konfigurieren. Dazu wird der bestehende Eintrag am Anfang der Datei `/config/database.php` bei Bedarf zu `:'default' => env('DB_CONNECTION', 'mongodb'),` geändert. Standartmäßig nutzt Laravel eine MySQL Datenbank. Sollte es zu Problemen mit dem automatischen Laden der Abhängigkeiten in Laravel kommen so kann folgendes zum `'providers'` Objekt in `app.php` hinzugefügt werde:
`Jenssegers\Mongodb\MongodbServiceProvider::class, .`
Möchte man den Speicherort für ein Model von dem Standardspeicherort abweichen lassen so fügt man folgendes zur Modelklasse hinzu:
`protected $connection$ = 'mongodb'; .`

[8]

4.5 CAP Theorem

Das CAP Theorem besagt, dass das ein verteiltes System nur zwei der drei Charakteristika der CAP Theorie pro Zeit erfüllen kann. Die drei Charakteristika sind dabei: consistency, availlability und partition tolerance. Ein verteiltes System ist ein System, das Daten auf mehr als einer Node, ein Server in einem Cluster, speichert. NoSQL-Datenbanken sind fast immer verteilte Systeme, da man nur so ihr volles Potenzial ausschöpfen kann (vergleiche Abbildung 1). Im Bezug auf ein nRDBMS sind die drei Charakteristika der CAP Theorie die folgenden:

Consistency: Alle Clients der Datenbank greifen auf die selben Daten zu, egal mit welcher Node der Datenbank sie verbunden sind.Um dies zu gewährleisten müssen die Daten, die auf eine Node geschrieben werden, zeitgleich auf alle anderen Nodes geschrieben werden.
Availability: Jede Node des Datenbankclusters ist in der Lage eine Anfrage zu bearbeiten. Die Datenbank ist auch voll funktionsfähig, wenn eine oder mehrere Nodes nicht erreichbar sind.
Partition tolerance: Die Datenbank ist auch dann voll funktionstüchtig, wenn Pakete zwischen den Nodes gar nicht oder nur verspätet ankommen.

5 Vergleich von SQL- und NoSQL-Datenbanken

5.0.1 Wann sind RDBMS geeignet?

RDBMS sind vor allem dann gut geeignet, wenn die Daten einen hohen Grad an Verknüpfungen aufweisen oder die ACID-Kriterien erfüllt werden müssen. Für einige Anwendungen, wie zum Beispiel Bankanwendungen, ist die Erfüllung der ACID-Kriterien zwingend erforderlich. Zudem lassen sich Mittels SQL komplexe ad-hoc Abfragen einfach gestalten. Die feste Struktur ist zudem gut geeignet, wenn keine Änderungen in der Menge oder der Art der Daten, die gespeichert werden sollen, zu erwarten ist. Des weiteren bieten RDBMS mit SQL eine standartisierde query language, was es Entwicklern ermöglicht, das RDBMS zu wechseln ohne eine neue query language lernen zu müssen oder große Änderung in ihren Anwendungen vornehmen zu müssen. [3]

5.0.2 Wann sind nRDBMS geeignet?

Ein nRDBMS ist besonders dann gut geeignet, wenn sehr große Datenmengen mit wenig Verknüpfungen und vielen Änderungen gespeichert werden sollen. Zudem bietet sie durch ihre Fähigkeit des horizontalen Skalierens große Vorteile beim Cloud Computing. Wird zum Beispiel ein weiterer Cachingserver benötigt um einer erhöhten Nachfrage nachzukommen kann dieser einfach beim Cloud Service Provider gebucht werden um das Cluster zu erweitern. Nimmt der Bedarf wieder ab kann ein Server wieder abbestellt werden um Kosten zu sparen. Ein weitere Vorteil besteht wenn nur sehr kurze Entwicklungszeiten gegeben sind. Wird zum Beispiel eine Anwendung innerhalb von wenigen Wochen entwickelt, kann es vorkommen, dass keine Zeit frei ist um eine sorgfältig Vorausplanung der Datenbank vorzunehmen. Stattdessen entsteht und wächst sie mit der Entwicklung der Anwendung einfach mit. [3]

6 Literaturverzeichnis

[1] Wajid Ali, Muhammad Usman Shafique, Muhammad Arslan Majeed, and Ali Raza. Comparison between sql and nosql databases and their relationship with big data analytics. *Asian Journal of Research in Computer Science*, pages 1–10, 2019.

[2] Michael Kaufmann Andreas Meier. *SQL- und NoSQL-Datenbanken*.

[3] Brad Nicholson Benjamin Anderson. Sql vs. nosql databases: Whats the difference? URL: `https://www.ibm.com/cloud/blog/sql-vs-nosql`.

[4] Edgar F Codd. A relational model of data for large shared data banks. In *Software pioneers*, pages 263–294. Springer, 2002.

[5] Edgar F Codd et al. *Relational completeness of data base sublanguages*. Citeseer, 1972.

[6] Hugh Darwen. *An introduction to relational database theory*. Bookboon, 2009.

[7] John Rydning David Peinsel, John Ganrt. Data age 2025: The evolution of data to life-critical.

[8] MongoDB Dokumentation. Mongodb and laravel integration. URL: `https://www.mongodb.com/compatibility/mongodb-laravel-intergration`.

[9] Brad Fowler, Joy Godin, and Margaret Geddy. Teaching case: introduction to nosql in a traditional database course. *Journal of Information Systems Education*, 27(2):99, 2016.

[10] Santhosh Kumar Gajendran. A survey on nosql databases. *University of Illinois*, 2012.

[11] Narain Gehani. Introduction to database systems. 1990.

[12] Dayne Hammes, Hiram Medero, and Harrison Mitchell. Comparison of nosql and sql databases in the cloud. *Proceedings of the Southern Association for Information Systems (SAIS), Macon, GA*, pages 21–22, 2014.

[13] Changlin He. Survey on nosql database technology. *Journal of Applied Science and Engineering Innovation Vol*, 2(2):50–54, 2015.

[14] Thomas Jüngling. Datenvolumen verdoppelt sich alle zwei jahre. 07 2013. URL: `https://www.welt.de/wirtschaft/webwelt/article118099520/Datenvolumen-verdoppelt-sich-alle-zwei-Jahre.html`.

[15] Dipl.-Ing. (FH) Stefan Luber. Was ist acid? URL: `https://www.bigdata-insider.de/was-ist-acid-a-776182/`.

[16] Ian Moor. An introduction to sql and postgresql, 2001.

[17] Ameya Nayak, Anil Poriya, and Dikshay Poojary. Type of nosql databases and its comparison with relational databases. *International Journal of Applied Information Systems*, 5(4):16–19, 2013.

[18] Putu Adi Guna Permana and Evi Triandini. Performance with eloquent and query builder in crowdfunding system with laravel framework. 2021.

[19] Vatika Sharma and Meenu Dave. Sql and nosql databases. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(8), 2012.

[20] Carlo Strozzi. Nosql: a non-sql rdbms. URL: `http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page`.

[21] Antonia Thiele. Deutsche speichern mehr als 1000 fotos auf dem handy. URL: `https://www.welt.de/wirtschaft/article202489422/Handybilder-Deutsche-speichern-mehr-als-1000-Fotos-auf-dem-Smartphone.html`.

[22] W3.CSS. What is json? URL: `https://www.w3schools.com/whatis/whatis_json.asp`.

A Anhang

A.1 MongoDB Entwicklungsdatenbank in der Cloud hosten

Zu Entwicklung- und Testzwecken lässt sich eine MongoDB Datenbank einfach und kostenfrei in der Cloud hosten. Dazu zu `https://www.mongodb.com/cloud/atlas` gehen und ein Konto erstellen. Nach dem ersten Anmelden müssen ein paar Fragen beantwortet werden, wie man plant MongoDB zu nutzen. Danach wählt man aus den Deployment Optionen Shared aus. Als Cloudprovider wird der Provider der Wahl ausgewählt und ein Standort gewählt, der nahe der eigenen geographischen Lage ist. Die restlichen Einstellungen können auf den Standardeinstellungen belassen werden. Das sind, Stand 17.11.2021, Cluster Tier: M0 Sandbox(Shared RAM, 512MB Storage), Additional Settings: MongoDB 4.4, No Backup und Cluster Name: Cluster0. Hat man das Cluster erstellt, dauert es ein paar Minuten bis es zur Verfügung steht. Steht das Cluster zur Vefügung geht man auf den Punkt Connect und fügt durch einen Klick auf Add Your Current IP Address seine IP Adresse zu der Whitelist der IP Adressen hinzu, die sich mit dem Cluster verbinden dürfen. Danach wird ein Datenbanknutzer erstellt. Dann wählt man unter den Verbindungsoptionen MondoDB Connect aus und kopiert den connection string in das Feld `DB_DSN` in der .env Datei. das Feld `DB_Database` wird mit dem gewünschten Datenbanknamen ausgefüllt.

A.2 Postman API Client

Postman bietet eine Möglichkeit API's leicht mit einer grafischen Oberfläche zu testen. Postman ist erhältlich über den Pakagemanger der Wahl oder unter `https://app.getpostman.com/app/download/win64`. In der Oberfläche von Postman lässt sich die HTML Methoden und eine URL angeben. Unter dem Punkt Header muss der Eintrag Accept - application/json ergänzt werden. Im Body lassen sich bei Bedarf zum Beispiel JSON-Daten mitschicken.

A.3 JSON

JSON ist die Abkürzung für **J**ava**S**cript**O**bject**N**otation. Die Syntax ist identisch zu dem Code von Objekten in JavaScript. Es gelten folgende Regel:

- Daten werden in key-value-Paaren gespeichert
- Daten werden mit Kommata getrennt.
- Geschwungene Klammern werden verwendet um Objekte zu speichern.
- Eckige Klammern werden verwendet um Array zu speichern.

[22]

B Eigenständigkeitserklärung

"Ich versichere, dass ich die Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen oder anderen Quellen entnommen sind, sind als solche kenntlich gemacht."

Ort, Datum

Unterschrift