Bericht

### Welche NoSQL Datenbank hast Du gewählt und wieso diese?

# Meine Problemstellung

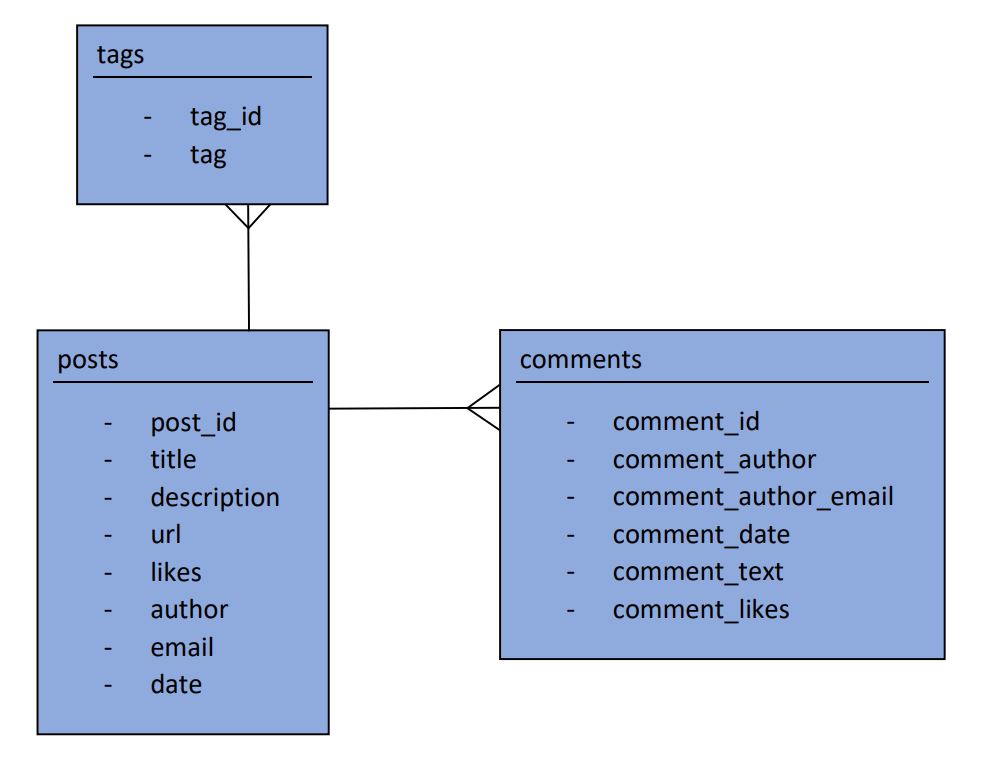
Ich habe mich entschieden MongoDB mit SQL zu vergleichen. Ich habe mich für MongoDB entschieden, weil MongoDB die bekannteste NoSQL-Datenbank ist. Da MongoDB ein sehr breites Anwendungsgebiet hat, machte es Sinn MongoDB etwas genauer anzuschauen. Zudem ist auch die flexible Struktur der Dokumente ein grosser Vorteil gegenüber SQL. Besonders bei Projekten, bei denen sich die Datenbank mit hoher Wahrscheinlichkeit noch verändern wird, kann dies ein enormer Vorteil sein.

Ich will SQL mit MongoDB vergleichen. Dabei will ich eine Datenbank erstellen, welche die Metadaten von Blogartikeln speichert, inklusiv Kommentaren und Tags. Dabei sehe ich die Vorteile von MongoDB vor allem bei der Abfragezeit, weil zum Darstellen auf eines Blog-Posts genau ein gesamtes Dokument abgerufen werden muss. Dies ist bei einer Blog-Webseite wohl die häufigste Abfrage. Zudem sind Abfragen so auch sehr simpel. MongoDB ist allgemein für die Schnelligkeit bekannt und verspricht somit besonders gute Ergebnisse. Weitere Vorteile sind die Flexibilität und Einfachheit, die Datenstruktur anzupassen und ganz allgemein die Implementationszeit.

# Meine Vorkenntnisse

Ich habe keine berufliche Erfahrung mit Datenbanken. Jedoch habe ich in dem Data Science Studium schon ein paar SQL-Datenbanken genutzt. Ich habe ebenfalls in der Form des Modul GDB schon mit Datenbanken Erfahrungen sammeln können. Jedoch habe ich vor dieser Abgabe noch keine eigenen Datenbanken aufgesetzt und eigene Daten eingelesen.

# Konzeptionelles Datenmodell

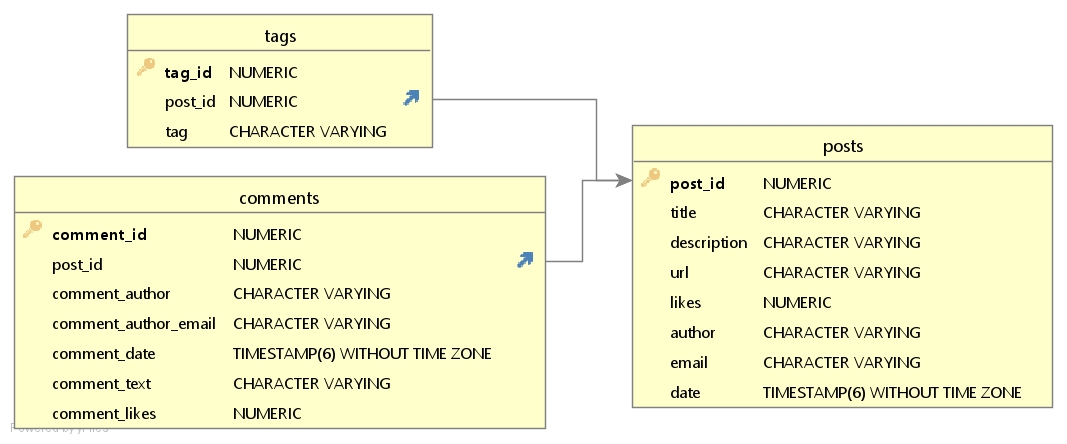
Ich habe versucht ein ERD mithilfe von Visio zu erstellen. Leider fehlte mir dafür die nötige Lizenz. Deshalb habe ich dann das ERD mithilfe von Microsoft Word erstellt. Dies ist suboptimal, doch funktionierte hierfür einwandfrei. Hierbei geht es grundsätzlich darum zu sehen, welche Daten in unserer Datenbank gespeichert werden und in welchem Zusammenhang sie mit den restlichen Daten stehen.

Die Datenbank lässt sich demnach mit drei simplen Tabellen implementieren. Die erste Tabelle speichert die Metadaten der Posts. Diese beinhalten übliche Metadaten für Blogposts wie zum Beispiel URL, Titel, Anzahl Likes und Datum der Veröffentlichung. Weiterhin haben wir die Tabelle mit den Tags. Darin wird nur der (Hash)tag gespeichert. Dieser muss einfach korrekt mit einem Post verbunden werden, aber mehr dazu später. Die Datenbank hat ebenfalls Kommentare. Dies sind die Kommentare, welche unterhalb eines Posts geschrieben werden können. Somit kann es 0 bis viele Kommentare pro Post geben. Auch hier speichern wir die nötigsten Informationen wie Autor, Datum der Veröffentlichung und natürlich den Text des Kommentars.

# Meine Umsetzung

**Wie sieht die Umsetzung in den beiden Datenbanken aus?**

Ich habe die beiden Datenbanken mithilfe von drei Python-Scripts aufgesetzt. Das erste Script `create\_database.py` erstellt die Datenbanken. Bei SQL habe ich mit für eine PostgreSQL Datenbank entschieden. Im ersten Schritt muss hier eine Verbindung mit pgAdmin aufgebaut werden. Dies tun wir mithilfe des Packages `psycopg2`. Danach wird eine Datenbank aufgesetzt und die nötigen Tabellen erstellt. Dabei müssen die Struktur der Tabellen und die Verbindungen zwischen den Tabellen mitgegeben werden. Dies ist zeitaufwendig und braucht etwas Planung, um einwandfrei zu funktionieren. Um die Datenbank zu visualisieren, habe ich bei der PostgreSQL Datenbank mithilfe der Applikation DbVisualizer vollautomatisch ein UML-Diagramm erstellt. Darauf sind die drei Tabellen mit ihren Datentypen zu sehen. Es ist ebenfalls ersichtlich, wie die Foreign-Keys mit den Primary-Keys verbunden sind.



Für MongoDB muss ebenfalls eine Verbindung mit dem Cluster aufgebaut werden und danach muss eine Collection erstellt werden. Dafür nutzen wir das Package `pymongo`. Die Verbindung wird mit nur 3 Zeilen Code aufgebaut. Danach braucht es nur eine weitere Zeile Code um alle Blogposts hinzuzufügen. Dabei kann die Struktur von verschiedenen Dokumenten unterschiedlich sein und muss auch nicht angegeben werden. Um unsere MongoDB Datenbank zu visualisieren, haben wir ein Beispielsdokument, welches alle möglichen Variablen enthält, herauskopiert. Dieses zeigt auf, wie ein solches Dokument aufgebaut ist.

{

"\_id": { "$oid": "61d4a8b47e51883717060e1a" },

"post\_id": { "$numberInt": "4" },

"title": "Here buy TV at anyone.",

"description": "East despite work perform first.",

"url": "http://www.clark.com/",

"likes": { "$numberInt": "40495" },

"author": "williamsjennifer",

"email": "rodgersheather@yahoo.com",

"date": { "$date": { "$numberLong": "1614290590000" } },

"tags": ["#Here", "#buy", "#TV", "#at", "#anyone"],

"comments": [

{

"comment\_id": { "$numberInt": "37" },

"comment\_author": "timothymayer",

"comment\_author\_email": "isaiah02@hotmail.com",

"comment\_date": { "$date": { "$numberLong": "1635125256000" } },

"comment\_text": "Into.",

"comment\_likes": { "$numberInt": "11" }

},

{

"comment\_id": { "$numberInt": "38" },

"comment\_author": "fowlertodd",

"comment\_author\_email": "gholt@hotmail.com",

"comment\_date": { "$date": { "$numberLong": "1605428573000" } },

"comment\_text": "Be.",

"comment\_likes": { "$numberInt": "772" }

}

]

}

Die Daten wirken teilweise, wie beispielswiese bei der URL etwas sinnfrei. Dies liegt daran, dass die Daten mithilfe des Packages `faker` erstellt wurden. Wichtig war, dass die Daten synthetischen Daten das richtige Format hatten, der genaue Inhalt war mir dabei egal. Für das Erstellen der synthetischen Daten nutzte ich das zweite Python File `create\_data.py`. Dieses erstellt dieselben Daten für MongoDB und PostgreSQL im korrekten Format zur Befüllung der Datenbanken.

# Vorteile von MongoDB

**Welche Vorteile hast Du bei der NoSQL Datenbank erwartet?**

Ein wichtiger Vorteil gegenüber SQL ist die Dynamische Entwicklung und die hohe Skalierbarkeit der Datenbank von einzelnen Servern bis hin zu komplexen Architekturen über mehrere Rechenzentren. Da die Dokumente sehr unterschiedlich sein können, ist man in der Weiterentwicklung der Datenbank extrem flexibel. Bei einer Website mit Blogposts ist es möglich, dass man später die Website erweitern will und somit neuere Blogposts andere Metadaten als ältere Blogposts haben. Dies ist für MongoDB kein Problem.

Bei SQL ist das schon etwas schwieriger. Auch ist die Struktur, da in JSON (respektive BSON) Frontend tauglich und erleichtert somit die Implementation. Generell habe ich erwartet, dass der Entwicklungsaufwand kleiner ist als bei SQL. Dies hat sich auch schon beim Aufsetzen der Datenbank bestätigt.

Ich sehe bei diesem Projekt die Vorteile von MongoDB vor allem bei der Abfragezeit, weil zum Darstellen auf eines Blog-Posts genau ein gesamtes Dokument abgerufen werden muss. Zudem sind Abfragen so auch sehr simpel. Bei den Abfragen sind keine Joins nötig was die Abfragen theoretisch weniger Komplex macht.

# Messkriterien

**Welche Messkriterien hast Du festgelegt, wie sind sie, interpretiere sie?**

Ich habe mich entschieden die beiden Datenbanken anhand der folgenden Messkriterien zu bewerten.

* Die Abfragezeit: Wie lange brauchen die beiden Datenbanken, um mir die in einem allfälligen Frontend darzustellenden Daten zur Verfügung zu stellen? Hier will ich mehrere die beiden Datenbanken für mehrere Abfragen vergleichen.
* Die Anzahl Code-Zeilen: Wie viele Zeilen Code brauchte ich für die beiden Implementationen?
* Die Implementationszeit: Sie soll vergleichen, wie lange ich brauchte, um die beiden Datenbanken aufzusetzen.
* Der Speicherbedarf: Welche Datenbank braucht mehr Speicher?

## Abfragezeit

## Anzahl Code-Zeilen

Hier war der Unterschied wirklich enorm. Zum Erstellen der Datenbank brauchte ich mit SQL 88 Zeilen Code und mit MongoDB nur 10 Zeilen. Bei SQL kamen dann noch zusätzlich 57 Zeilen für Materialized-Views und Indexes dazu. SQL brauchte vor allem so viel Zeilen, weil ich die Struktur der Datenbank angeben musste, während ich bei MongoDB nur eine leere Collection erstellen musste. Gesamthaft brauchte SQL also mit 145 Zeilen Code gegenüber MongoDB mit 10 Zeilen Code fast 15-mal mehr Code.

Auch die Abfragen von MongoDB waren hier deutlich kürzer und simpler. Tatsächlich waren unsere MongoDB Abfragen nur eine Zeile lang, während die SQL-Abfragen mit den Joins schon mal ein Paar Zeilen Code brauchen konnten.

Deutlich mehr Zeilen SQL-Code zu schreiben braucht natürlich auch mehr Zeit, weshalb wir jetzt die Implementationszeit anschauen.

## Implementationszeit

## Speicherbedarf

# Fazit

### Gibt es andere wichtige Punkte, die auf Deinen Lernerfolg einen wichtigen Einfluss hatten?

### Was war für Dich die wichtigste Erkenntnis? (Des gesamten Projekts)