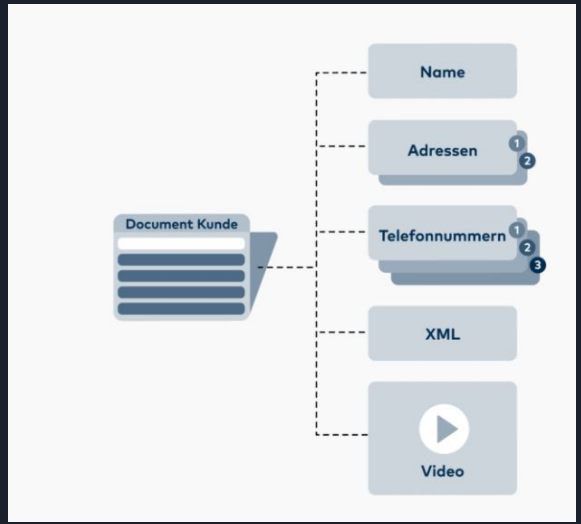
MongoDB/KMongo

von Alexander Czegley und Gabriel Lukas Schön
(764 796) (765 118)

Einstieg: MongoDB anhand eines Beispiels

id	Anrede	Name	Vorname	Strasse	PLZ	TelPrivat	TelBuero	Fax	eMail
1	Herr	asfxy	asdfa	asdf	345	3535			a@b.de
2	Frau	xy			0				

- Unternehmen mit Kundenverwaltung
- Erste Idee: relationale Datenbank einsetzen
- Was ist, wenn wir nun möglicherweise eine, zwei, ... Telefonnummer/Adressen für einen Kunden hinzufügen möchten?
- Mögliche Lösung: mehrere Tabellen, einzelne Attribute auslagern und über FK auf jeweiligen Kunden verweisen



- Wie löst MongoDB dieses Problem?
- Daten werden in sogenannten „Documents“ gespeichert, Dokumente sind nicht darauf beschränkt, die gleiche Anzahl von Spaltendokumenten/Datenfeldern zu haben

Gliederung

- Was ist NoSQL ?
- Arten von NoSQL Datenbanken
- Ziele von NoSQL
- Vergleich zu SQL
- Was ist MongoDB ?
- MongoDB
  - Was ist ein Dokument ?
  - Datentypen
  - Eingebettete Dokumente
  - Collections
  - Schema Design
  - Data Model Design
  - Kardinalitäten/Relationships
  - One-to-One Relationships
    - Embedded Document Pattern
    - Subset Pattern
  - One-to-Many Relationships
    - Embedded Document Pattern
    - Subset Pattern
  - Many-to-Many Relationships
    - two-way-Embedding
    - one-way-Embedding
- Schema-Design
  - Vorgehen
- Installation von MongoDB
- Überprüfen der Installation/erste Schritte
- MongoDB Crud Operationen
  - Create
  - Read
  - Update
  - Delete
- Was ist KMongo ?
- Installation von KMongo
- Probleme beim Installieren
- Beispielcode
- Sicherheit
- Live-Beispiel mit Kotlin
- Quellen

Was ist NoSQL ?

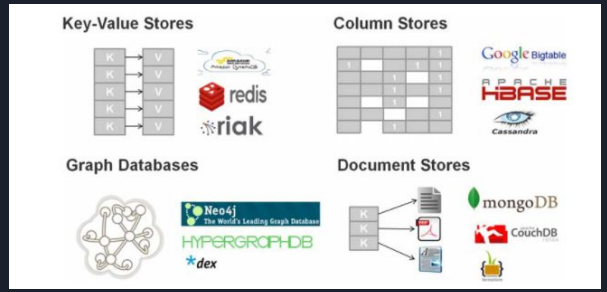
https://learning.oreilly.com/library/view/mongodb/9781492015499/ (Trelle, Tobias MongoDB Der praktische Einstieg dpunkt.Verlag, Kapitel 1.2 NoSQL)

- NoSQL: leitet sich ab von not only SQL, wobei SQL als Synonym für relationale Datenbanksysteme verwendet wird
- Idee: Beim Umsetzen von konkreten Anforderungen nicht im Vorhinein auf relationale Datenbanksysteme (etabliert) beschränken
- Ziel: Datenbanksystem einsetzen, das jew. Problem optimal löst
- Entstehung: im Web-2: Andere Anforderungen an die Datenhaltung, viele Daten sehr schnell speichern und abrufen
- NoSQL Datenbanken sind für Anwendungen optimiert, bei denen die SQL-basierten relationalen Datenbanken an ihre Grenzen stoßen

Arten von NoSQL Datenbanken

https://www.ovhcloud.com/sites/default/files/styles/text_media_horizontal/public/2021-05/MongoDB_Logo_FullColorBlack_RGB-4td3yuxzjs.png

- Dokumentenorientierte Datenbanken (speichert Daten in Dokumenten)
- Key-Value-Datenbanken (Nutzt einfache Schlüssel und Werteschemata)
- Graphendatenbanken (zu speichernden Informationen mithilfe der Eigenschaften von Knoten und Kanten abgebildet)
- spaltenorientierte Datenbanken (Einträge in Spalten statt in Zeilen gespeichert. Vorteil dieses Modells ist, dass im Gegensatz zu zeilenorientierten DB's keine unnötigen Daten gelesen werden müssen, Leseprozesse schneller)



https://mikedietrichde.com/wp-content/uploads/2017/08/xdb.png

https://learning.oreilly.com/library/view/mongodb/9781492015499/ (Trelle, Tobias MongoDB Der praktische Einstieg dpunkt.Verlag, Kapitel 1.2 NoSQL)

https://www.suse.com/c/wp-content/uploads/2018/10/SAP_HANA-768x256.jpg

Ziele von NoSQL

- Horizontale Skalierbarkeit
- Vermeiden unnötiger Komplexität
- Hohe Performance und hoher Durchsatz
- Vermeidung von relationalen Ansätzen des Datenmappings
- Einfache Installation und Konfiguration von verteilten Datenbank-Clustern

Vergleich zu SQL

	SQL	NoSQL
Entwicklung	In den 1970er Jahren für die erste Welle der Anwendungen für die Datenspeicherung entwickelt	Ende der 2000er Jahre entwickelt Im Hinblick auf : Skalierbarkeit, unterschiedlichen Strukturen von Daten, geografischen Verteilung von Daten, und der agilen Entwicklung
Schema	Relational: Datensätze als Zeilen in Tabellen, jede Spalte enthält Attribut/Eigenschaft des Datensatzes Verwandte Daten in jeweils einer Tabelle gespeichert (z.B. alle Mitarbeiter) für Anfragen können mehrere Tabellen verbunden werden (Konzept Fremdschlüssel)	z.B. bei Dokumentdatenbanken: alle wichtigen Daten werden in einem einzigen „Dokument“ in JSON, XML oder einem anderen Format gespeichert, Werte hierarchisch in Dokument verschachtelt

Vergleich zu SQL

	SQL	NoSQL
Struktur	Struktur und Datentypen müssen vorab definiert werden, bei Änderungen muss die gesamte DB geändert/aktualisiert werden	meist dynamisch, Felder können in Echtzeit neu hinzugefügt werden, ungleiche Daten können zusammen erfasst werden
Skalierung	Vertikal: meist ein Server, dessen Leistung erhöht werden muss Verteilte DB's möglich, aber nur sehr aufwändig realisierbar (Verlust Transaktionsintegrität, JOINs für Anfragen)	Horizontal: Kapazität erhöhen, indem ein zusätzlicher Server zur DB hinzugefügt wird Daten können neu verteilt werden
Änderung von Daten	SQL-Operationen	OO-API

Was ist MongoDB ?

- Mongo: Ableitung aus englischem Wort “humongous”:, gigantisch/ wahnsinnig groß
- Universelle, dokumentbasierte, verteilte Datenbank
- Daten in einem JSON-ähnlichen Dokumentformat gespeichert (BSON, = Binary JSON)
- Ziel: Natürlichere Erfassung von Daten
- Verwaltung von semistrukturierten Daten (Daten, die keiner festen Struktur folgen, sondern die Struktur quasi in sich selbst tragen)
- Speicherung von Daten: Schlüssel+spezifisches Dokument, welches eigentliche Informationen enthält

MongoDB: Was ist ein Dokument ?

- Einzelner Datensatz in MongoDB
- Schreibende Zugriffe auf Ebene eines Dokuments sind atomar
- Dokument besteht aus einer geordneten Menge von Feldern
- Ein Feld ist ein Key-Value-Paar,
  - Value ist ein einfacher Datentyp, ein Array oder wiederum ein Dokument
  - Key ist "_id"-Schlüssel. Der Wert des "_id"-Schlüssels kann ein beliebiger Typ sein, standardmäßig ist er jedoch eine ObjectId.

```
{ "_id" : ObjectId("6171b63af8895dc3eb78c756")
```

MongoDB: Datentypen

{"x" : null}	Null: Nullwert oder nicht vorhandenes Feld
{"x" : true}	boolean
{"x" : 3.14} {"x" : 3}	Number: standardmäßig 64-Bit-Gleitkommazahlen
{"x" : NumberInt("3")} {"x" : NumberLong("3")}	NumberInt, NumberLong für Ganzzahlen
{"x" : "foobar"}	String
{"x" : new Date()}	Date
{"x" : ["a", "b", "c"]}	Array

MongoDB: Eingebettete Dokumente

- Ein Dokument kann als Wert für einen Schlüssel verwendet werden
- Daten könne auf eine natürlichere Weise organisiert werden, als nur durch eine flache Struktur von Schlüssel/Wert-Paaren
- Über MongoDB ist, Zugriff in eingebettete Dokumente möglich
  - Indizes zu erstellen,
  - Abfragen durchführen
  - Aktualisierungen vorzunehmen

https://learning.oreilly.com/library/view/mongodb-the-definitive/9781491954454/

MongoDB: Collections

- Collection: Gruppe von Dokumenten
- Vergleich:
  - Dokument in MongoDB kann analog zu einer Zeile in einer relationalen Datenbank gesehen werden
  - Collection analog zu einer Tabelle
- Dynamische Schemata: Dokumente innerhalb einer einzigen Sammlung können eine beliebige Anzahl verschiedener "Formen" haben
- Beispiel:
  - {"greeting" : "Hello, world!", "views": 3}
  - {"signoff": "bye"}
- Warum braucht es verschiedene Collections?
  - Aufbewahrung verschiedener Arten von Dokumenten in einer Collection oft umständlich
  - Sicherstellen, dass Anwendungscode, der eine Abfrage ausführt, mit Dokumenten unterschiedlicher Form umgehen kann

MongoDB: Beispiel: Dokument und Collection

```
> db.items.insert({
...         "year" : 2013,
...         "title" : "Turn It Down, Or Else!",
...         "info" : {
...             "directors" : [ "Alice Smith", "Bob Jones"],
...             "release_date" : "2013-01-18T00:00:00Z",
...             "rating" : 6.2,
...             "genres" : ["Comedy", "Drama"],
...             "image_url" : "http://ia.media-imdb.com/images/N/O9ERWAU7FS797AJ7LU8HN09AMUP908RL1o5JF90EWR7LJKQ7@@._V1_SX400_.jpg",
...             "plot" : "A rock band plays their music at high volumes, annoying the neighbors.",
...             "actors" : ["David Matthewman", "Jonathan G. Neff"]
...         }
...     })
```

```
{ "_id" : ObjectId("6171b63af8895dc3eb78c756"), "year" : 2013, "title" : "Turn It Down, Or Else!", "info"
 : { "directors" : [ "Alice Smith", "Bob Jones" ], "release_date" : "2013-01-18T00:00:00Z", "rating" : 6.
2, "genres" : [ "Comedy", "Drama" ], "image_url" : "http://ia.media-imdb.com/images/N/O9ERWAU7FS797AJ7LU8
HN09AMUP908RL1o5JF90EWR7LJKQ7@@._V1_SX400_.jpg", "plot" : "A rock band plays their music at high volumes,
 annoying the neighbors.", "actors" : [ "David Matthewman", "Jonathan G. Neff" ] } }
```

MongoDB: Schema Design

- MongoDB: schemafreie Datenbank (technisch gesehen)
- Trotzdem nimmt Schema-Design hohen Stellenwert ein
- Um möglichst performante Anwendungen zu entwerfen und den fehlenden dokumentenübergreifenden Transaktionen zu begegnen, sind Überlegungen zur Strukturierung der Datenablage unerlässlich
    - Grundsätzlich kann eine Collection Dokumente mit beliebiger Struktur speichern.
    - Jedes dieser Dokumente kann sich dabei völlig von den anderen Dokumenten unterscheiden, sowohl in der Menge der Felder als auch deren Typ
    - Fachlich macht es wenig Sinn, alle Ihre Geschäftsobjekte in einer Collection zu speichern

MongoDB: Data Model Design

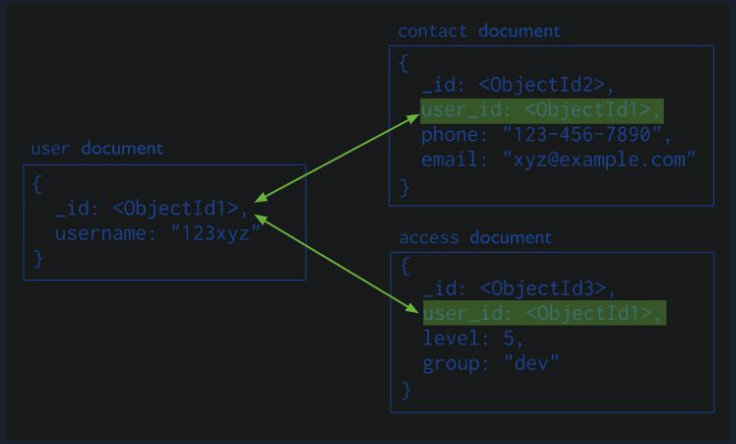
Embedded Data Models

https://docs.mongodb.com/manual/core/data-model-design/

- Eingebettete Datenmodelle ermöglichen es Anwendungen, zusammenhängende Informationen im selben Datenbankeintrag zu speichern
- Somit müssen weniger Abfragen und Aktualisierungen durchgeführt werden
- Eingebettete Datenmodelle verwenden bei:
  - "contains"-Beziehungen zwischen Entitäten (one-to-one)
  - One-to-Many-Beziehungen zwischen Entitäten

MongoDB: Data Model Design

Normalized Data Models



https://docs.mongodb.com/manual/core/data-model-design/

- Normalisierte Datenmodelle verwenden bei:
  - Wenn die Einbettung zu einer Duplizierung von Daten führen würde, aber keine ausreichenden Vorteile bei der Leseleistung bietet, um die Auswirkungen der Duplizierung aufzuwiegen
  - Um komplexere Many-to-many-Beziehungen darzustellen
  - Zur Modellierung großer hierarchischer Datensätze

MongoDB: Data Model Design

Embedding is better for...	References are better for...
Small subdocuments	Large subdocuments
Data that does not change regularly	Volatile data
When eventual consistency is acceptable	When immediate consistency is necessary
Documents that grow by a small amount	Documents that grow by a large amount
Data that you'll often need to perform a second query to fetch	Data that you'll often exclude from the results
Fast reads	Fast writes

https://learning.oreilly.com/library/view/mongodb-the-definitive/9781491954454/

MongoDB: Kardinalitäten/Relationships

- Übliche Kardinalitäten:
  - one-to-one
  - one-to-many
  - many-to-many
- Bei Verwendung von MongoDB kann es sinnvoll sein, many in “few” ,“many” und “squillions” zu unterteilen
- few-Beziehungen einbetten
- many-Beziehungen referenzieren
- Einbettung verbundener Daten in ein einziges Dokument kann die Anzahl der Lesevorgänge reduzieren, die zum Abrufen von Daten erforderlich sind
- Schema möglichst so strukturieren, dass die Anwendung alle erforderlichen Informationen in einem einzigen Lesevorgang erhält

MongoDB: One-to-One Relationships: Embedded Document Pattern

normalized data model

```
// patron document
{
   _id: "joe",
   name: "Joe Bookreader"
}

// address document
{
   patron_id: "joe", // reference to patron document
   street: "123 Fake Street",
   city: "Faketon",
   state: "MA",
   zip: "12345"
}
```

https://docs.mongodb.com/manual/tutorial/model-embedded-one-to-one-relationships-between-documents/

embedded data model

```
{
   _id: "joe",
   name: "Joe Bookreader",
   address: {
              street: "123 Fake Street",
              city: "Faketon",
              state: "MA",
              zip: "12345"
            }
}
```

https://docs.mongodb.com/manual/tutorial/model-embedded-one-to-one-relationships-between-documents/

- Normalisierten Datenmodell: Adressdokument enthält einen Verweis auf das Kundendokument
- Wenn die Adressdaten häufig zusammen mit den Namensinformationen abgerufen werden, muss die Anwendung bei der Referenzierung mehrere Abfragen durchführen, um die Referenz aufzulösen
- Bessere Einbettung: Anwendung kann mit einer einzigen Abfrage die vollständigen Informationen über den Kunden abrufen.

MongoDB: One-to-One Relationships: Subset Pattern

Embedded Document


```
{
  "_id": 1,
  "title": "The Arrival of a Train",
  "year": 1896,
  "runtime": 1,
  "released": ISODate("01-25-1896"),
  "poster": "http://ia.media-
imdb.com/images/M/MV5BMjEyNDk5MDYzOV5BMl5BanBnXkFtZTgwNjIxMTEwMzE@._
V1_SX300.jpg",
  "plot": "A group of people are standing in a straight line along the platform of a
railway station, waiting for a train, which is seen coming at some distance. When the
train stops at the platform, ...",
  "fullplot": "A group of people are standing in a straight line along the platform of a
railway station, waiting for a train, which is seen coming at some distance. When the
train stops at the platform, the line dissolves. The doors of the railway-cars open, and
people on the platform help passengers to get off.",
  "lastupdated": ISODate("2015-08-15T10:06:53"),
  "type": "movie",
  "directors": [ "Auguste Lumière", "Louis Lumière" ],
  "imdb": {
    "rating": 7.3,
    "votes": 5043,
    "id": 12
  },
  "countries": [ "France" ],
  "genres": [ "Documentary", "Short" ],
  "tomatoes": {
    "viewer": {
      "rating": 3.7,
      "numReviews": 59
    },
    "lastUpdated": ISODate("2020-01-09T00:02:53")
  }
}
```


- Potenzielles Problem Embedded Document Pattern:
  - Große Dokumente mit Feldern, die die Anwendung nicht (häufig) benötigt
  - Lesevorgänge werden verlangsamt
- Subset-Pattern:
  - Nicht häufig genutzte Daten in eigene Dokumente auslagern
  - Teilmenge von Daten auf der “one” Seite vorhalten, auf die am häufigsten in einem einzigen Datenbankaufruf zugegriffen wird
- Vorteil: Verbesserte Leseleistung
- Nachteil:
  - Bei falscher Aufteilung mehrfacher DB-Zugriff durch Anwendung nötig
  - Wartungsaufwand für die Datenbank kann erhöht werden (Löschen, Aktualisieren…)

https://docs.mongodb.com/manual/tutorial/model-embedded-one-to-one-relationships-between-documents/

MongoDB: One-to-One Relationships: Subset Pattern Beispiel

```
{
  "_id": 1,
  "title": "The Arrival of a Train",
  "year": 1896,
  "runtime": 1,
  "released": ISODate("1896-01-25"),
  "type": "movie",
  "directors": [ "Auguste Lumière", "Louis Lumière" ],
  "countries": [ "France" ],
  "genres": [ "Documentary", "Short" ],
}
```

https://docs.mongodb.com/manual/tutorial/model-embedded-one-to-one-relationships-between-documents/

```
// movie_details collection

{
  "_id": 156,
  "movie_id": 1, // reference to the movie collection
  "poster": "http://ia.media-imdb.com/images/M/MV5BMjEyNDk5MDYzOV5BMl5BanBnXkFtZTgwNjIx
  "plot": "A group of people are standing in a straight line along the platform of a ra
  "fullplot": "A group of people are standing in a straight line along the platform of
  "lastupdated": ISODate("2015-08-15T10:06:53"),
```

https://docs.mongodb.com/manual/tutorial/model-embedded-one-to-one-relationships-between-documents/

- movie_details enthält mehrere Felder, die die Anwendung nicht benötigt, um einen einfachen Überblick über einen Film zu geben, z. B. “plot” und “fullplot”
- movie_details, referenziert zugehörigen Film mit dem Feld: “movie_id”
- Bei Bedarf können somit Zusatzdaten aus der movie_details Collection geladen werden

MongoDB: One-to-Many Relationships: Embedded Document Pattern

normalized data model

```
// patron document
{
   _id: "joe",
   name: "Joe Bookreader"
}

// address documents
{
   patron_id: "joe", // reference to patron document
   street: "123 Fake Street",
   city: "Faketon",
   state: "MA",
   zip: "12345"
}

{
   patron_id: "joe",
   street: "1 Some Other Street",
   city: "Boston",
   state: "MA",
   zip: "12345"
}
```

embedded data model

```
{
   "_id": "joe",
   "name": "Joe Bookreader",
   "addresses": [
                 {
                   "street": "123 Fake Street",
                   "city": "Faketon",
                   "state": "MA",
                   "zip": "12345"
                 },
                 {
                   "street": "1 Some Other Street",
                   "city": "Boston",
                   "state": "MA",
                   "zip": "12345"
                 }
               ]
}
```

- Embedded Document Pattern:
  - Dokumente, zu denen die many-Relation besteht auf one-Seite einbinden
  - Sinnvoll, wenn die Daten der many-Seite häufig zusammen mit der one-Seite abgefragt werden
  - Häufig bei one-to-few Relationship

https://docs.mongodb.com/manual/tutorial/model-referenced-one-to-many-relationships-between-documents/

MongoDB: One-to-Many Relationships: Subset Pattern

Produkt-collection

```
{
  "_id": 1,
  "name": "Super Widget",
  "description": "This is the most useful item in your toolbox.",
  "price": { "value": NumberDecimal("119.99"), "currency": "USD" },
  "reviews": [
    {
      "review_id": 786,
      "review_author": "Kristina",
      "review_text": "This is indeed an amazing widget.",
      "published_date": ISODate("2019-02-18")
    }
    ...
    {
      "review_id": 777,
      "review_author": "Pablo",
      "review_text": "Amazing!",
      "published_date": ISODate("2019-02-16")
    }
  ]
}
```

Bewertungs-collection

```
{
  "review_id": 785,
  "product_id": 1,
  "review_author": "Trina",
  "review_text": "Nice product. Slow shipping.",
  "published_date": ISODate("2019-02-17")
}
...
{
  "review_id": 1,
  "product_id": 1,
  "review_author": "Hans",
  "review_text": "Meh, it's okay.",
  "published_date": ISODate("2017-12-06")
}
```

https://docs.mongodb.com/manual/tutorial/model-referenced-one-to-many-relationships-between-documents/

- Potenzielles Probleme beim Embedded Document Pattern:
  - Dokument auf der one-Seite kann groß werden
  - Eingebettetes Feld (die n-Seite) kann potenziell unendlich wachsen
  - z.B. bei eingebetteten Logs, Bewertungen, Kommentare…
- Subset-Pattern:
  - Eingebettetes Feld enthält nur Teilmenge der N-Seite (z.B. die neuesten Bewertungen)
  - N-Seite als jeweils ein Dokument in eigener Collection, Referenz auf one-Seite (product_id)
  - Problem: Duplikate (z.B. die aktuellsten Bewertungen in zwei Collections)

MongoDB: One-to-Many Relationships: Document References

embedded data model

```
{
   title: "MongoDB: The Definitive Guide",
   author: [ "Kristina Chodorow", "Mike Dirolf" ],
   published_date: ISODate("2010-09-24"),
   pages: 216,
   language: "English",
   publisher: {
              name: "O'Reilly Media",
              founded: 1980,
              location: "CA"
            }
}

{
   title: "50 Tips and Tricks for MongoDB Developer",
   author: "Kristina Chodorow",
   published_date: ISODate("2011-05-06"),
   pages: 68,
   language: "English",
   publisher: {
              name: "O'Reilly Media",
              founded: 1980,
              location: "CA"
            }
}
```

- Einbettung kann zu Wiederholung von Daten führen
- Lösung: Referenzen verwenden

https://docs.mongodb.com/manual/tutorial/model-referenced-one-to-many-relationships-between-documents/

MongoDB: One-to-Many Relationships: Document References

- Wachstum der Beziehung bestimmt , welche Seite der Beziehung die Referenz speichern soll (One Seite bei One-to-Many, N-Seite bei one-to-squillions)
- Mögliches unendliches Wachstum vermeiden (maximale Dokumentengröße:16 MB)
- Beispiel: Anzahl der Bücher eines Verlags könnten unendlich wachsen, deshalb auf der “Buch-Seite” die id/Referenz auf den Verlag speichern

```
{
    name: "O'Reilly Media",
    founded: 1980,
    location: "CA",
    books: [123456789, 234567890, ...]
}
```

https://docs.mongodb.com/manual/tutorial/model-referenced-one-to-many-relationships-between-documents/

```
{
    _id: "oreilly",
    name: "O'Reilly Media",
    founded: 1980,
    location: "CA"
}

{
    _id: 123456789,
    title: "MongoDB: The Definitive Guide",
    author: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    publisher_id: "oreilly"
}

{
    _id: 234567890,
    title: "50 Tips and Tricks for MongoDB Developer",
    author: "Kristina Chodorow",
    published_date: ISODate("2011-05-06"),
    pages: 68,
    language: "English",
    publisher_id: "oreilly"
}
```

MongoDB: One-to-Many Relationships: Auswahl des richtigen Design-Schemas

- Unterteile Beziehungen in: one-to-few, One-to-many, One-to-squillions
- Fragestellungen:
  - Müssen die Entitäten auf der "N"-Seite der One-to-N-Beziehung allein stehen
  - Wie groß ist die Kardinalität der Beziehung: one-to-few, one-to-many oder one-to-Squillions
- Anhand dieser Fragen eines der drei grundlegenden One-to-N-Schema-Designs auswählen:
  - Einbetten der N-Seite, wenn die Kardinalität one-to-few ist und kein Bedarf besteht, auf das eingebettete Objekt außerhalb des Kontexts des übergeordneten Objekts zuzugreifen
  - Verwendung eines Arrays von Verweisen auf die Objekte der N-Seite, wenn die Kardinalität One-to-Many ist oder wenn die Objekte der N-Seite aus irgendwelchen Gründen allein stehen sollen
- Verweis auf die One-Side in den N-Side-Objekten, wenn die Kardinalität One-to-squillions ist

MongoDB: Many-to-Many Relationships: two-way-Embedding

- Grundidee: Gleichgewicht von Beziehungen herstellen
- Schätze maximale Größe von N und M.
- Für ungleiche Beziehungen one-way-Embedding wählen (nur eine Seite mit Referenz auf andere Seite)
- Für gleiche Beziehungen two-way-Embedding wählen (Beide Seiten Speichern Referenz (id) auf jeweils andere Seite)
  - z. B. N = 3 Kategorien, M = 500000 Bücher
    - one-way-Embedding wählen
  - z. B. N = 3 Kategorien, M = 5 Bücher
    - two-way-Embedding wählen

Dokumente aus Autoren-Collection

```
{
  _id: 1,
  name: "Peter Standford",
  books: [1, 2]
}
{
  _id: 2,
  name: "Georg Peterson",
  books: [2]
}
```

Dokumente aus Bücher-Collection

```
{
  _id: 1,
  title: "A tale of two people",
  categories: ["drama"],
  authors: [1, 2]
}
{
  _id: 2,
  title: "A tale of two space ships",
  categories: ["scifi"],
  authors: [1]
}
```

http://learnmongodbthehardway.com/schema/schemabasics/#many-to-many-n-m

MongoDB: Many-to-Many Relationships: one-way-Embedding

- Optimiert die Leseleistung einer N:M-Beziehung
- Verweise auf eine Seite der Beziehung über Referenz
- Verhindert, dass Dokumentengröße von 16 MB überschritten wird (im Beispiel bei Kategorie)

Dokumente aus Kategorie-Collection

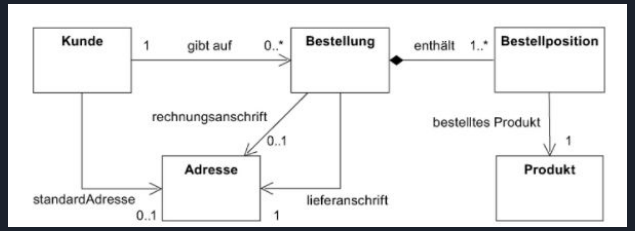
```
{
  _id: 1,
  name: "drama"
}
```

Dokumente aus Buch-Collection

```
{
  _id: 1,
  title: "A tale of two people",
  categories: [1],
  authors: [1, 2]
}
```

http://learnmongodbthehardway.com/schema/schemabasics/#many-to-many-n-m

Schema-Design: Vorgehen (1)



https://learning.oreilly.com/library/view/mongodb/9781492015499/ (Trelle, Tobias: MongoDB Der praktische Einstieg dpunkt.Verlag 278 Seiten, Softcover Erschienen 07/2014 ISBN 978-3-86490-153-9)

- Ausgangsbasis: Fachliches Modell des zu entwickelndes Systems, z.B. mit UML Diagramm
- Checkliste Fragestellungen:
  - Was sind die zentralen Geschäftsobjekte?
  - Wie eng bzw. lose sind diese miteinander gekoppelt?
  - Wie gestaltet sich die Kardinalität und Navigierbarkeit dieser Beziehungen?
  - Welches sind die häufigsten Abfragen und auf welchen Geschäftsobjekten werden sie ausgeführt?
  - Wie ist das Verhältnis zwischen Schreib- und Leseoperationen auf den Geschäftsobjekten?
  - Wie sieht das Mengengerüst der Geschäftsobjekte aus?

Schema-Design: Vorgehen (2)

- MongoDB Rahmenbedingungen
  - Dokumente können Arrays und eingebettete Dokumente enthalten
  - Maximale Dokumentengröße:16 MB
  - Referenzen auf Dokumente in anderen Collections sind möglich
  - Schreibende Operationen (Einfügen, Ändern) sind auf Ebene eines einzelnen Dokuments atomar.
  - Keine Transaktionen, für Operationen auf mehreren Dokumenten
  - Keine Joins zwischen Collections.
  - Abfragen können Einschränkungen auf Arrays und eingebettete Dokumente enthalten

Zusammenfassung: Schema-Design

- Passender Ansatz (embedded, normalized Data Model) hängt von jeweils geplanten Einsatzumfeld ab
- Vorteile der Einbettung:
  - Datenlokalität innerhalb eines Dokuments
  - Atomare Aktualisierungen an einem Dokument vorzunehmen
- Nachteil Einbettung: Geringere Flexibilität

Installation von MongoDB

1. Die Seite https://www.mongodb.com aufrufen
2. Oben rechts auf “Try Free” klicken
3. “On-premises” auswählen
4. “MongoDB Community Server” anklicken und rechts auf den grünen “Download” Button klicken
5. Die heruntergeladene Datei installieren (einfach durchklicken für Standard Einstellungen)
6. Den Pfad zum MongoDB “bin” Ordner zu Path unter Systemvariablen hinzufügen (z.B. C:\Program Files\MongoDB\Server\5.0\bin zu Path hinzufügen)
7. Mit “md c:datadb” das Standard-Datenbank-Verzeichnis für MongoDB erstellen

Genauere Anleitung: https://www.opc-router.de/was-ist-mongodb/

Überprüfen der Installation/erste Schritte

- Mit Befehl “mongo” die DB starten
- Mit Befehl “show dbs” die vorhanden Datenbanken einsehen
- Mit Befehl “use NeuerDatenbankName” wird eine neue Datenbank mit dem gewählten Namen erzeugt und diese als derzeitige Datenbank festgelegt
- Mit Befehl “use DatenbankName” kann ebenfalls zu einer vorhandenen Datenbank gewechselt werden
- Mit Befehl “db.items.insert({name: "test", age:21, birthday: "01.01.2000"})” wird ein neuer Dateneintrag (document) erstellt. “items” ist hierbei der Name der Collection
- Mit Befehl “db.items.find()” werden die Einträge (documents) in der Collection “items” angezeigt
- Mit Befehl “db.items.deleteOne({name: “test”})” wird ein vorhandener Dateneintrag (document) gelöscht
- Mit Befehl “db.items.drop()” wird die Collection “items” gelöscht

MongoDB Crud Operationen: Create

- db.collection.insertOne()
- db.collection.insertMany()
- Beispiel:

```
 db.items.insert({name: "test", age: 21, birthday: "01.01.1900"})
riteResult({ "nInserted" : 1 })
 db.items.find()
 "_id" : ObjectId("61716e76f8895dc3eb78c755"), "name" : "test", "age" : 21, "birthday" : "01.01.1900" }
```

MongoDB Crud Operationen: Read

- db.collection.find()
- Angabe verschiedener Filter/ Kriterien innerhalb von find()
- Beispiel:

```
> db.items.find({age: 21})
{ "_id" : ObjectId("61716e76f8895dc3eb78c755"), "name" : "test", "age" : 21, "birthday" : "01.01.1900" }
>
```

MongoDB Crud Operationen: Update

- db.collection.updateOne()
- db.collection.updateMany()
- db.collection.replaceOne()
- Beispiel:

```
> db.items.updateMany({age: 21}, {$set: {name: "update"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.items.find()
{ "_id" : ObjectId("61716e76f8895dc3eb78c755"), "name" : "update", "age" : 21, "birthday" : "01.01.1900" }
>
```

MongoDB Crud Operationen: Delete

- db.collection.deleteOne()
- db.collection.deleteMany()
- Beispiel:

```
> db.items.deleteOne({name: "update"})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.items.find()
>
```

Was ist KMongo ?

- Framework, ermöglicht leichtes Arbeiten mit MongoDB unter Kotlin
- Object-Document-Mapper
- Kotlin toolkit für Mongo
- Verfügbar als Kotlin Erweiterung
- Nutzt die core MongoDB java driver API
- Unterstützt synchrone und asynchrone java driver
- Bietet ein type-safe query system
- Unterstützt Mongo shell queries

Installation von KMongo

Voraussetzung: IDE: IntelliJ, Projekt: Ktor, Build: Gradle Kotlin

1. In der “build.gradle.kts” unter dependencies "implementation("org.litote.kmongo:kmongo:4.3.0")” einfügen
2. Gradle neu laden (sollte rechts oben ein icon dafür auftauchen)
3. In der Applikation “import org.litote.kmongo.*” einfügen

Probleme beim Installieren

Unter Windows soweit keine Probleme :-)

Beispielcode

```
package com.example

import org.litote.kmongo.*

fun main() {
    data class Items(val name: String)

    val client = KMongo.createClient() //get com.mongodb.MongoClient new instance
    val database = client.getDatabase( databaseName: "TestDB") //normal java driver usage
    val col = database.getCollection<Items>() //KMongo extension method
    //here the name of the collection by convention is "items"
    //you can use getCollection<items>("otherItems") if the collection name is different

    col.insertOne(Items( name: "Max"))
    val user1 : Items? = col.findOne( filter: Items::name eq "Max")

    if (user1 != null) {
        println(user1.name)
    }
}
```

Sicherheit

- MongoDB bietet verschiedene Funktionen wie:
  - Authentifizierung,
  - Zugriffskontrolle
  - Verschlüsselung,
- Wichtigste Sicherheitsfunktionen:

Authentication	Authorization	TLS/SSL
Authentication	Role-Based Access Control	TLS/SSL (Transport Encryption)
SCRAM	Enable Access Control	Configure `mongod` and `mongos` for TLS/SSL
x.509	Manage Users and Roles	TLS/SSL Configuration for Clients

Enterprise Only	Encryption
Kerberos Authentication	Client-Side Field Level Encryption
LDAP Proxy Authentication	
Encryption at Rest	
Auditing	

https://docs.mongodb.com/manual/security/

Sicherheitscheckliste mit einer Liste empfohlener Maßnahmen zum Schutz einer MongoDB:
https://docs.mongodb.com/manual/administration/security-checklist/

Quellen

- https://www.bigdata-insider.de/was-ist-nosql-a-615718/ (zuletzt aufgerufen am 29.10.2021)
- https://aws.amazon.com/de/nosql/ (zuletzt aufgerufen am 29.10.2021)
- https://www.mongodb.com/de-de (zuletzt aufgerufen am 29.10.2021)
- https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design-part-1
- https://litote.org/kmongo/ (zuletzt aufgerufen am 29.10.2021)
- https://www.opc-router.de/was-ist-mongodb/ (zuletzt aufgerufen am 29.10.2021)
- https://www.ionos.de/digitalguide/hosting/hosting-technik/dokumentenorientierte-datenbank
- https://aws.amazon.com/de/nosql/document/ (zuletzt aufgerufen am 29.10.2021)
- http://learnmongodbthehardway.com/schema/schemabasics/#many-to-many-n-m
- Shannon Bradshaw, Eoin Brazil, and Kristina: MongoDB: The Definitive Guide, Third Edition Chodorow (O'Reilly), 2019, (https://learning.oreilly.com/library/view/mongodb-the-definitive/9781491954454/) (zuletzt aufgerufen am 29.10.2021)
- Peter Bakkum Kyle Banker Shaun Verch, Douglas Garrett, and Tim Hawkins: MongoDB in Action, Second Edition: Covers MongoDB version 3.0, Manning Publications, April 2016, (https://learning.oreilly.com/library/view/mongodb-in-action/9781617291609/) (zuletzt aufgerufen am 29.10.2021)

Quellen

- https://docs.mongodb.com/guides/server/introduction/ (zuletzt aufgerufen am 29.10.2021)
- https://medium.com/@rinu.gour123/mongodb-data-modeling-with-document-structure-a6e69de8e37f (zuletzt aufgerufen am 29.10.2021)
- https://www.guru99.com/what-is-mongodb.html (zuletzt aufgerufen am 29.10.2021)
- https://www.tutorialspoint.com/mongodb/mongodb_data_modeling.htm (zuletzt aufgerufen am 29.10.2021)
- Rick Copeland: MongoDB Applied Design Patterns, O'Reilly Media, Inc., 2013, (https://learning.oreilly.com/library/view/mongodb-applied-design/9781449340056/) (zuletzt aufgerufen am 29.10.2021)
- Amit Phaltankar, Juned Ahsan, Michael Harrison, Liviu Nedov: MongoDB Fundamentals, Packt Publishing, 2020, (https://learning.oreilly.com/library/view/mongodb-fundamentals/9781839210648/) (zuletzt aufgerufen am 29.10.2021)
- Trelle, Tobias: MongoDB Der praktische Einstieg dpunkt.Verlag 07/2014 (https://learning.oreilly.com/library/view/mongodb/9781492015499/) (zuletzt aufgerufen am 29.10.2021)