5 MongoDB

Sonntag, 10. September 2023 16:07

5 MongoDB

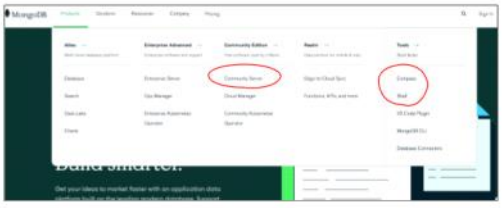
5 NOSQL – MongoDB

MongoDB ist ein universelle, dokumentenbasiertes NoSQL-Datenbankmanagementsystem für die moderne Anwendungsentwicklung und die Cloud. Es gehört zu den meisten genutzten NoSQL-Datenbanken und ist sehr weit verbreitet. Das Format der Dokumente, welche in MongoDB verwaltet werden, sind dem JSON-Format sehr ähnlich. MongoDB ist Open-Source und für Windows, MacOS und Linux verfügbar.[4]

5.1 Installation und Einrichtung von MongoDB

Für die weitere Arbeit mit MongoDB ist es nötig, dass Sie lokal einen MongoDB Server auf Ihrem Computer haben. Gehen Sie dabei wie folgt vor:

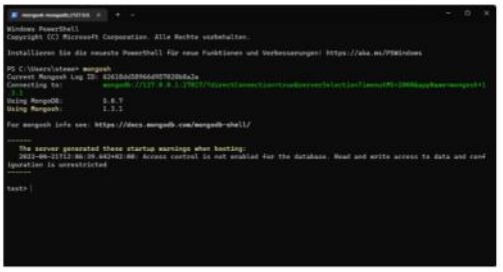
1. **Schritt:** Laden Sie sich den Community Server auf https://www.mongodb.com herunter.



Zusätzlich können Sie sich auch unter Tools die MongoDB Shell und Compass herunterladen.

[4] IT-Berufe: Fachstufe Technische IT-Berufe Lernfelder 6-9, Westermann Verlag, S. 287.

2. **Schritt**: Nach der erfolgreichen Installation starten Sie die Mongo Shell. Dazu starten Sie das Terminal und geben den Befehl *mongosh* ein und bestätigen diesen.



Im weiteren Verlauf werden wir nun mit der MongoDB arbeiten.

5.2 CRUD

Unter dem Akronym[5] **CRUD** werden die 4 grundlegenden Datenbankoperationen verstanden, welche Ihnen bereits aus dem Bereich SQL bekannt sind.

Arbeitsauftrag

a) **Weisen** Sie die entsprechenden **SQL-Befehle** den **Datenbankoperationen** zu. Es ist nur ein Befehl zu benennen.

b) **Finden** Sie die entsprechenden **Befehle** für **MongoDB** mittels **selbstständiger Internetrecherche**.

	Bedeutung	SQL	MongoDB
Create	Datensatz erzeugen		
Read	Datensätze lesen		
Update	Datensätze ändern		
Delete	Datensätze löschen		

[5] Aus den Anfangsbuchstaben oder -silben mehrerer Wörter oder der Bestandteile eines Kompositums gebildetes Kurzwort (z. B. EDV aus elektronischer Datenverarbeitung, Kripo aus *Kri*minal*po*lizei)

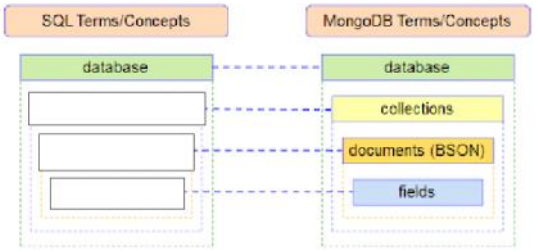
5.3 Kollektionen

Eine Datenbank bleibt eine Datenbank! Auch MongoDB bleibt, als eine NoSQL-Datenbank, eine Datenbank. Doch in dieser finden wir keinen Tabellen. MongoDB ist, wie bereits beschrieben, eine dokumentenbasierte Datenbank.

Die folgende Grafik[6] zeigt den schematischen Aufbau.

Arbeitsauftrag

Betrachten Sie den grafischen Vergleich zwischen SQL und MongoDB. Vervollständigen Sie die Grafik und weisen Sie den Begriffen aus MongoDB das Äquivalent aus SQL zu.



[6] Quelle: https://www.geeksforgeeks.org/mongodb-database-collection-and-document/

Nun betrachten wir uns die Datenbanken in MongoDB im Detail und konkret an. Um sich alle Datenbanken anzuschauen, werden in MongoDB ähnlich Befehle verwendet wie

	Befehl MongoDB
Alle Datenbanken anzeigen	`show dbs`
Datenbank benutzen	`use namedatenbank`
Aktuelle Datenbank anzeigen	db

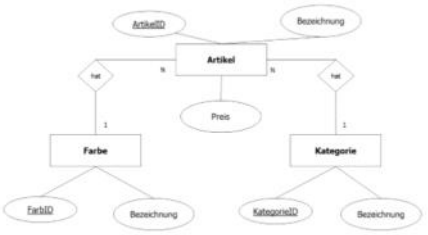
Da Sie nun die Datenbanken kennen, wählen Sie eine Datenbank mit dem entsprechenden Befehl aus und lassen sich die Kollektionen anzeigen.

	Befehl MongoDB
Kollektionen anzeigen	`show collections`
Kollektion löschen	`db.meineKollektion.drop()`

In einer Kollektion können beliebig strukturierte JSON-Dokumente gespeichert werden. Üblicherweise strukturiert man Dokumentendatenbanken jedoch meistens so, dass die Dokumente innerhalb einer Kollektion in etwa das gleiche Schema haben. Anders als bei SQL ist es aber nicht notwendig, dieses Schema vorher festzulegen. Es muss noch nicht einmal eine Kollektion erstellt werden. Sobald das erste Dokument in eine noch nicht existierende Kollektion eingefügt wird, ist sie existent.

5.4 Einfügen von Dokumenten

Arbeitsauftrag

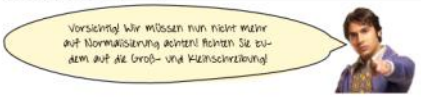


a) **Überführen** Sie das ER-Modell zur KOSidasdb in eine **JSON-Datei** mit folgenden Beispiel-Daten:

ArtikelNr.	Bezeichnung	Farbe	Kategorie
134456	Sportschuh River	Schwarz	Schuhe
134457	Sprotschuh River	Blau	Schuhe
134465	UEFA EURO 2016 Deutschland	Weiss	Kleidung
134483	FIFA WM 2014 Ball	Weiss	Accessoires

b) Geben Sie das **JSON-Objekt** nun in die **Datenbank** der MongoDB ein.

	Befehl MongoDB
Dokument einfügen	`db.nameKollektion.insert()`
Kollektion löschen	`db.meineKollektion.drop()`



c) **Informieren** Sie sich mittels **selbstständiger Recherche**, wie MongoDB mit dem Feld **"_id"** umgeht? Was ist zu berücksichtigen?

5.5 Dokumente finden

Jetzt wollen wir alle Dokumente auslesen. MongoDB kennt dabei folgende Befehle:

	Befehl MongoDB
Alle Dokumente lesen	`db.meineKollektion.find()`
Zufälliges Dokument aus Kollektion	`db.meineKollektion.findOne()`

Sortieren und Limitieren
Ähnlich wie in SQL können die Ausgaben auch in MongoDB sortiert und limitiert werden.

	Befehl MongoDB
Sortierung	`db.meineKollektion.find().sort()`
Limitierung	`db.meineKollektion.find().limit()`

Der Befehl **sort()** erwartet, ob das nach dem zu sortierenden Feld aufsteigend oder absteigend sortiert werden soll:

Aufsteigend sortiert	**Absteigend sortiert**
`.. sort( { feld : } )`	`.. sort( { feld : } )`

Der Befehl limit() erwartet die Anzahl der limitierenden Dokumente:

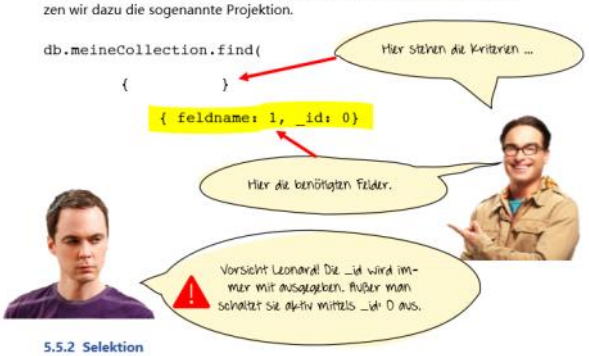
5 Dokumente sollen ausgegeben werden

Arbeitsauftrag
Prüfen Sie nun selbst in der von Ihnen angelegten Kollektion nach, ob die Daten vorhanden sind. Verwenden Sie beide oben aufgeführte Befehle.

5.5.1 Projektion

Diese einfachen Abfragen können nun verfeinert werden. In SQL würden wir nun die Spaltenliste (SELECT spaltenname1, ...) definieren. In MongoDB nutzen wir dazu die sogenannte Projektion.

```
db.meineCollection.find(
        {          }
    { feldname: 1, _id: 0}
```



5.5.2 Selektion

Bei Selektion werden die Kriterien für die Eingrenzung festgelegt.

```
db.meineCollection.find(
    { feldname: "Bedingung"}
        { feldname: 1, _id: 0}
```

Zudem sind auch Vergleichsoperatoren möglich:

Vergleichsoperatoren	**Umsetzung MongoDB**
Kleiner / größer	`$lt / $gt`
Kleiner gleich / größer gleich	`$lte / $gte`
Gleich / ungleich	`$eq / $ne`
Existiert	`$exists`

Beispiele

```
db.personen.find({name: "Max"})
db.personen.find({geboren: {$gt:1960}})
db.personen.find({geboren: {$gt:1960, $lte: 2000}})
db.personen.find({name: "Max",geboren: 1960})
db.personen.find({geboren: {$exists: true}})
```

5.5.3 ODER-Verknüpfung

Die oben aufgezeigten Bedingungen in der Selektion sind UND-Verknüpfungen. Aber es gibt auch die Möglichkeit mit ODER-Verknüpfungen in MonogoDB zu arbeiten.

```
db.meineCollection.find( { $or : [
    { feldname: "Bedingung"},
        { feldname: "Bedingung"}
])
```

Natürlich können in das formulierte Array [] mehr als 2 Bedingungen gepackt werden.

5.5.4 Dokumente zählen

Wenn man nicht die Ergebnisse einer Abfrage wissen möchte, sondern nur die Anzahl der gefundenen Dokumente, kann die count-Methode hilfreich sein.

```
db.meineCollection.count()

db.meineCollection.find(
    { feldname: "Bedingung"}).count()
```


5.6 Änderungsoperationen

Ähnlich wie bei SQL sorgt der Befehl update für eine Änderungen.

Beispiel

```
db.Mitarbeiter.update ( {_id : 5 },
            { name : "Max" , gehalt : 1344.88} )
```

Neben der ID als Kriterium für das Dokument, können auch andere Kriterien (Felder) zur Eingrenzung von Dokumenten oder Auswahl mehrerer Dokumente herangezogen werden.

Beim Update von mehreren Dokumenten (Multi-Update) muss jedoch darauf geachtet werden, dass am Ende „multi : true" verwendet wird.

Beispiel

```
db.Mitarbeiter.update ( {geboren :  {$gt:2000}},
            { geboren : "zu jung"} , {multi: true} )
```

Eine weitere Besonderheit bei MongoDB ist der sogenannte Upsert. Er ist eine Mischung eines Updates mit einem Insert. Ist das im Update-Befehl angegebene Kriterium für mind. ein Dokument erfüllt, so wird das Update wie gehabt ausgeführt (Dokument wird ersetzt!). Falls nicht, wird ein neues Dokument, welches im zweiten Parameter angegeben ist, neu eingefügt.

Beispiel

```
db.Mitarbeiter.update ( {name : "Max M."}
            { name : "Maximilian"} , {upsert: true} )
```

Wird allerdings im 1. Oder 2. Parameter eine id angegeben, erhält das neue Dokument diese. Andernfalls wird diese automatisch generiert.

Möchte man nicht das ganze Dokument ersetzen, sondern nur gewisse Werte verändern, während die anderen Felder unverändert bleiben, so bietet sich folgende Ergänzung des Update-Befehls an:

Beispiel

```
db.Mitarbeiter.update ( {_id : 7}
            { $set : {gehalt: 7000}})
```

Weitere Operationen im Update-Befehl sind:

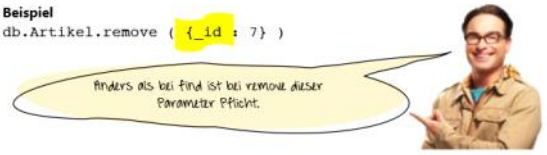
	Umsetzung MongoDB
Wert um 1 erhöhen / senken	`$inc`
Feld entfernen	`$unset`

5.7 Dokumente löschen

An dieser Stelle wollen wir uns dem Löschen von Dokumenten widmen. Hierfür wird der Remove-Befehl verwendet.

Beispiel

```
db.Artikel.remove ( {_id : 7} )
```



Genau wie bei find und update können hier neben einfachen Vergleichen auch die bereits bekannten Operationen (z.B. $gt, $or, ...) verwendet werden.

Beispiel

```
db.Mitarbeiter.remove ( {geboren: {$gt:1990}} )
```

Beispiel

```
1.db.meineKollektion.remove ( {} )

2.db.meineKollektion.drop ()
```

Bei Variante 1 werden alle Dokumente nach und nach gelöscht. Die dann leere Kollektion bleibt weiterhin bestehen. Bei der Variante 2 ist deutlich effizienter und löscht die Kollektion komplett.

5.8 Komplexe Datentypen

Wie bei JSON-Dokumenten bereits angesprochen können neben atomaren Werten (z.B. Zahlen, Zeichenketten, Datum, ...) auch Subdokumente und Arrays enthalten sein. Wie betrachten nun, wie wir die Werte dieser Felder lesen und modifizieren können.

5.8.1 Subdokumente / Dot-Notation

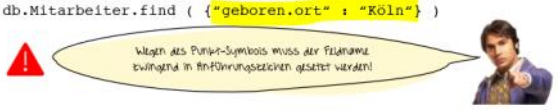
Wie betrachten ein JSON-Beispiel mit Subdokumenten:

```
{ _id: 1,
name: "Franke",
vorname:"Klaus",
geboren: {
     jahr: 2007,
     ort:"Köln" }
}
```

Nun versuchen wir die Person herauszufinden, die in Köln geboren wurde. Die Abfrage in Dot-Notation könnte in einer Variante so aussehen:

Beispiel

```
db.Mitarbeiter.find ( {"geboren.ort" : "Köln"} )
```



Wenn man die Dot-Notation nicht verwenden möchte, gibt es noch diese Varianten, welche kaum verwendet werden:

```
db.Mitarbeiter.find ( {geboren: {ort : "Köln"} } )
```

Oder mit mehreren Bedingungen:

```
db.Mitarbeiter.find ( {geboren:
            {jahr:2007, ort : "Köln"} } )
```


5.8.2 Arrays

Nun wird das JSON-Dokument von oben um weitere Felder erweitert.

```
{ _id: 1,
name: "Franke",
vorname:"Klaus",
geboren: {
     jahr: 2007,
     ort:"Köln" }
hobbys: ["Tennis", "Yoga"]
}
```

Wenn man eine Selektion, wie gewohnt, auf Arrays anwendet, wird ein Prädikat als wahr ausgewertet, wenn mindestens ein Element im Array wahr ist. Um alle Personen zu finden, die als eines ihrer Hobbys Yoga angegeben haben, ist ein einfacher Vergleich ausreichend:

Beispiel

```
db.Mitarbeiter.find ( {hobbys: "Yoga"} )
```

Wenn man alle Elemente aus dem Array haben möchte, muss der Befehl um den $all-Operator erweitert werden:

Beispiel

```
db.Mitarbeiter.find (
{hobbys: {$all: ["Yoga", "Tennis"] } })
```

Weitere Möglichkeiten mit Arrays in Verbindung mit update zu arbeiten:

	Umsetzung MongoDB
Array leeren	`$set`

Beispiel
`db.Mitarbeiter.update ( {_id: 1},`
`{$set: {hobbys:[] } })`	

Element dem Array hinzufügen	`$push`

Beispiel
`db.Mitarbeiter.update ( {_id: 1},`
`{$push: {hobbys:"Schwimmen" } })`	

Entfernen von Elementen aus dem Array **mit Dopplungen**	`$pull`

Beispiel
`db.Mitarbeiter.update ( {_id: 1},`
`{$pull: {hobbys:"Schwimmen" } })`	

Elemente zum Array hinzufügen **ohne Dopplungen**	`$addToSet`

Beispiel
`db.Mitarbeiter.update ( {_id: 1},`
`{$addToSet: {hobbys:"Schwimmen" } })`	

5.9 Gesamtübungsaufgaben MongoDB

Nutzen Sie die MongoDB „musik" mit der Collection „artikel".

a) Fügen Sie ein weiteres Produkt Ihrer Wahl ein. Halten Sie sich an das Schema der bisherigen Produkte.

b) Geben Sie alle Produkte des Herstellers „Yomoho" aus, die mehr als 100,00 Euro kosten.

c) Welche Produkte sind in der Kategorie „Zubehoer" oder „Noten"? Geben Sie nur deren Produktbezeichnung und die _id aus.

d) Wie viele Produkte haben das Schlagwort „jazz"?

e) Was sind die 2 teuersten Produkte?

f) Ändern Sie den Preis des Produktes „Klavier" auf 3800.

g) Fügen Sie allen Produkten, für die eine Seitenzahl angegeben wurde, das Schlagwort „buch" hinzu. Achten Sie darauf, dass danach kein Artikel dieses Schlagwort zweimal hat.