DB Crashkurs (MongoDB)

Christian Lehner

Warum MongoDB?

- dokumentenorientierte DB (Dokumente = JSON)
- Dokumente besitzen kein bestimmtes Schema
- NoSQL
- Collection <—> Tabelle

MongoDB start + Shell

- Mongo Daemon: mongod
- Mongo Shell: mongo
- Mongo Shell ist eine JavaScript Umgebung

Collection erzeugen

```
db.createCollection(“name”, options)
```

Collection löschen

```
db.<COLLECTION>.drop()
```

Einfügen

```
db.<COLLECTION>.insert({
  name: “Chri”,
  age: 25,
  address: {
    street: “Urstein Süd 3”,
    zip: 5412,
    city: “Puch bei Hallein”
  },
  hobbies: [“snorkelling”, “travelling”]
})
```

```
db.<COLLECTION>.insertOne({ … })
```

```
db.<COLLECTION>.insertMany([{}, {}, …])
```

Suchen

```
db.<COLLECTION>.find()
```

```
db.<COLLECTION>.find({
  key: “value”
})
```

```
db.<COLLECTION>.find({
  “key”: “value”
})
```

```
db.<COLLECTION>.findOne({
  “key”: “value”
})
```

```
db.<COLLECTION>.find({
  “key”: /Mo/
})
```

```
db.<COLLECTION>.find({
  “key”: “value”
}, {
  _id: true, name: true
})
```

Query und Projection Operators (Vergleichsoperatoren)

- $eq: equals
- $gt(e): greater than (or equal)
- $lt(e)
- $ne: not equal
- $in: ähnlich zu IN in SQL. Man übergibt ein Array und ein Dokument wird ausgewählt, wenn ein Wert vom Array gleich ist

Query und Projection Operators (logische Operatoren)

- $and
- $or
- $not
- $nor
- http://docs.mongodb.org/manual/reference/operator/query/

Query and Projection Operators

```
db.<COLLECTION>.find({
  $or: [
    { name: “Chri” },
    { hobbies: “cooking” }
  ]
})
```

```
db.<COLLECTION>.find({
  $not: {
    age: 25
  }
})
```

Update

```
db.<COLLECTION>.update({
  “name”: “Chri”
  }, {
    $set: {
      “name”: “Christian”
    }
})
```

```
db.<COLLECTION>.updateOne(query, update)
```

```
db.<COLLECTION>.updateMany(query, update)
```

Update Operators

- $inc: den Wert des Attributs um die angegebene Zahl erhöhen
- $mul: … multiplizieren
- $rename: Key umbenennen
- $set: den Wert des Key-Value-Paares neu setzen
- $unset: angegebenes Key-Value Paar löschen
- $min: update nur, wenn Wert < Wert des vorhandenen Attributes ist
- $max: … Wert > Wert des vorhandenen Attributes ist

Update Operators

```
db.<COLLECTION>.update({
  “name”: “Chri”
  }, {
    $mul: {
      age: 4
    }
})
```

```
db.<COLLECTION>.update({
  “name”: “Chri”
  }, {
    $min: {
      age: 25
    }
})
```

Update von Array Elementen

- Um Elemente von Arrays zu ändern, benötigt man den Positionsoperator: $

```
db.<COLLECTION>.update({
  “name”: “Chri”, hobbies: “snorkelling”
  }, {
    $set: {
    “hobbies.$”: “scuba diving”
    }
  }
})
```

Update von Dokumenten in Arrays

```
db.<COLLECTION>.update({
  “name”: “Chri”, “placesVisited.name”: “CZ”
  }, {
    $set: {
    “placesVisited.$.name”: “Czech Republic”
    }
  }
})
```

Delete

```
db.<COLLECTION>.remove({
  “name”: “Chri”
})
```

```
db.<COLLECTION>.deleteOne({
  “name”: “Chri”
})
```

```
db.<COLLECTION>.deleteMany({
  “name”: “Chri”
})
```

Beziehungen(?)

- Keine Joins wie in SQL
- Max Document size: 16MB
- Schema Design?
  - One-to-few (einbetten?)
  - One-to-many (normalisieren?)
- Muss man überhaupt auf die Objekte auf der n-Seite separat zugreifen?

Schema Design (1:n)

- Einbetten vs.
- Referenzen:
  - “child-referencing”
  - “parent-referencing”
  - bidirektionale Referenzierung
- Denormalisierung
- http://blog.mongodb.org/post/87200945828/6-rules-of-thumb-for-mongodb-schema-design-part-1