Prof. DI Dr. Erich Gams

Einführung und Anwendung

MongoDB

informationssysteme htl-wels

Übersicht ☛ Was lernen wir?

- Aggregate
- Modellierung
- Hands-on -> Tutorial

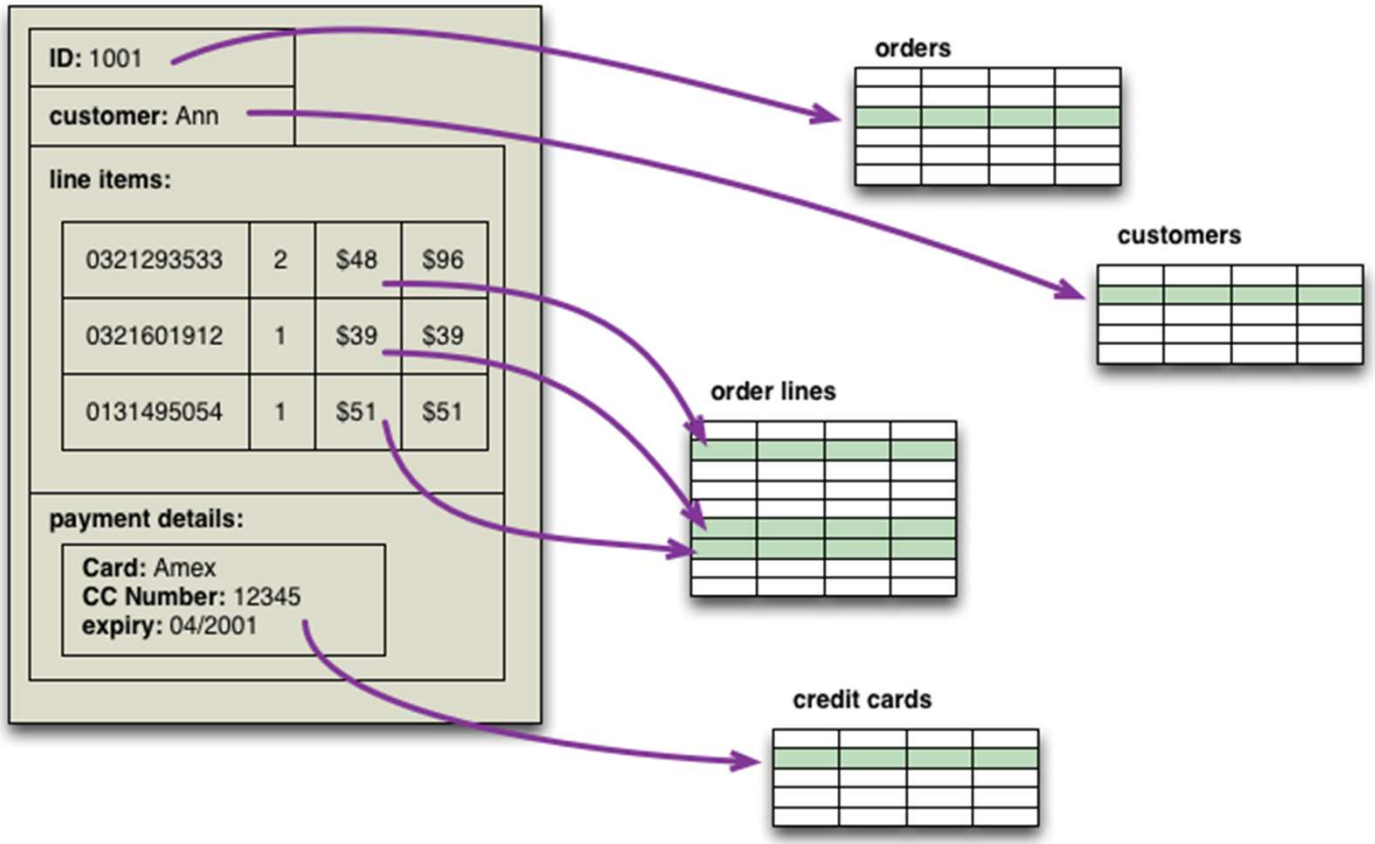
Gründe für die Nutzung von MongoDB

- Einfache Schemaanpassung
- Einfaches Key-Value-Prinzip zur Suche
- Keine Joins, sondern direkte Abfrage auf Collections
- Datenmenge: variable Größe der Datenbankdateien

Aggregate statt Relationen

- Key-Value-Stores und Dokumentendatenbanken sind „Aggregat-orientiert“.
- Begriff aus Domain-driven Design von Martin Fowler für NoSQL-Datenbanken
- Aggregat: „collection of related objects that we wish to treat as a unit“
- z.B.: ein Kunde mit seinen Bestellungen.
- Aggregate vereinfachen die Verteilung der Daten
  - Replikation
  - Sharding

Aggregate Beispiel

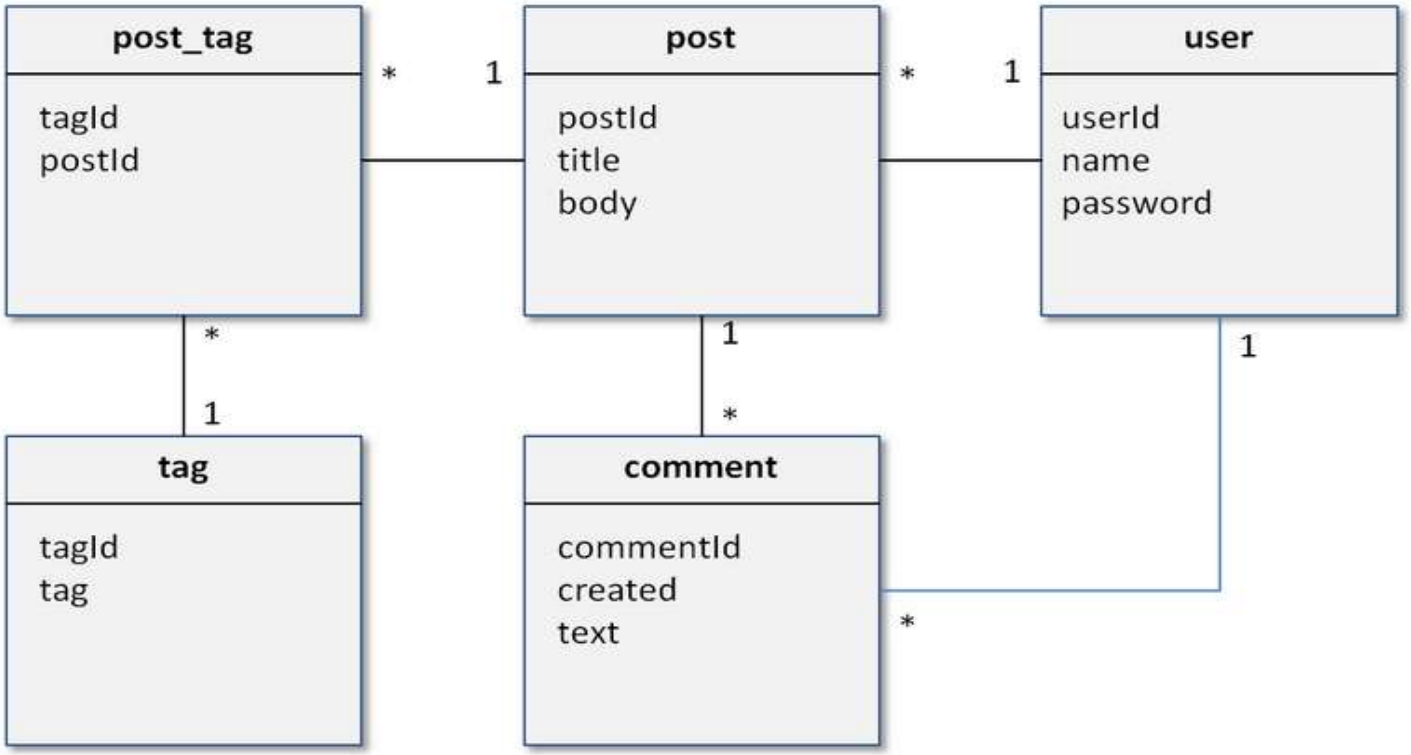


Aggregate

- Ich lege eine Strukturierung der Daten als Aggregate fest.
  - Bestellungen (orders) werden als Aggregate gespeichert.
  - Bestellungen sind leicht suchbar.
- Was passiert, wenn ich eine andere Datensicht brauche?

  Wie löse ich das Relational?
  - z.B.: Quartalszahlen von Verkäufen.
  - Anwendung des Map/Reduce Algorithmus

Aggregate Beispiel



› entnommen aus https://entwickler.de/online/datenbanken/datenmodellierung-in-nicht-relationalen-datenbanken-137872.html

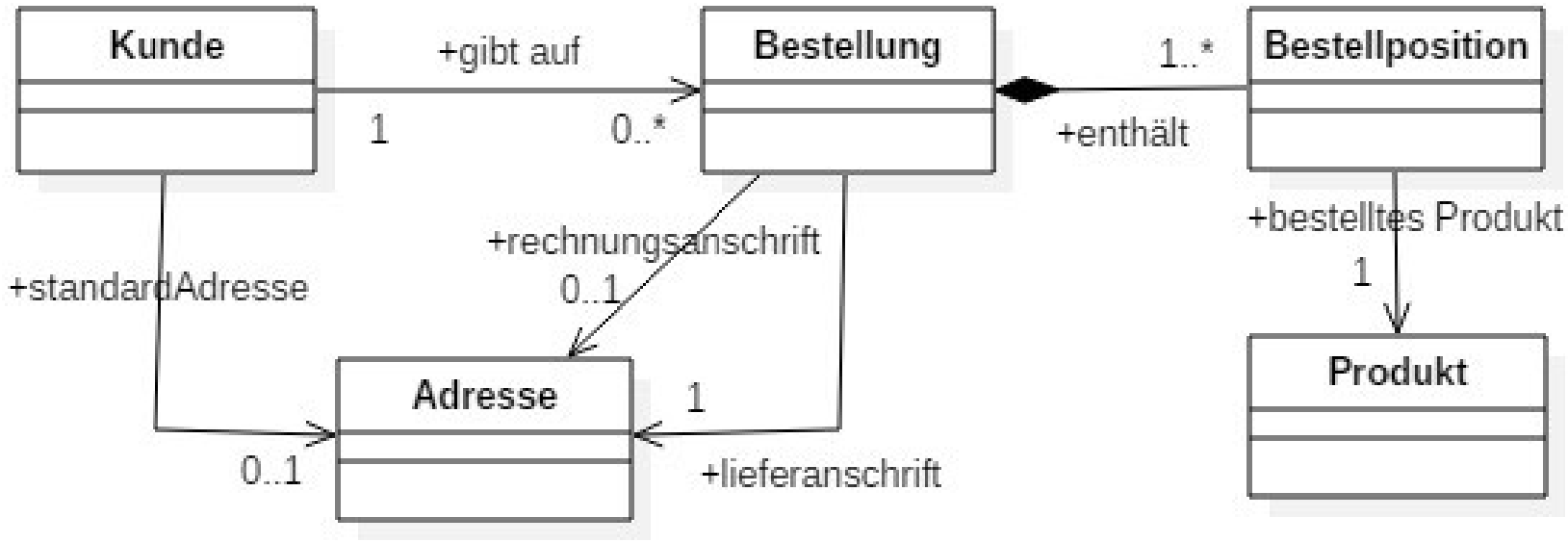
Aggregate Beispiel

Listing 1

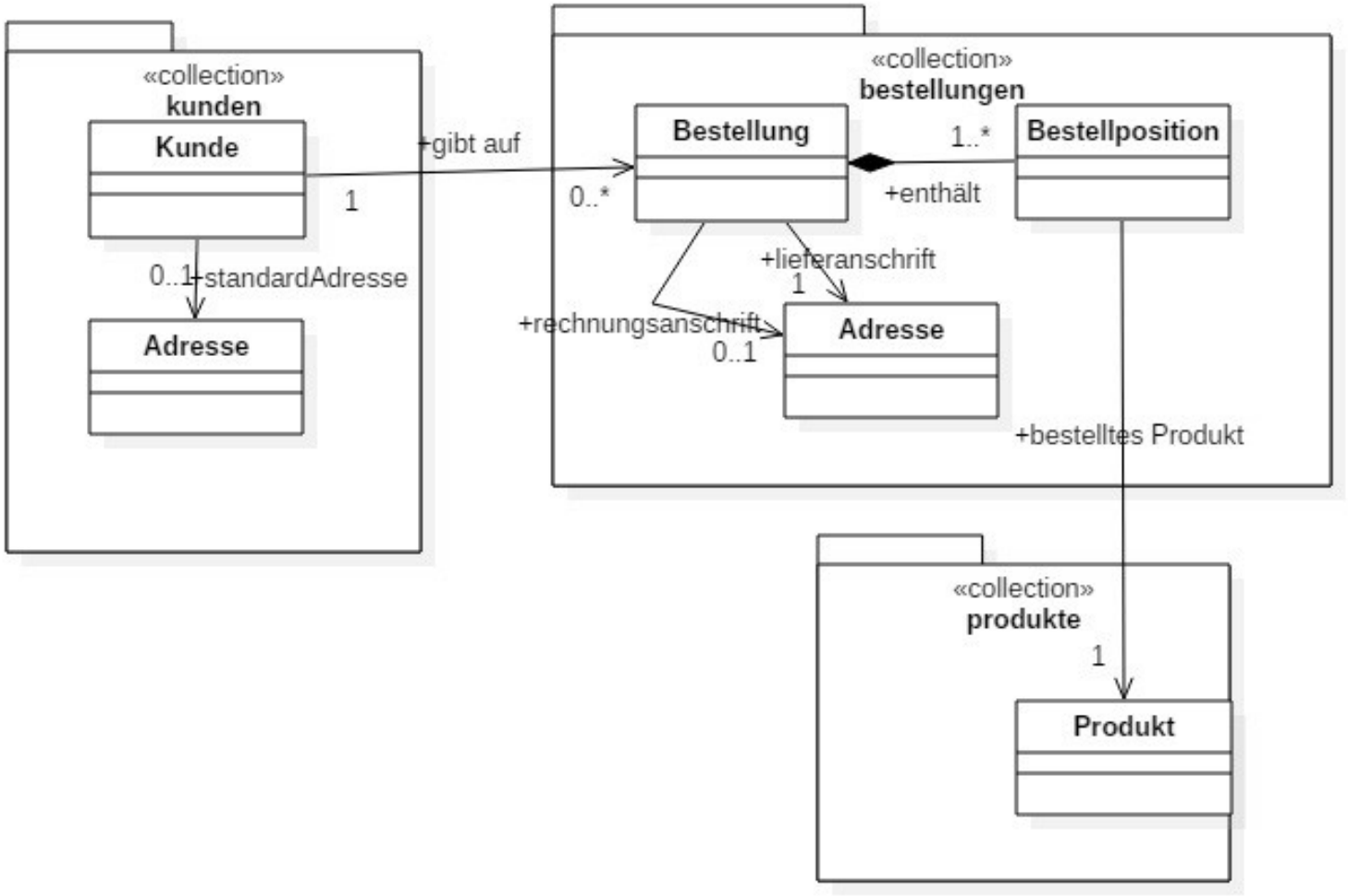
```
{
  "id": 1,
  "author": "henry",
  "created": "2012-09-05T11:19:21.000Z",
  "title": "My brand new blog rulez",
  "body": "I wrote a poem for it:...",
  "tags": ["poetry", "general", "art"],
  "comments": [{
    "author": "birdie23",
    "created": "2012-09-05T11:23:15.000Z",
    "comment": "Lovely, man!"
  }, {
    "author": "tom",
    "created": "2012-09-06T08:15.21.000Z",
    "comment": "OMG!"
  }]
}
```

```
{
"name": "henry",
  "fullname": {
    "first": "Henrik",
    "last": "Helenius"
  },
  "password": "$1$6r6dhpex$thvqpqyccicdjlotwr842/"
}
```

Shop Beispiel



Analyse


«collection»
kunden
Kunde
1
+gibt auf
0..1
+standardAdresse
Adresse
«collection»
bestellungen
Bestellung
0..*
1..*
Bestellposition
+enthält
+lieferanschrift
1
+rechnungsanschrift
0..1
Adresse
+bestelltes Produkt
«collection»
produkte
1
Produkt

DBRef vs. ObjectID

- **Referenzen** können vollqualifiziert in Form einer DBRef samt Namen der Datenbank und Collection abgespeichert werden oder als einfache ObjectID.
- Es gibt **keine referenzielle Integrität**, da die Collection im Prinzip unabhängig voneinander sind.

1:1 Beziehungen

```
Test.a
{
   _id: ObjectId(“a1“),
    b_id:ObjectId(“b1“),
}
```

```
Test.b
{
   _id: ObjectId(“b1“),
….
}
```

- // von a nach b:
- var a = db.a.find(..)
- var b = db.b.find({_id:a.b_id})
-
- // von b nach a
- var b = db.b.find(..)
- var a = db.a.find({b_id:b._id})

1:n Beziehungen

shop.bestellungen

```
{
_id: ObjectId("b0001")
positionen: [
    {
        anzahl: 1,
        product_id: ObjectId("p0001"),
        ...
    },
{
        anzahl: 2,
        product_id: ObjectId("p4711"),
        ...
    },
],
...
}
```

1:n Beziehungen

› Array von ObjectIds der referenzierten Dokumente

shop.kunden

```
{
_id: ObjectId("k0001")
bestellungen: [
   ObjectId("b0001"),
   ObjectId("b0003")
],
...
}
```

shop.bestellungen

```
{
   _id: ObjectId("b0001"),
   datum: IsoDate("2013-12-20"),
   ...
},
{
   _id: ObjectId("b0002"),
   datum: IsoDate("2013-12-20"),
   ...
},
```

1:n Beziehungen

- // bestellungen zum kunden
- var kunde = db.kunden.find(…)
- var bestellungen = db.bestellungen.find({_id: {$in: kunde.bestellungen}})

- // kunde zur bestellung
- var bestellung = db.bestellungen.find(…)
- var kunde = db.kunden.find({bestellungen: bestellung._id})

1:n Beziehungen

› Fremdschlüssel-Feld im referenzierten Dokument

shop.kunden

```
{
_id: ObjectId("k0001")
...
}
```

shop.bestellungen

```
{
    _id: ObjectId("b0001"),
    kunden_id: ObjectId("k0001"),
    datum: IsoDate("2013-12-20"),
    ...
},
{
    _id: ObjectId("b0002"),
    kunden_id: ObjectId("k0001"),
    datum: IsoDate("2013-12-20"),
    ...
},
...
```

1:n Beziehungen

- // bestellungen zum kunden
- var kunde = db.kunden.find(…)
- var bestellungen = db.bestellungen.find({kunden_id: kunde._id}}
-)
- // kunde zur bestellung
- var bestellung = db.bestellungen.find(…)
- var kunde = db.kunden.find({_id: bestellungen: kunden._id})

N:M Beziehungen

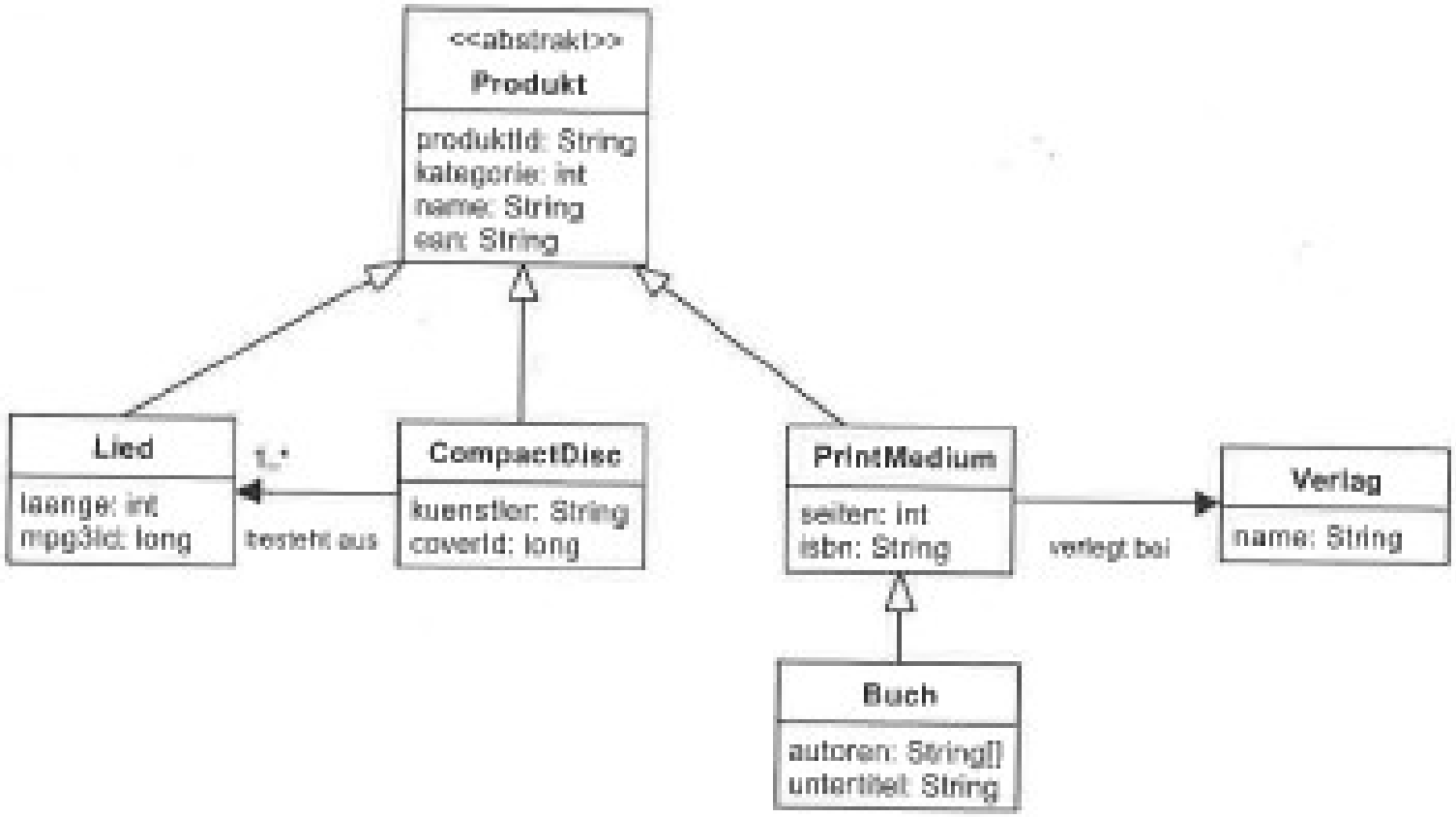
test.m

```
{
_id: ObjectId("m0001"),
n_ids: [
    ObjectId("n0002"),
    ObjectId("n0003"),
    ...
],
...
},
{
_id: ObjectId("m0002"),
n_ids: [
    ObjectId("n0001")
],
...
},
...
```

test.n

```
{
_id: ObjectId("n0001"),
m_ids: [
    ObjectId("m0002")
],
...
},
{
_id: ObjectId("n0002"),
m_ids: [
    ObjectId("m0001"),
    ObjectId("m0002"),
],
...
},
...
```

Vererbung


<<abstrakt>>
Produkt
produktId: String
kategorie: int
name: String
ean: String
Lied
laenge: int
mpg3id: long
1..*
besteht aus
CompactDisc
kuenstler: String
coverId: long
PrintMedium
seiten: int
isbn: String
verlegt bei
Verlag
name: String
Buch
autoren: String[]
untertitel: String

Vererbung

- Instanzen der Kategorie *Buch* und *Lied* abspeichern. (+ Diskriminator: *kategorie*)

```
> db.produkte.drop()
> db.produkte.insert({
  kategorie: 1,
  produktId: "4711",
  name: "Stille Nacht, Heilige Nacht",
  mpg3Id: 8573838585
})
> db.produkte.insert({
  kategorie: 5,
  produktId: "0815",
  name: "Stille Nacht",
  untertitel: "Die schönsten Weihnachtsgeschichten aus aller Welt"
})
```

- db.produkte.find({name:/StilleNacht/})

Auf los geht‘s los ;-)