
MongoDB

Eine der populärsten NoSQL-
Datenbanken im Überblick

Roadmap

- ❖ MongoDB — Schlagwörter
- ❖ NoSQL
- ❖ Grundlagen Document Stores
- ❖ Eigenschaften von MongoDB
- ❖ MongoDB in Action
- ❖ Weiterführende Funktionalität
- ❖ Zusammenfassung



MongoDB – Schlagwörter I

- ❖ Name von **humongos** (GIGANTISCH)
- ❖ Populärste NoSQL-Datenbank
- ❖ Open-Source
- ❖ Leichte Installation
- ❖ Verfügbar für gängige Betriebssysteme

MongoDB – Schlagwörter II

- ❖ Dokument-orientiert
- ❖ Schemafrei
- ❖ Hohe Flexibilität

- ❖ Für grosse Datenmengen
- ❖ Hohe Leistung

- ❖ Einfache Skalierbarkeit
- ❖ Ausfallsicherheit

NoSQL

- ❖ Was ist das?
- ❖ Warum NoSQL-Datenbanken?
- ❖ Blick zurück auf RDBMS
- ❖ Varianten von NoSQL-Datenbanken



NoSQL – Was ist das?

- ❖ Name ist unglücklich gewählt — NoSQL steht nicht für “KEIN SQL”, sondern eher für “Not Only SQL”
- ❖ Fokus ist falsch: es geht eher um KEIN relationales Datenmodell
- ❖ KEINE Kampfansage gegen traditionelle (relationale) Datenbanken
- ❖ VIELMEHR sinnvolle Ergänzung, da wo diese Schwächen zeigen
- ❖ Antwort auf den zunehmenden Bedarf ...
 - ❖ ... nach flexibler Datenspeicherung (kein fixes Modell)
 - ❖ ... nach guter, insbesondere horizontaler Skalierbarkeit

Warum NoSQL-Datenbanken?

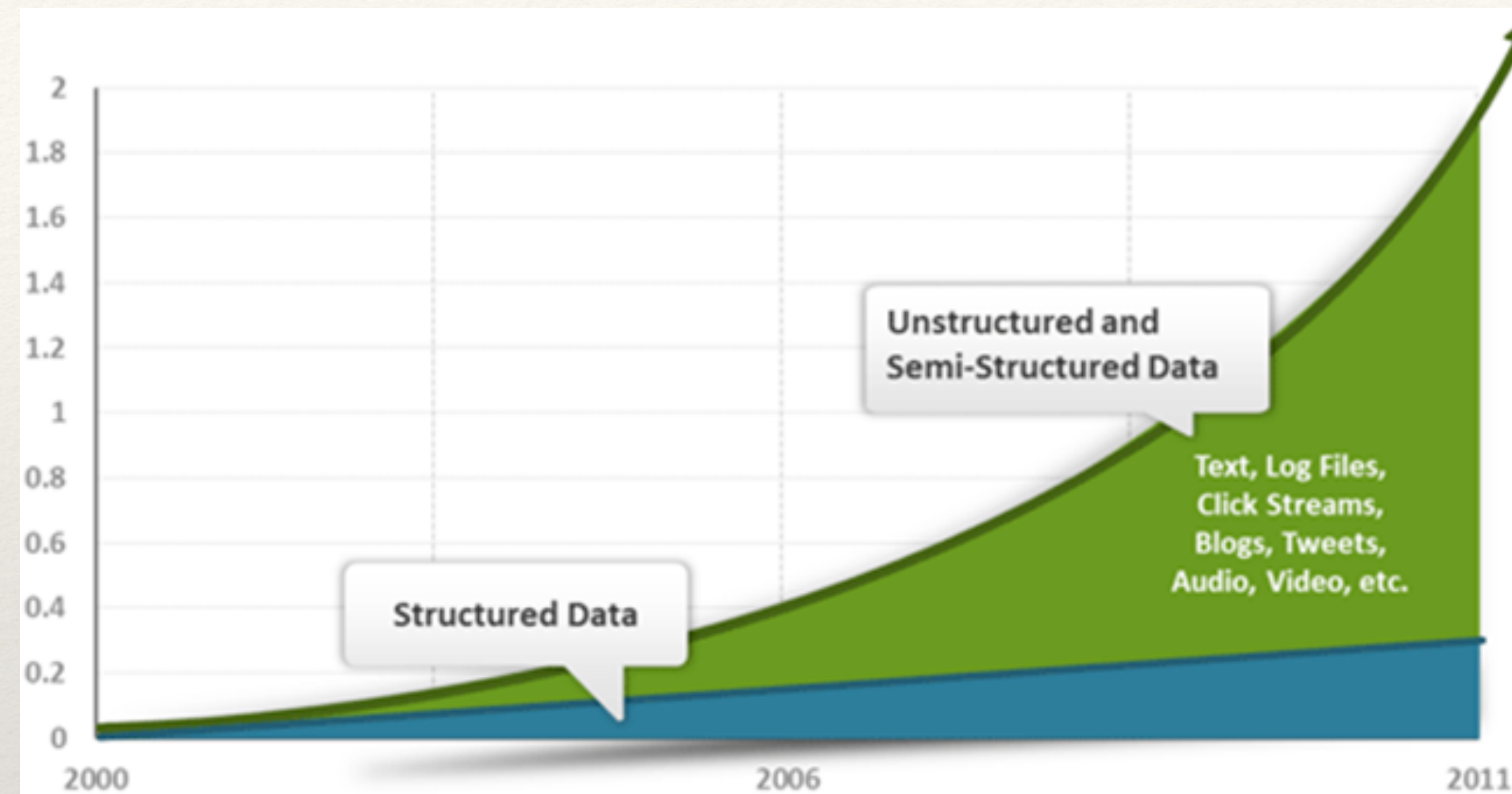
❖ Trend

- ❖ Stetig wachsende Datenmengen und Anzahl an Benutzern

- ❖ Twitter:

2010: ~65 Millionen Tweets/Tag

2011: ~200 Millionen Tweets/Tag



- ❖ Daten sind häufig immer stärker miteinander vernetzt (Facebook, XING,...)
- ❖ Daten sind weniger strukturiert, mehr Varianzen
- ❖ weniger Anforderungen an die Konsistenz
- ❖ Zunehmende Erwartungshaltung an Geschwindigkeit und Datendurchsatz

Warum NoSQL-Datenbanken?

- ❖ Zukunft: Big Data
 - ❖ Nahezu unüberschaubare Datenmengen in Tera-, Peta-Bytes und mehr
 - ❖ Verarbeitung und Speicherung stellt (grosse) Herausforderungen dar
 - ❖ Einzelner Datensatz besitzt kaum Mehrwert, die Kombination macht es
 - ❖ Kaum mithilfe relationaler Datenbanksysteme (RDBMS) zu bewerkstelligen

Ein Blick zurück auf RDBMS / Stärken

- ❖ Daten in relationalen Datenbanken (RDBMS) in Tabellen gespeichert
- ❖ Zeile = Datensatz + besitzt die durch die Spalten festgelegten Eigenschaften
- ❖ Datensätze können über Primärschlüssel schnell gefunden werden
- ❖ Verbindung zu anderen Tabellen über Fremdschlüsselbeziehungen

- ❖ **Gut verstandenes Modell**
- ❖ SQL recht einfach zu lernen
- ❖ umgangssprachliche Formulierung auch von (komplexeren) Abfragen

Potenzielle Probleme von RDBMS

- ❖ Organisation in Tabellen bereitet Probleme:
 - ❖ wie speichert man Listen => Verweis auf andere Tabellen => JOIN
 - ❖ wie geht man mit Varianzen in den Daten um (NULLABLE Spalten)
- ❖ Weitere Schwachstellen:
 - ❖ RDBMS gelten als schlecht (horizontal) skalierbar
 - ❖ RDBMS performen schlechter bei hoher Anfragelast,
 - ❖ Probleme vor allem bei Konflikten oder im Transaktionskontext
 - ❖ Problem mit Konsistenz und ACID-Transaktionen bei grossen Systemen

Varianten von NoSQL-Datenbanken

- ❖ **Key/Value-Stores (Analogie Map):**

- ❖ Speichern zu einem Schlüssel einen Wert
- ❖ Wert kann durchaus komplexe Daten speichern
- ❖ Komplexere Datenstrukturen aber kaum sinnvoll abzubilden

- ❖ **Document-Stores (Analogie Liste):**

- ❖ Speichern Einträge als Dokumente, die Objekten ähneln
- ❖ Dokumente müssen keiner festen Struktur folgen
- ❖ Abfragen durch spezielle Abfragesprachen (in der Regel nicht SQL)

Varianten von NoSQL-Datenbanken

- ❖ **Wide-Column-Stores:**

- ❖ Einer Zeile können beliebig viele Spalten zugeordnet werden

- ❖ **Graphen-DB (Analogie Graphen):**

- ❖ Speichern Graphen, also Knoten und Verbindungen dazwischen
- ❖ In RDBMS nur durch aufwendige und viele JOINS ermittelbar
- ❖ Ideal um Netzwerke und Beziehungen zu modellieren
- ❖ Abfragen durch spezielle Abfragesprachen

Grundlagen Document Stores

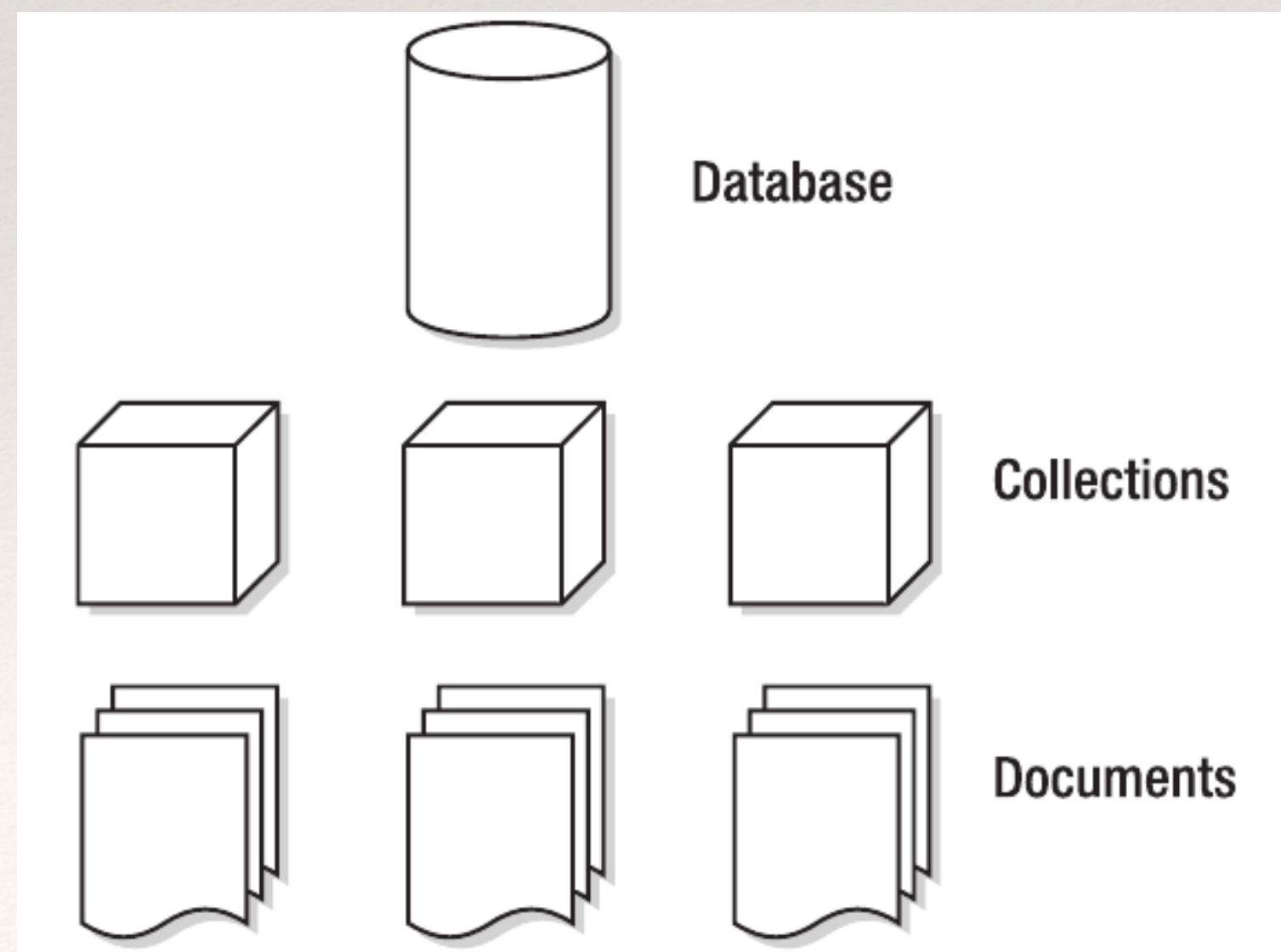
- ❖ Analogie RDBMS
- ❖ Schemafreiheit
- ❖ Gedanken zum ORM



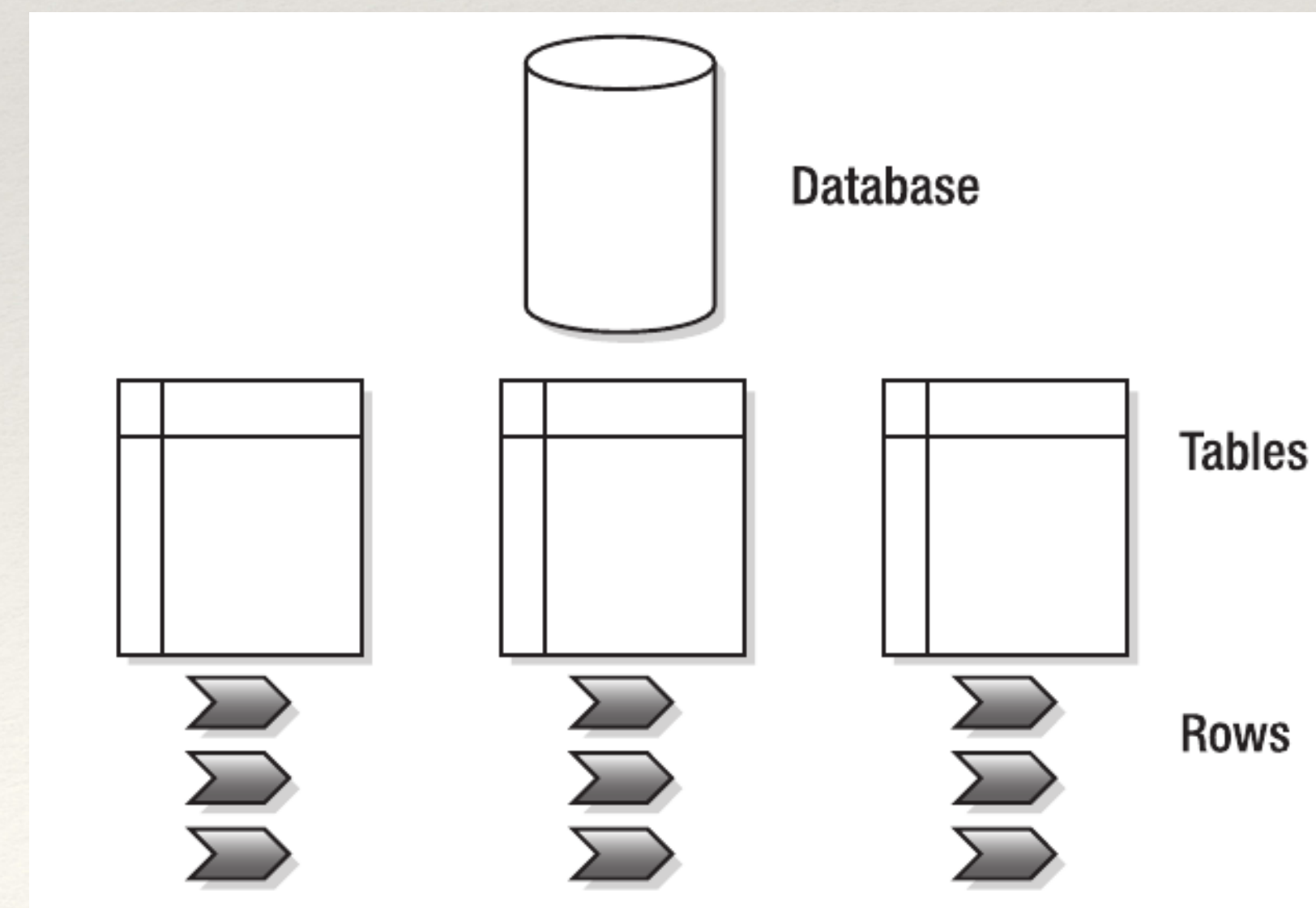
Grundlagen Document Stores – Analogie RDBMS

- | | | |
|-----------------------------------|-----|---------------------|
| ❖ Datenspeicherung in Collections | vs. | in Tabellen |
| ❖ Daten als Dokument / Objekt | vs. | Zeile einer Tabelle |
| ❖ Dokumente besitzen Attribute | vs. | Spalten einer Zeile |

Dokument Store



RDBMS



Grundlagen Document Stores

- ❖ Document-Stores sind schemafrei, das bedeutet:
 - ❖ Daten müssen keine einheitliche Struktur besitzen
 - ❖ Objekte können aus verschiedenen Attributen bestehen + hierarchisch sein

```
{  
  "name": "Inden",  
  "vorname": "Michael",  
  "adresse": { "ort" : "Zürich",  
               "strasse" : "Letzigraben"  
            },  
  "programmiersprachen": [ "Java", "Groovy", "C++" ]  
}
```

Grundlagen Document Stores

- ❖ Notation häufig JSON
- ❖ Typen und Attribute können sich von Dokument zu Dokument unterscheiden
- ❖ Attribute können mehrere Werte enthalten, z.B. Liste von Hobbies

```
{  
  "name": "Mayer",  
  "vorname": "Peter",  
  "hobbies": [ "Java", "Karate", „Inline-Skating" ]  
}
```

Grundlagen Document Stores – ORM

- ❖ Mapping Objekt <-> Datenbank einfach
- ❖ Kein Impedance Mismatch (wie bilde ich Objekte auf Tabellen ab)
- ❖ Abhilfe für Herausforderungen beim Einsatz von RDBMS
 - ❖ Optionale Attribute
 - ❖ Speicherung von Collections (Arrays, Listen, Sets, ...)
 - ❖ Vererbung

Grundlagen Document Stores

- ❖ Schemafrei impliziert auch:
 - ❖ **Aufbau der Dokumente muss vor der Speicherung und beim Einrichten der Datenbank nicht bekannt sein**
 - ❖ und trotzdem können mächtige Abfragen ausgeführt werden
- ❖ Schema-Gebundenheit bei RDBMS
 - ❖ Gibt Aufbau und möglichen Inhalt vor und erlaubt so Integritäts- und Konsistenzprüfungen durch die Datenbank
 - ❖ Daten werden in der Regel zur Vermeidung von Redundanz auf diverse Tabellen verteilt und müssen später über JOINS wieder zusammengestellt werden

Eigenschaften von MongoDB

- ❖ Allgemeines
- ❖ Unterschiede zu RDBMS
- ❖ Gemeinsamkeiten mit RDBMS
- ❖ Installation



Eigenschaften MongoDB

- ❖ Dokument-orientiertes Datenmodell
- ❖ BSON-Format ist ein Binärformat für JSON
- ❖ Eigene Query-Language auf JavaScript basierend
- ❖ Teilweise etwas kryptisch, vor allem komplexere Ausdrücke
- ❖ Treiber für verschiedene Programmiersprachen
<http://docs.mongodb.org/ecosystem/drivers/>

Unterschiede RDBMS und MongoDB

- ❖ RDBMS: Tabellen, Datensätze, Spalten
- ❖ MongoDB: Collections, Dokumente / Objekte, Attribute
- ❖ RDBMS: Tabellenstruktur (Spalten) durch Schema vorgegeben
- ❖ MongoDB: Schema-frei: Speicherung von Objekten mit unterschiedlichem Aufbau möglich
- ❖ RDBMS: ACID-Transaktionen
- ❖ MongoDB: Nur einzelne Schreiboperationen sind atomar

Gemeinsamkeiten RDBMS und MongoDB

- ❖ Jeder Datensatz bzw. jedes Objekte in MongoDB besitzt eine eindeutige ID
- ❖ ID ist Primärschlüssel
- ❖ MongoDB baut automatisch einen Index über die ID auf
- ❖ Es können weitere Indizes basierend auf beliebigen Attributen möglich
- ❖ Abfragen können durch Indizes (extrem) beschleunigt werden

MongoDB-Installation

- ❖ Die Installation gestaltet sich extrem einfach
- ❖ Download von der Seite <http://www.mongodb.org/>
- ❖ Dort findet man sowohl die Datenbank als Applikation (`mongod`)
- ❖ als auch einen Kommandozeilen-Client (`mongo`)

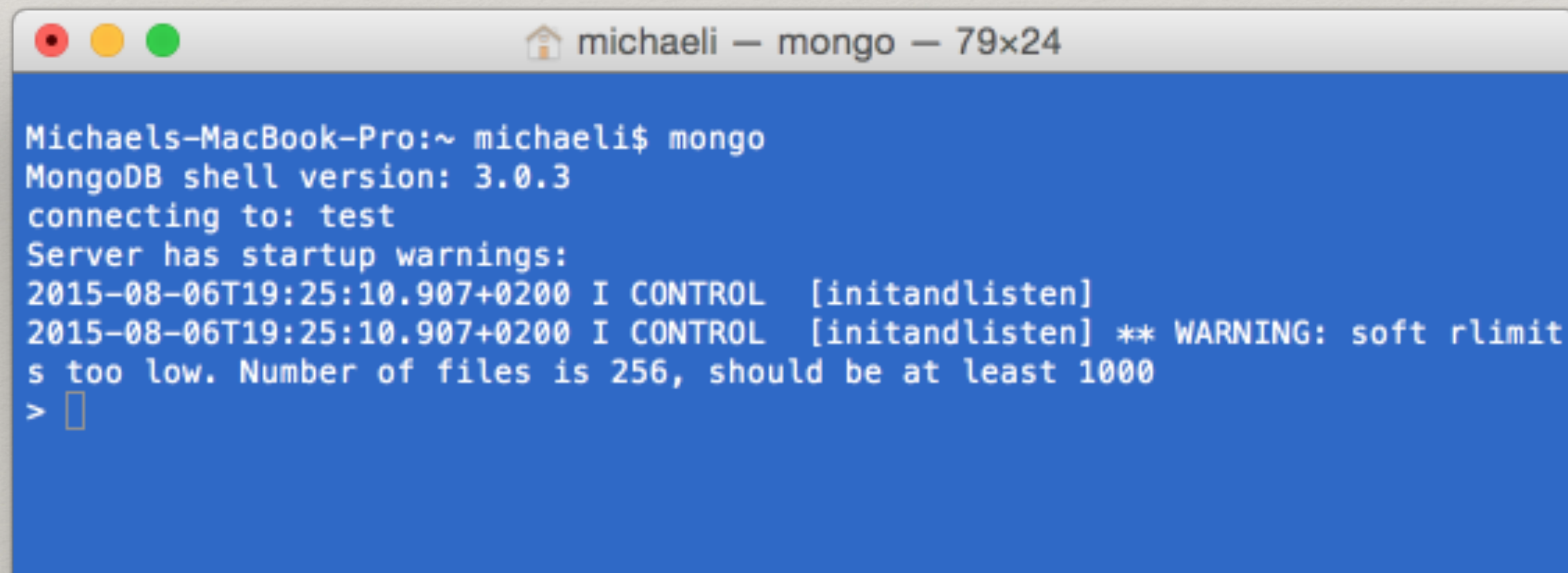
MongoDB in Action

- ❖ MongoClient
- ❖ First Steps
- ❖ Kommandos
- ❖ Gemeinsamkeiten mit RDBMS



MongoDB in Action – MongoClient

- ❖ Mongo-Server per `mongod -dbpath data` starten und FERTIG ;-)
- ❖ MongoDB liefert eine Kommando-Shell mit
 - ❖ Ist ein JavaScript Interpreter
 - ❖ DB-Abfragen über die Konsole

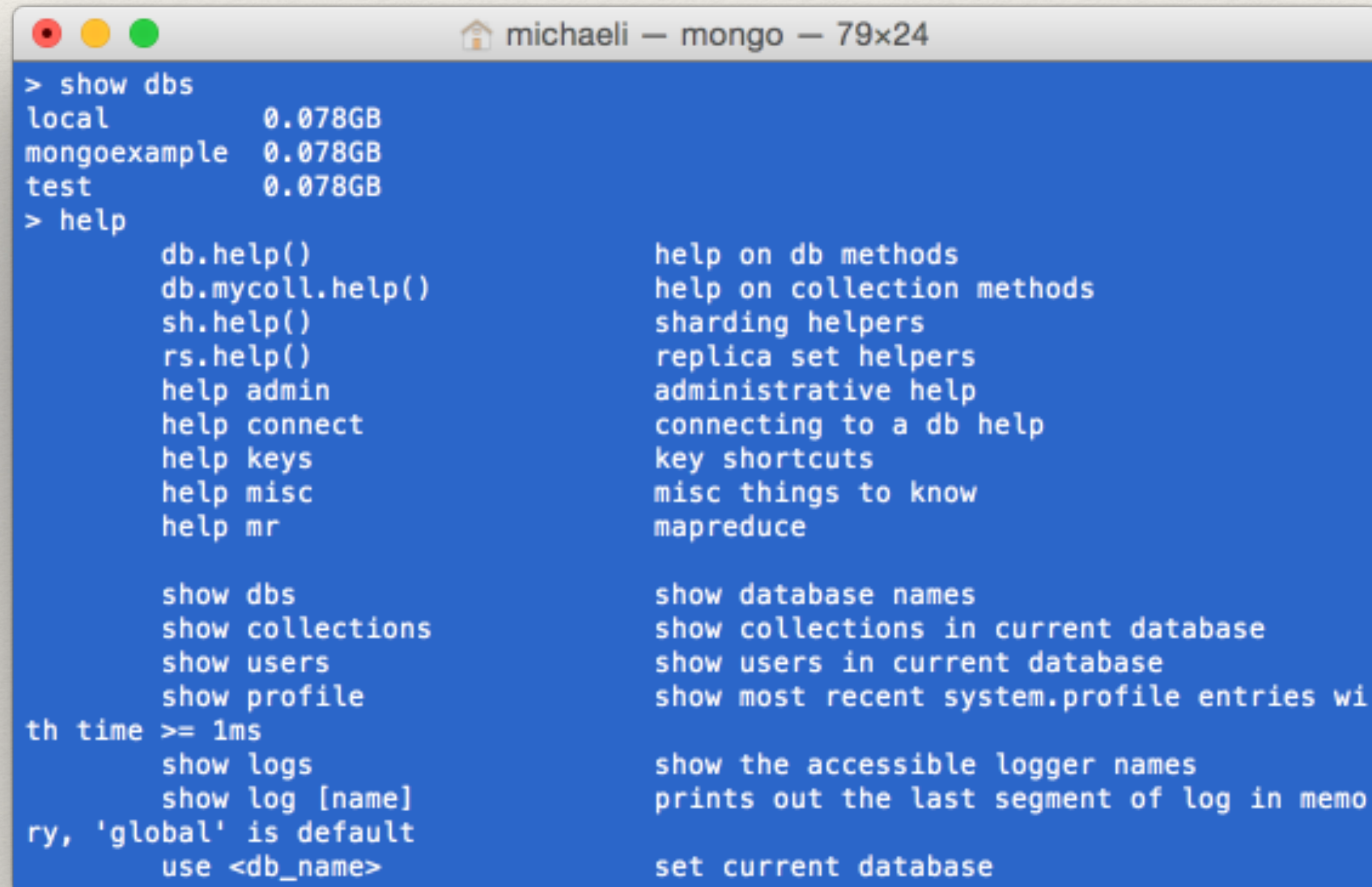


```
michaeli — mongo — 79x24

Michaels-MacBook-Pro:~ michaeli$ mongo
MongoDB shell version: 3.0.3
connecting to: test
Server has startup warnings:
2015-08-06T19:25:10.907+0200 I CONTROL [initandlisten]
2015-08-06T19:25:10.907+0200 I CONTROL [initandlisten] ** WARNING: soft rlimit
s too low. Number of files is 256, should be at least 1000
> 
```


MongoDB in Action – First Steps

- ❖ Anzeige aller vorhandenen Datenbanken: `show dbs`
- ❖ Aber: Woher weiss ich, welche Kommandos alle möglich sind? `help`



```
michaeli — mongo — 79x24
> show dbs
local          0.078GB
mongoexample   0.078GB
test           0.078GB
> help
db.help()                help on db methods
db.mycoll.help()          help on collection methods
sh.help()                 sharding helpers
rs.help()                 replica set helpers
help admin                administrative help
help connect              connecting to a db help
help keys                 key shortcuts
help misc                 misc things to know
help mr                   mapreduce

show dbs                  show database names
show collections           show collections in current database
show users                show users in current database
show profile              show most recent system.profile entries with time >= 1ms
show logs                 show the accessible logger names
show log [name]           prints out the last segment of log in memory, 'global' is default
use <db_name>             set current database
```

MongoDB in Action – First Steps

❖ Nützliche Kommandos

- ❖ `help` => Anzeige einer Hilfeseite
 - ❖ `db.help()` => Hilfe zu Datenbankzugriffen
 - ❖ `db.<mycoll>.help()` => Hilfe zu Methoden auf Collections
 - ❖ `show dbs` => Liste der verfügbaren Datenbanken
 - ❖ `use <dbname>` => Aktivieren der Datenbank mit dem Name <dbname>
 - ❖ `show collections` => Collections der aktuellen DB
-
- ❖ Benutzen und (implizites) Anlegen einer Datenbank: `use bu-talk`
 - ❖ Welche Datenbank nutzen wir gerade? `db`

MongoDB in Action – MongoDB Commands

- ❖ **CRUD** (RDBMS) a.k.a. **IFUR** (MongoDB)

- ❖ **CREATE** => **INSERT**

- ❖ **READ** => **FIND**

- ❖ **UPDATE** => **UPDATE**

- ❖ **DELETE** => **REMOVE**

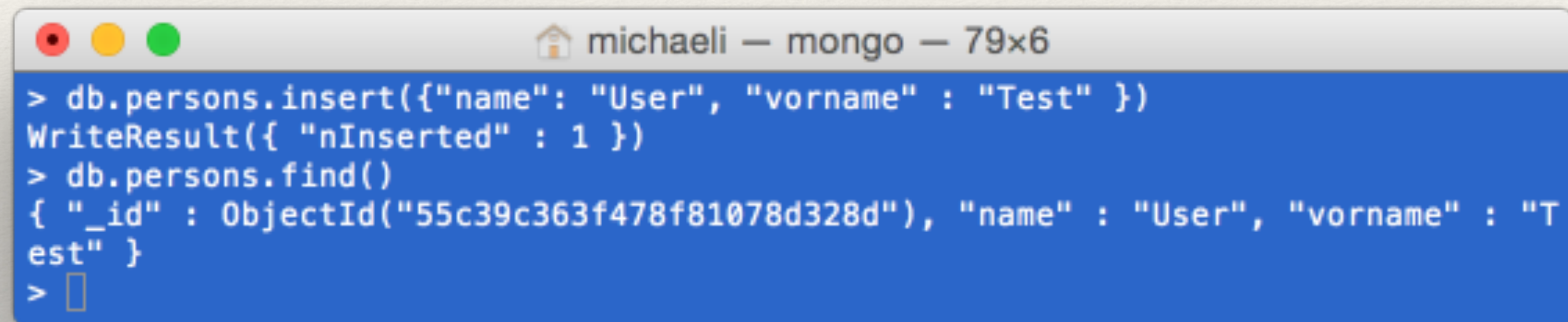
- ❖ **INSERT:**

Benutzen und (implizites) Anlegen einer Collection mit einem JSON-Objekt:

```
db.persons.insert( {"name": "User", "vorname" : "Test" } )
```

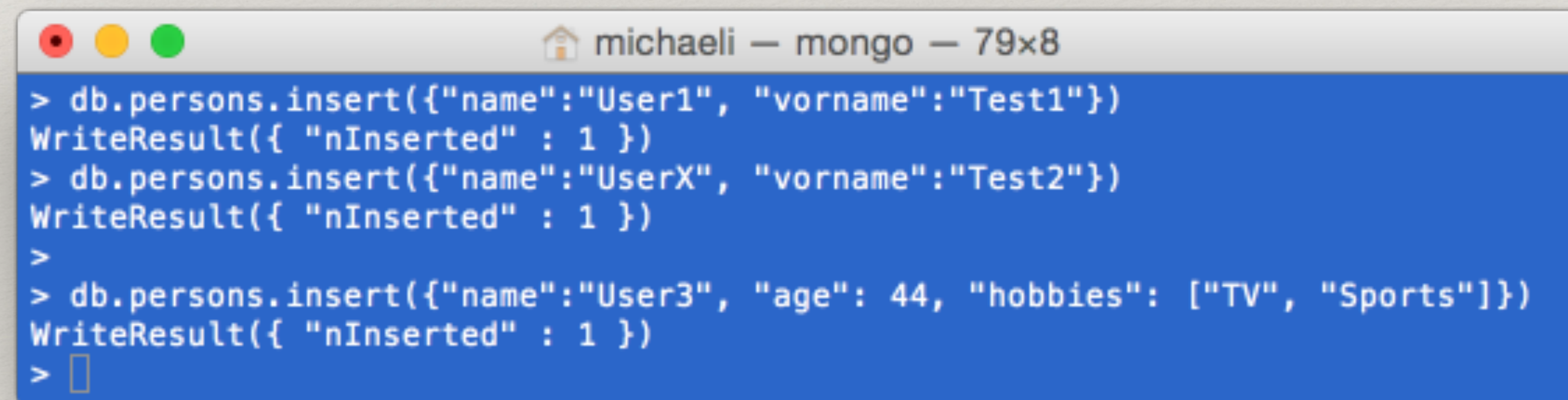

MongoDB in Action – IFUR-Commands

- ❖ **FIND:** Welche Daten sind gespeichert? `db.persons.find()`



```
michaeli — mongo — 79x6
> db.persons.insert({"name": "User", "vorname" : "Test" })
WriteResult({ "nInserted" : 1 })
> db.persons.find()
{ "_id" : ObjectId("55c39c363f478f81078d328d"), "name" : "User", "vorname" : "Test" }
>
```

- ❖ **INSERT:**

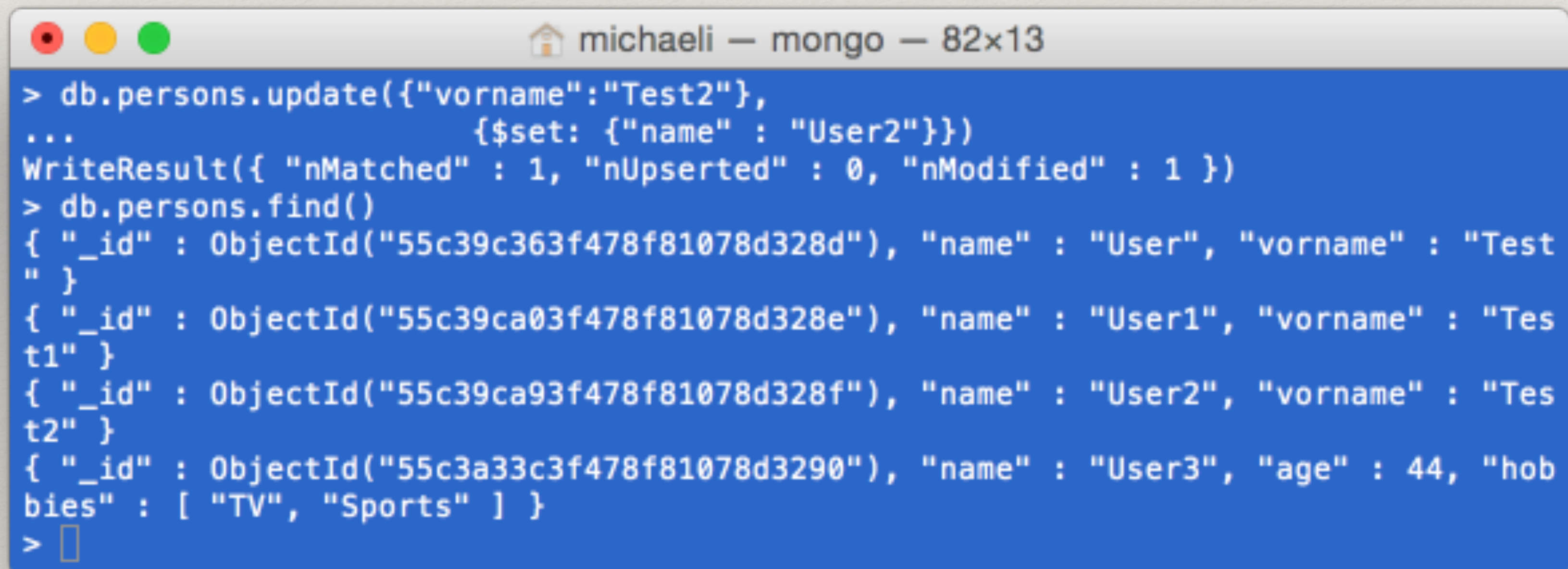


```
michaeli — mongo — 79x8
> db.persons.insert({"name":"User1", "vorname":"Test1"})
WriteResult({ "nInserted" : 1 })
> db.persons.insert({"name":"UserX", "vorname":"Test2"})
WriteResult({ "nInserted" : 1 })
>
> db.persons.insert({"name":"User3", "age": 44, "hobbies": ["TV", "Sports"]})
WriteResult({ "nInserted" : 1 })
>
```


MongoDB in Action – IFUR-Commands

- ❖ **UPDATE:** Ändern des Tippfehlers im Nachnamen UserX:

```
db.persons.update( {"vorname": "Test2"},  
                  { $set: { "name" : "User2" } } )
```



```
michaeli — mongo — 82x13  
> db.persons.update({"vorname": "Test2"},  
...                  { $set: { "name" : "User2" } })  
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })  
> db.persons.find()  
{ "_id" : ObjectId("55c39c363f478f81078d328d"), "name" : "User", "vorname" : "Test  
" }  
{ "_id" : ObjectId("55c39ca03f478f81078d328e"), "name" : "User1", "vorname" : "Tes  
t1" }  
{ "_id" : ObjectId("55c39ca93f478f81078d328f"), "name" : "User2", "vorname" : "Tes  
t2" }  
{ "_id" : ObjectId("55c3a33c3f478f81078d3290"), "name" : "User3", "age" : 44, "hob  
bies" : [ "TV", "Sports" ] }  
> 
```


MongoDB in Action – IFUR-Commands

- ❖ **REMOVE:** Löschen von Dokumenten

```
db.persons.remove( {"name" : "User2"} )
```

- ❖ **FIND:** Spezifikation von Suchbedingungen

```
db.persons.find( {"age": { $gt : 25 } } )
```



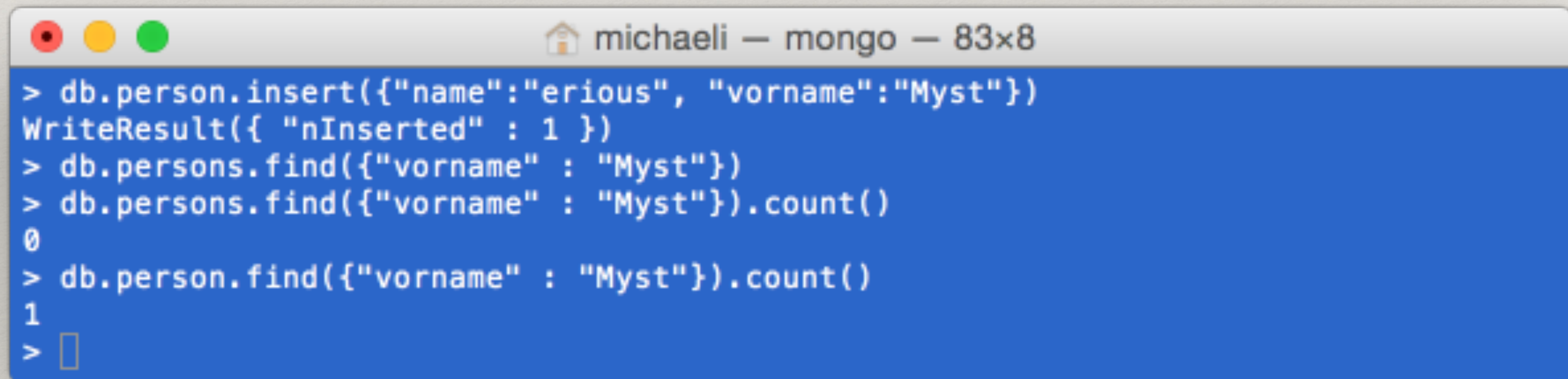
```
michaeli — mongo — 90x6
> db.persons.remove({"name" : "User2"})
WriteResult({ "nRemoved" : 1 })
> db.persons.find({"age": { $gt : 25 } })
{ "_id" : ObjectId("55c3a33c3f478f81078d3290"), "name" : "User3", "age" : 44, "hobbies" :
[ "TV", "Sports" ] }
> 
```


MongoDB in Action – Achtung Fehlertoleranz

- ❖ Nehmen wir an, wir hätten noch ein Objekt wie folgt hinzugefügt:

```
db.person.insert( {"name": "erious", "vorname": "Myst" } )
```

- ❖ **FIND:** Welche Daten sind gespeichert? `db.persons.find()` liefert das neu erzeugte Objekt nicht zurück? Wo ist es?



```
michaeli — mongo — 83x8
> db.person.insert({"name": "erious", "vorname": "Myst"})
WriteResult({ "nInserted" : 1 })
> db.persons.find({"vorname" : "Myst"})
> db.persons.find({"vorname" : "Myst"}).count()
0
> db.person.find({"vorname" : "Myst"}).count()
1
>
```

MongoDB in Action – IFUR-Commands

❖ **Insert:** Verschachtelte Dokumente

```
db.persons.insert({ "name" : "Tim",  
                  "address": { "city" : "Kiel", "PLZ": 24116 } })  
db.persons.insert({ "name" : "Mike",  
                  "address": { "city" : "Kiel", "PLZ": 24106 } })  
db.persons.insert({ "name" : "Mike",  
                  "address": { "city" : "Aachen", "PLZ": 52070 } })
```

❖ **Wie suche ich nach Ort oder Ort und PLZ?**

MongoDB in Action – IFUR-Commands

- ❖ **FIND:** Alle aus Kiel

```
db.persons.find( {"address.city": "Kiel" } )
```

- ❖ **FIND:** Alle aus Kiel mit PLZ 24106

```
db.persons.find( {"address":  
                  { "city" : "Kiel", "PLZ": 24116 } })
```

MongoDB in Action – IFUR-Commands

❖ **FIND: AND**

```
SELECT * FROM inventory WHERE status = "A" AND qty < 30
```

=>

```
db.inventory.find( { status: "A", qty: { $lt: 30 } } )
```

MongoDB in Action – IFUR-Commands

❖ **FIND:** AND und OR

```
SELECT * FROM inventory WHERE status = "A" AND  
                                (qty < 30 OR item LIKE "p%")
```

=>

```
db.inventory.find( { status: "A",  
                    $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ] } )
```

Weiterführende Funktionalität

- ❖ Spezialfunktionalitäten
- ❖ Ausfallsicherheit
- ❖ Skalierung durch Shards



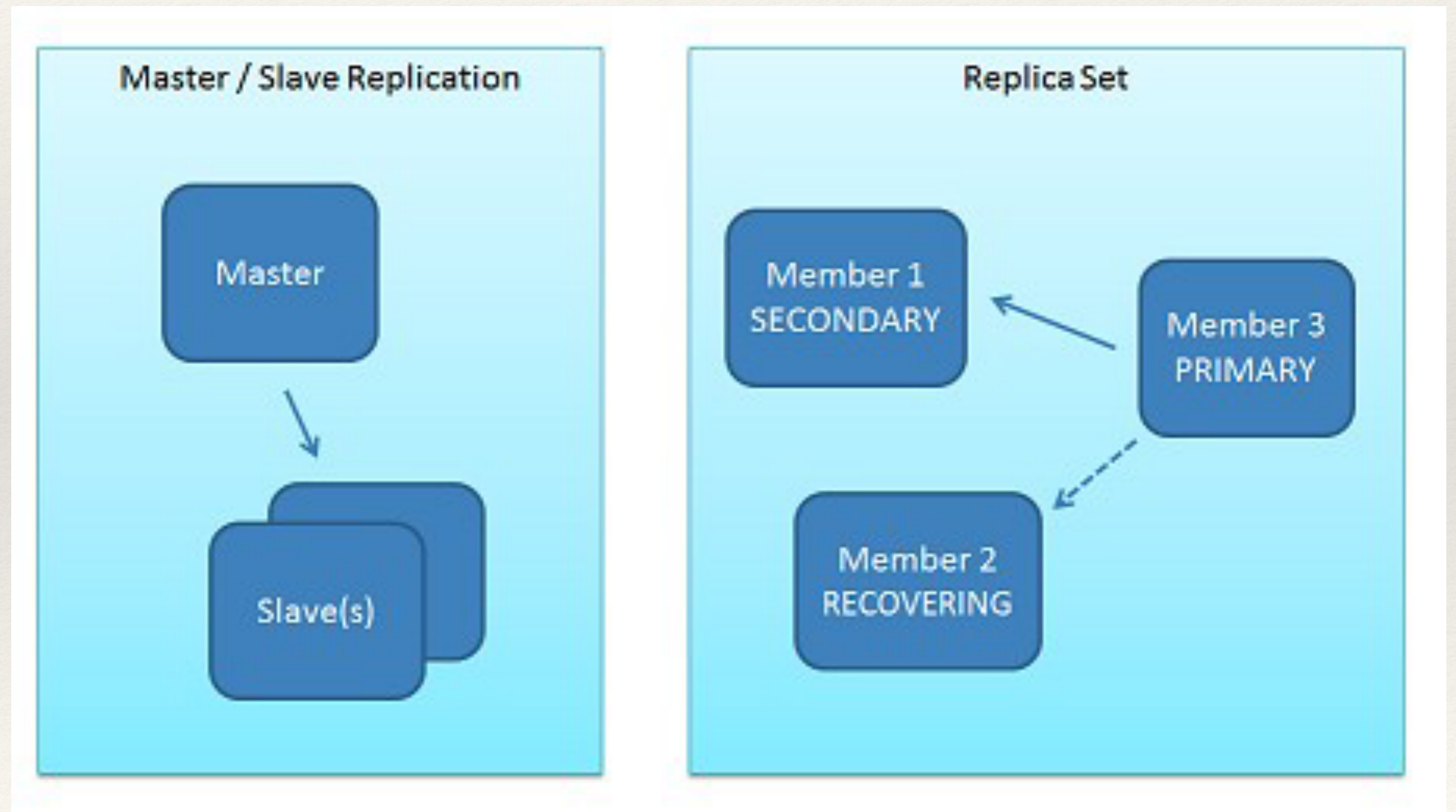
Spezialfunktionen: Map Reduce / GridFS

- ❖ **MapReduce-Framework** – Flexibles Aggregationsframework, map und reduce werden als JavaScript-Funktionen angegeben
 - ❖ Spezifische Aggregationen lassen sich nicht mit «normalen» Bordmitteln ausführen, etwa Anfragen, die GROUP BY in SQL nutzen würden.
- ❖ **GridFS** – eigenes Dateisystem zum Speichern auch sehr grosser Daten
 - ❖ Dokumente können maximal 16 MB gross werden

Ausfallsicherheit

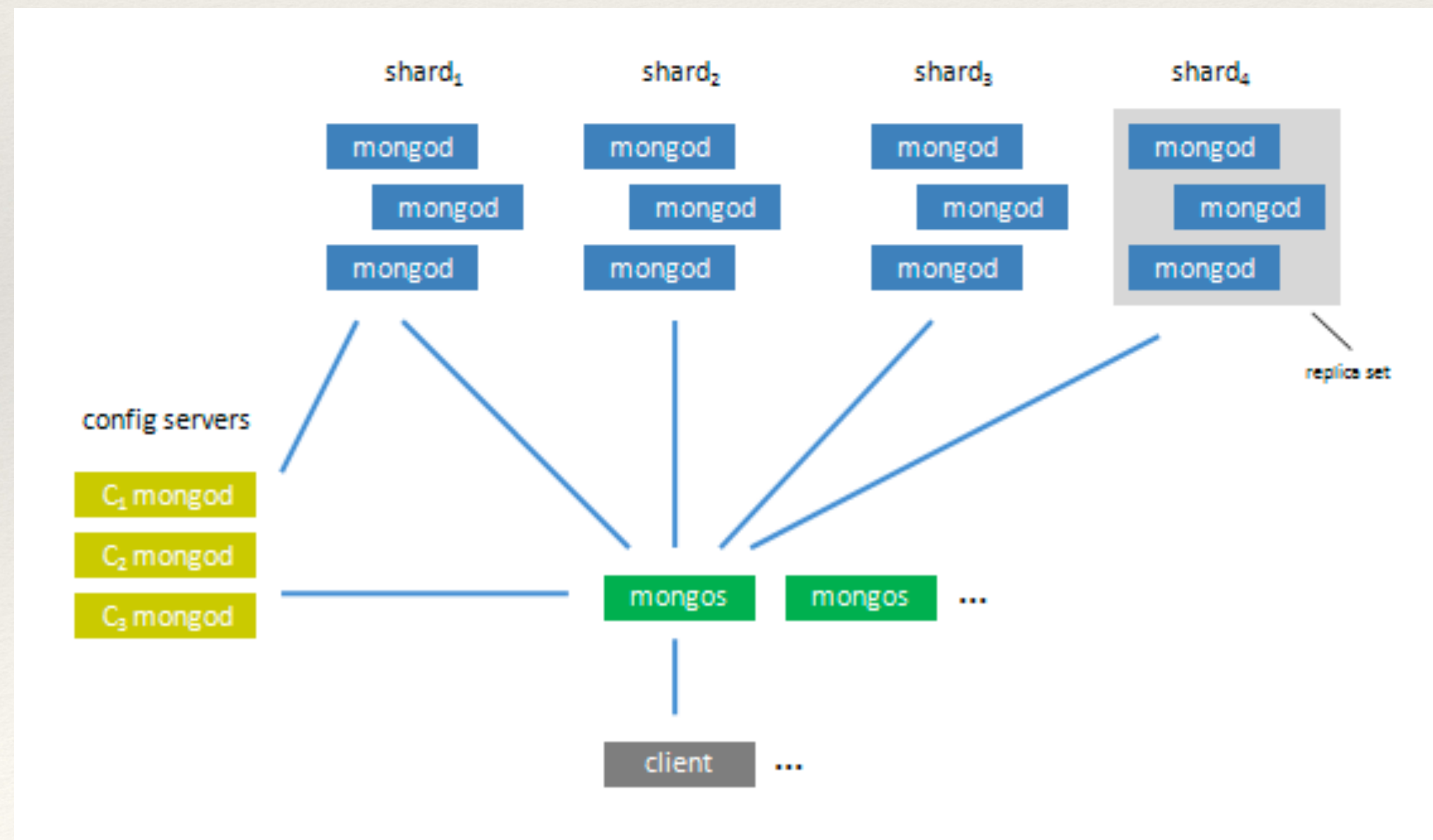
❖ MongoDB stellt zwei Varianten bereit:

- ❖ Alle Schreiboperationen gehen an den Master/Primary
- ❖ Die Secondaries werden asynchron basierend auf den Daten des Primary aktualisiert (repliziert)
- ❖ Lesezugriffe werden auf die Slaves/Secondaries verteilt



Skalierung durch Shards

- ❖ Sharding ist der Skalierungsansatz von MongoDB
- ❖ Dabei wird eine Collection unterteilt und auf verschiedene Rechner verteilt.



Zusammenfassung und Fazit



Zusammenfassung

- ❖ MongoDB bietet ...
 - ❖ ... eine einfache Installation
 - ❖ ... eine gute Dokumentation
 - ❖ ... viele Treiber für gängige Sprachen
 - ❖ ... eine JavaScript-basierte Abfragesprache
 - ❖ ... Ausfallsicherheit durch Replica Sets
 - ❖ ... Skalierbarkeit durch Sharing
 - ❖ ... einen leichten Einstieg mit der Mongo-Console

NoSQL – Wie populär und häufig genutzt sind NoSQL-DBs denn nun wirklich?


- ❖ August 2013: Nur MongoDB in den Top 10 der DBs, Rest (noch) RDBMS

Ranking > Vollständiges Ranking RSS RSS Feed

DB-Engines Ranking

Das DB-Engines Ranking ist eine Rangliste der Popularität von Datenbankmanagementsystemen. Die Rangliste wird monatlich aktualisiert.

Lesen Sie mehr über die [Berechnungsmethode](#) der Bewertungen.



189 Systeme im Ranking, August 2013

Rang	Vormonat	DBMS	Datenbankmodell	Punkte	Änderung
1.		Oracle	Relational DBMS	1544,44	+16,80
2.	↑	MySQL	Relational DBMS	1324,83	+19,59
3.	↓	Microsoft SQL Server	Relational DBMS	1304,96	-17,02
4.		PostgreSQL	Relational DBMS	182,22	-0,16
5.		DB2	Relational DBMS	162,94	-5,49
6.	↑	MongoDB	Document Store	155,99	+15,53
7.	↓	Microsoft Access	Relational DBMS	150,88	+9,24
8.		Sybase	Relational DBMS	81,59	-4,61
9.		SQLite	Relational DBMS	79,44	-3,36
10.		Teradata	Relational DBMS	53,83	-2,51

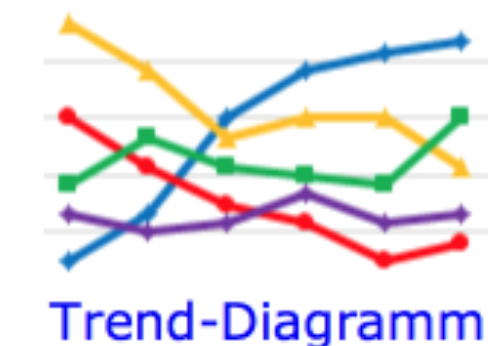
NoSQL – Wie populär und häufig genutzt sind NoSQL-DBs denn nun wirklich?

- ❖ August 2015: Schon 3 NoSQL-DBs in den Top 10 der DBs, Top 3 (noch) RDBMS

DB-Engines Ranking

Das DB-Engines Ranking ist eine Rangliste der Popularität von Datenbankmanagementsystemen. Die Rangliste wird monatlich aktualisiert.

Lesen Sie mehr über die [Berechnungsmethode](#) der Bewertungen.



280 Systeme im Ranking, August 2015

Rang			DBMS	Datenbankmodell	Punkte		
Aug 2015	Jul 2015	Aug 2014			Aug 2015	Jul 2015	Aug 2014
1.	1.	1.	Oracle	Relational DBMS	1453,02	-3,70	-17,83
2.	2.	2.	MySQL	Relational DBMS	1292,03	+8,69	+10,81
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1108,66	+5,60	-133,84
4.	4.	↑ 5.	MongoDB +	Document Store	294,65	+7,26	+57,30
5.	5.	↓ 4.	PostgreSQL	Relational DBMS	281,86	+9,04	+32,01
6.	6.	6.	DB2	Relational DBMS	201,23	+3,12	-5,19
7.	7.	7.	Microsoft Access	Relational DBMS	144,20	-0,10	+4,58
8.	8.	↑ 10.	Cassandra +	Wide Column Store	113,99	+1,28	+32,09
9.	9.	↓ 8.	SQLite	Relational DBMS	105,82	-0,05	+16,95
10.	10.	↑ 11.	Redis +	Key-Value Store	98,81	+3,73	+28,01

Fazit

- ❖ NoSQL-DBs sind kein allgemeiner Ersatz für RDBMS, sondern eher eine gute Ergänzung im Werkzeugkasten
- ❖ Je nach Einsatzzweck sollte man zwischen RDBMS bzw. NoSQL-DB mit Bedacht wählen
- ❖ Für Verarbeitung grosser Datenmengen und vieler Schreibzugriffe ohne Bedarf nach Transaktionen scheint MongoDB ideal
- ❖ Typische Anwendungsfälle: Logging, Auditing, sonstige Protokollierung