Nicht-relationale Datenbanken in Web-Anwendungen

Relationale Datenbanken

- Komplexe Abfragen per SQL möglich
- Effizient bei kleinen Transaktionen
- Benötigen feste Tabellenschemata
- Schwierig zu skalieren

Nicht-relationale Datenbanken (NoSQL)

- Keine festgelegten Tabellenschemata
- Horizontal skalierbar
- Optimiert für viele Schreib-/Leseanfragen
- Viele verschiedene Datenbanken mit unterschiedlichem Focus
- Meist schwache Garantie der Konsistenz (verzicht auf ACID-Eigenschaften)

Typen von NoSQL Datenbanken

Key-Value Stores

Dokumentenorientierte Datenbanken

Spaltenorientierte Datenbank

In-memory key-value store

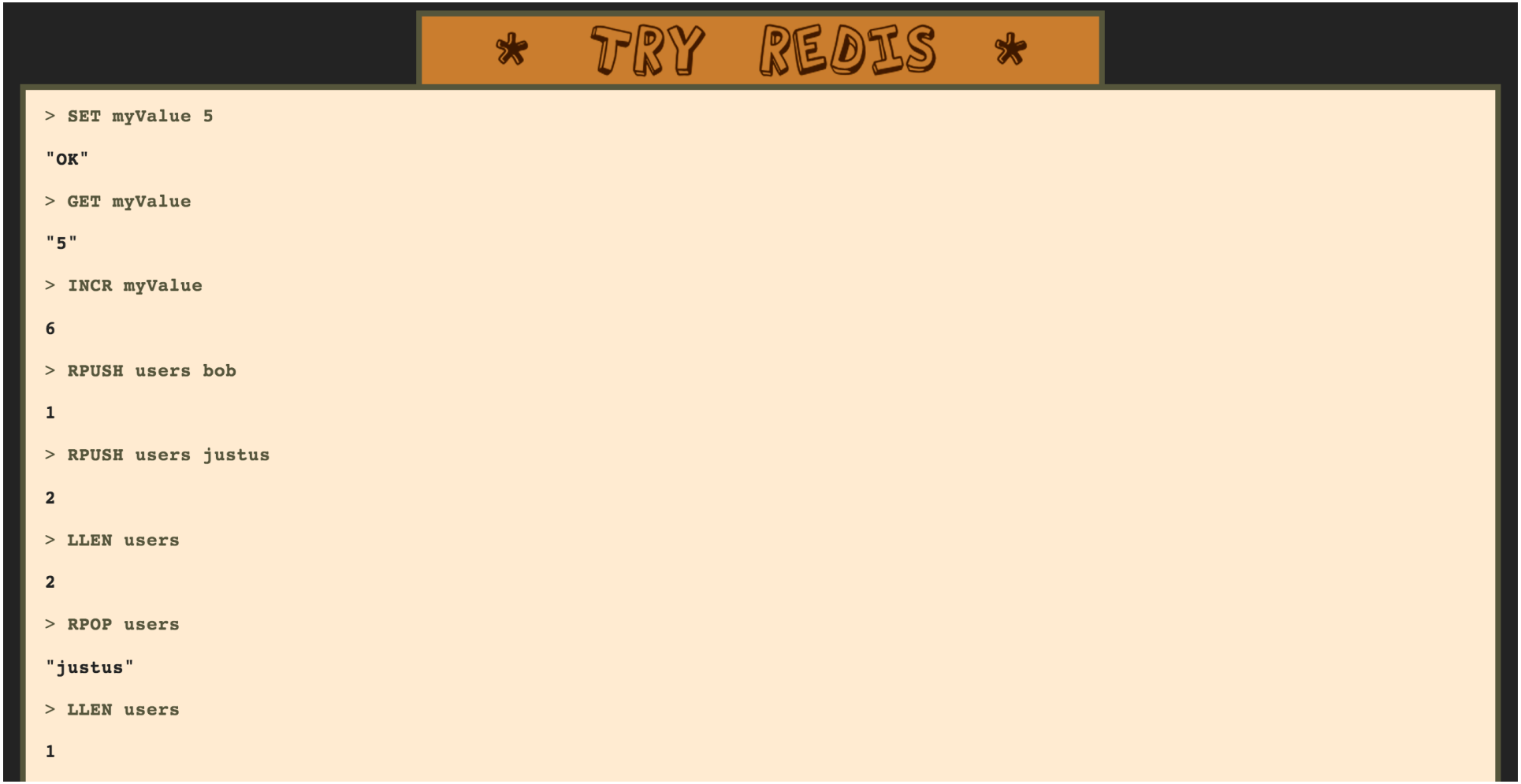
Sprache: C/C++

Protokoll: Telnet-ähnlich

Besonderheiten: Kennt Transaktionen, Daten können expiren, Pub/Sub Unterstützung

Anwendungsbeispiel: Häufig geänderte Daten mit absehbarer Größe – z.B. Börsenkurse, Logdaten, Messaging. Das bessere Memcache.

Ausprobieren auf: http://try.redis.io



Indexed document store

Sprache: C++

Protokoll: Eigenes, BSON encoded

Besonderheiten: Dynamische Queries, kann große BLOBS speichern, serverseitige Javascript Funktionen, sharding und Master/Slave Replikation

Anwendungsbeispiel: Schnelle Zugriffe auf große Datenmengen. Ersatz für MySQL in vielen Web-Szenarien.

Ausprobieren: http://www.mongodb.org -> Try it out

A Tiny MongoDB **Browser Shell**

Just enough to scratch the surface (mini tutorial included)

X CLOSE

```
> db.users.save({name: "Rick", email: "pannen@gmail.com"})
"ok"
> db.users.find({name: "Rick"})

[
  {   "name" : "Rick",   "_id" : {   "$oid" : "50f4a6abcc93742c160019b2"   },   "email" : "pannen@gmail.com"   }
]
> db.users.update({name: "Rick"}, {$set: {city: "Hamburg"}})
"ok"
> db.users.find({name: "Rick"})

[
  {   "name" : "Rick",   "city" : "Hamburg",   "_id" : {   "$oid" : "50f4a6abcc93742c160019b2"   },   "email" :
"pannen@gmail.com"   }
]
>
```

Map/reduce document store

Sprache: Erlang

Protokoll: HTTP / REST

Besonderheiten: Abfragen per Map/Reduce, Datei-Attachments an Dokumenten, behält Revisionen, Bi-direktionale Replikation!

Anwendungsbeispiel: Datenbestand, der anwächst, aber nicht oft geändert wird. Multi Site deployments.

Map:

```
function(doc) {
  if (doc.Type == "customer") {
    emit(doc.LastName, {FirstName: doc.FirstName, Address: doc.Address});
  }
}
```

Map/Reduce:

```
function map(doc) {
  if (doc.Type == ”rechnung") {
    emit(doc.Date, {sum: doc.Betrag});
  }
}
function reduce(key, values, rereduce) {
  return sum(values);
}
```

Column based database

Sprache: Java

Protokoll: Thrift

Besonderheiten: Schreiben schneller als Lesen, Big Table Features, Daten können expiren, Map/Reduce mit Hadoop, Alle Nodes sind gleich

Anwendungsbeispiel: Logging, Datananlyse, Banken, Versicherungen, Web-Analytics.


```
[default@DEMO] set Users[1234][name] = rick;
Value inserted.
Elapsed time: 10 msec(s).
[default@DEMO] set Users[1234][password] = geheim;
Value inserted.
Elapsed time: 10 msec(s).
[default@DEMO] get Users[1234];
=> (column=name, value=rick, timestamp=13507691616840)
=> (column=password, value=geheim, timestamp=13507692451910)

Returned 2 results.
Elapsed time: 67 msec(s).
```

MongoDB & Rails 3

- Rails-Projekt ohne ActiveRecord initialisieren

```
rails new MyProject --skip-active-record
```

- Mongo Gem installieren

MongoMapper -> http://mongomapper.com

oder Mongoid -> http://mongoid.org

Datamodel mit Mongoid

```
class Person
  include Mongoid::Document
  field :first_name, type: String
  field :middle_name, type: String
  field :last_name, type: String
end
```

Relationen (embedded)

```
class Band
  include Mongoid::Document
  embeds_many :albums
end

class Album
  include Mongoid::Document
  embedded_in :band
end
```

Relationen (referenced)

```
class Band
  include Mongoid::Document
  has_many :albums
end

class Album
  include Mongoid::Document
  belongs_to :band
end
```

Indizes

Definition:

```
class Person
  include Mongoid::Document
  field :age, :index => true
  field :location, type: Array

  index({ location: "2d" }, { min: -200, max: 200 })
end
```

Erzeugen mit:

```
> rake db:mongoid:create_indexes
```

Queries

```
Band.find("4baa56f1230048567300485c")

Band.where(name: "Depeche Mode")

Band.
  where(:founded.gte => "1980-1-1").
  in(name: [ "Tool", "Deftones" ]).
  sort(:founded => 1).
  limit(10)
```