Skalierbare Datenbanksysteme (NoSQL)

Document Stores (CouchDB)

Von Marcel Sinn 6IB
SEM, HS Mannheim 2012

Inhalt

- Geschichte CouchDB
- Was ist ein Document Store ?
- Warum CouchDB ?
- Die API
- Funktionen
- Replikation / Clustering
- Einsatzgebiete
- Live Demo

Geschichte

CouchDB: „**C**luster **o**f **u**nreliable **c**ommodity **h**ardware **D**ata **B**ase“
(Zu deutsch: „Datenbank auf einem Cluster aus unzuverlässiger Standardhardware“.)

- Entwicklung seit 2005 von Damien Katz

-> Ziel: Dokumentenorientierte DB mit MapReduce-Ansatz

- Seit 2007 eigenen Port 5984 von IANA
- Ab 2008 im „Brutkasten“-Status der Apache Foundation
- Nov. 2008 Aufwertung zu Apache Projekt
- Version 1.0 am 14. Juli 2010 (News)
- Aktuell Version 1.2.0 seit 10. April 2012 (News)

Was ist ein Document Store ?

Document Store: dokumentenorientierte DB

- NoSQL (not only) Bewegung, aber älter
- Dokumente bilden Grundeinheit
  (SQL: Tabellen mit festen Schema)
- Dokumente können sein:
  - Strukturierte Daten: Textdateien (z.B.: .txt)
  - Binary Large Object: Videofilm (z.B.: .mpeg)

Was ist ein Document Store ?

- Identifizierung: Eindeutige ID
  - acef9ec9ffe29733de784473c70000c7

- Inhalt: JSON-artige Key-Value Struktur

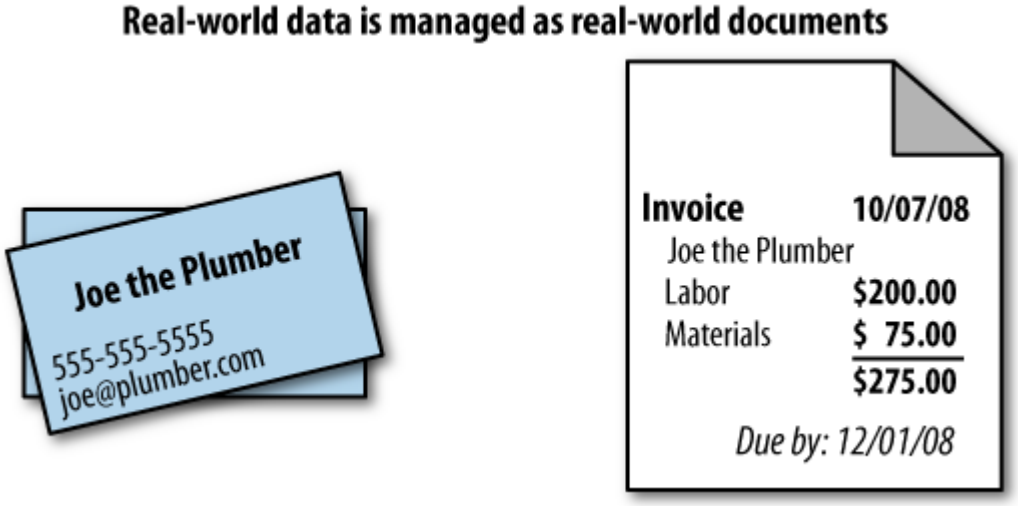

```
{
   "_id": "acef9ec9ffe29733de784473c70000c7",
   "_rev": "5-bd5f698adcea2ab57aaf66af5c13d86b",
   "_attachments": {
       "semtest2.txt": {
           "content_type": "text/plain",
           "revpos": 3,
           "digest": "md5-agwav1jn5moVkO5mWuHL6A==",
           "length": 5,
           "stub": true
       }
   }
}
```

Warum CouchDB ?

- „**Entspann dich**“
  - Leichtes verstehen der Grundkonzepte
  - Produktivbetrieb: Automatisierung, fehlertolerante Architektur (Fehler bleiben auf Request beschränkt)
- „Nicht im Weg zu stehen, wenn kreative Menschen versuchen ein Problem zu lösen“
  - Leichte Bedingung durch REST basierte API
- Gute, einfach gehaltene Dokumentation
- Ausgelegt auf wechselnde Last

Warum CouchDB ?

- „In sich abgeschlossene Daten“:



Rechnung enthält alle nötige Informationen.

(Jeder schätz es alle Informationen auf einen Blick zu haben)
Keine Abstrakten Referenzen (Verweis auf andere Rechnung)

Warum CouchDB ?

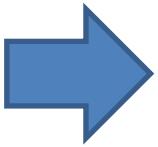
- Vergleich SQL:
  „Jede Rechnung wird als Zeile in einer Tabelle gespeichert.
  Diese Zeile verweist auf andere Zeilen in anderen Tabellen — eine für den Käufer, eine für den Verkäufer, eine für jedes Teil was verkauft wurde und noch mehr Zeilen, die wiederum die verkauften Teile genauer beschreiben.“

-> Manchmal eignen sich Relationen nicht so gut

-> Wichtiges Konzept dokumentbasierender DBs

Warum CouchDB ?

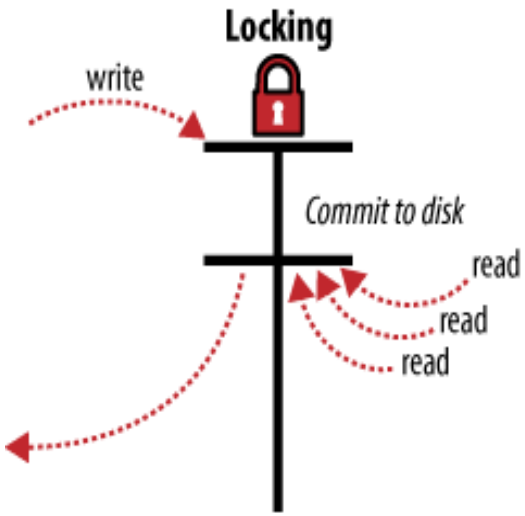
- Syntax und Semantik
  - Beispiel Visitenkarten: Reales Dokument (Menschen erkennen diese als solche)
    Viele Gemeinsamkeiten (Vor-,Nachname), aber manchmal Unterschiede (Fax, kein Fax)
  - > „Kein Fax“ muss nicht extra gespeichert werden, durch Fehlen dieser Information wird klar das kein Fax vorhanden
  - > Syntax kann sich unterscheiden, Semantik aber sehr ähnlich

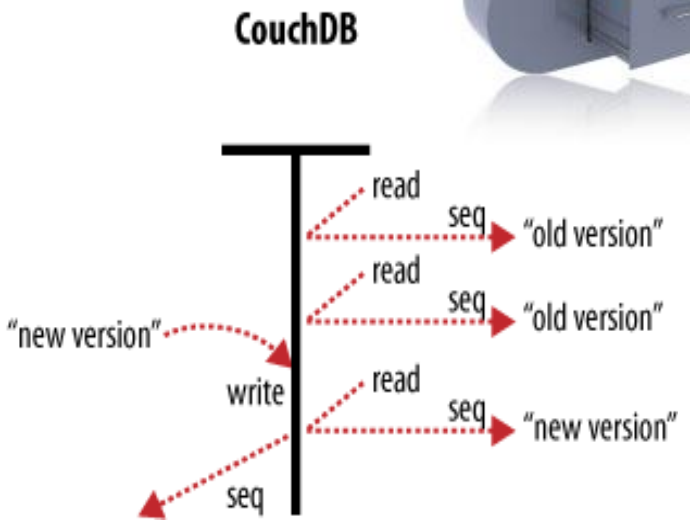


Traditionelle DBs erfordern Schema im voraus
CouchDB erlaubt Strukturierung nach dem Eintragen

Warum CouchDB ?

- MVCC (Multiversion Concurrency Control):
  - Vermeidet lese und schreib Blockaden
  - Daten werden versioniert
  - Änderung einer Datei = neue Version





Warum CouchDB ?

- Programmiert in Erlang, das ausgelegt ist für:
  - Parallelität
  - hohe Verfügbarkeit
  - Fehlertoleranz
  - Auswechseln von Modulen zur Laufzeit etc.

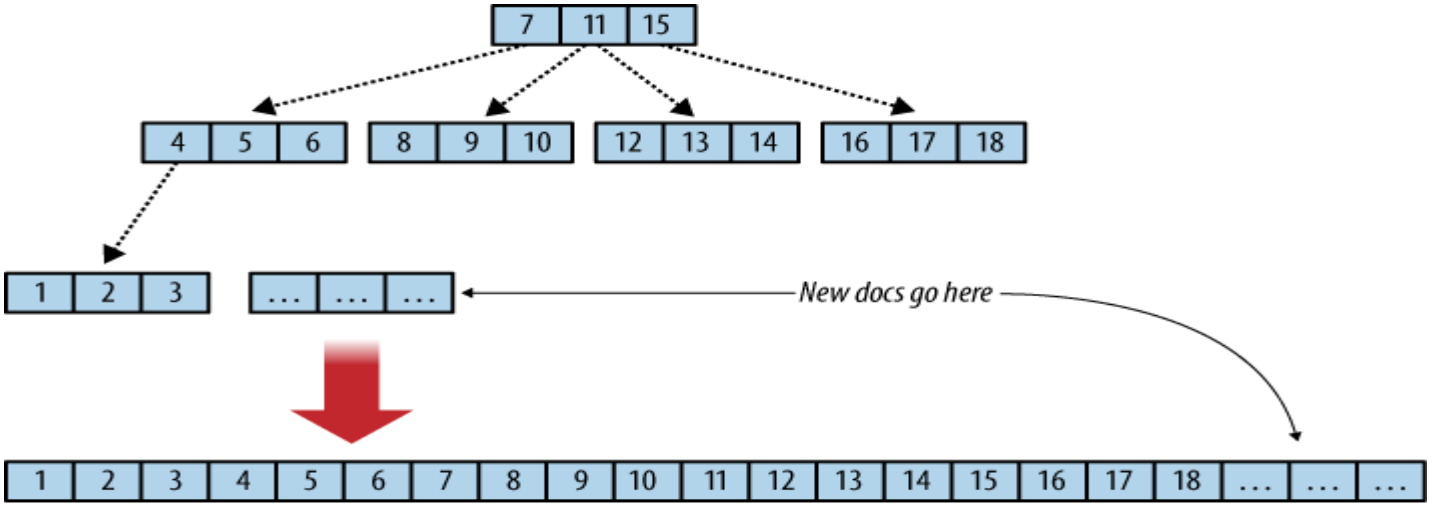
  da damals entworfen für z.B.: Vermittlungsstellen von Telefonnetzen

Warum CouchDB ?

- Benutzung von „B+ Trees“ als Datenstruktur

> „ From a practical point of view, B-trees, therefore, guarantee an access time of less than 10 ms even for extremely large datasets.
>
> —*Dr. Rudolf Bayer, inventor of the B-tree* ”

- Leicht iterierbar
- Blätter immer sortiert
- Etwas mehr Speicherplatz als B-Baum, dafür schneller



Warum CouchDB ? Zusammenfassung

CouchDB

Vorteile:

+ Flexibel, da die Dokumente keinem Schema entsprechen müssen;

+ Dokumente entsprechen sehr gut natürlichen Objekten: einfaches Design, kein objektrelationales Mapping und keine Normalisierung;

+ einfache Bedienung: einfache Web-Schnittstellen und bekannte Techniken wie HTTP-REST, JSON und Javascript;

+ einfacher bidirektionaler Replikationsmechanismus;

+ skaliert sowohl nach oben (Cluster) als auch nach unten (Smartphone).

Nachteile:

- Genügt keinen hohen Konsistenzanforderungen.

Die API

- API kann in vier Bereiche aufgeteilt werden:

1. Server
2. Databanken
3. Dokumente
4. Replikation

Die API - Server

- Aufruf des Servers:

  http://127.0.0.1:5984/

  Antwort:

  {"couchdb":"Welcome","version":"1.2.0"}

  Enthält als JSON-String

  – Willkommensgruß

  – Version

Die API - Datenbanken

- Erstellen einer Datenbank:

  PUT http://127.0.0.1:5984/albums

  Antwort:

  {"ok":true}

- Löschen einer Datenbank:

  DELETE http://127.0.0.1:5984/albums

  Antwort:

  {"ok":true}

Die API - Dokumente

- Anlegen eines Dokuments:
  - Benutzen einer UUIDs/GUIDs
    (falls keine vorhanden http://127.0.0.1:5984/_uuids)

```
PUT
http://127.0.0.1:5984/albums/6e1295ed6c29495e54cc05947f18c8af
-d '{"title":"There is Nothing Left to Lose","artist":"Foo Fighters"}
```

Antwort:

```
{"ok":true,"id":"6e1295ed6c29495e54cc05  947f18c8af","rev":"1-
2902191555"}
```

Die API - Dokumente

- Laden eines Dokumentes:

  GET
  http://127.0.0.1:5984/albums/6e1295ed6c29495e54cc05947f18c8af

  Antwort:

  {"_id":"6e1295ed6c29495e54cc05947f18c8af",“
  _rev":"1-2902191555","title":"There is Nothing Left to Lose","artist":"Foo Fighters"}

Die API - Dokumente

- Updaten eines Dokumentes:

PUT
http://127.0.0.1:5984/albums/6e1295ed6c2949
5e54cc05947f18c8af -d '{"_rev":"1-
2902191555","title":"There is Nothing Left to
Lose", "artist":"Foo Fighters","year":"1997"}’

Antwort:

{"ok":true,"id":"6e1295ed6c29495e54cc05947f1
8c8af","rev":"2-2739352689"}

Die API - Dokumente

- Kopieren eines Dokumentes:

  COPY /somedatabase/some_doc HTTP/1.1 Destination: some_other_doc

  Antwort:

  {"ok":true,"id":"some_other_doc","rev":"355068 078"}

Die API - Dokumente

- Anhänge an ein Dokument hinzufügen:

  PUT
  http://127.0.0.1:5984/albums/6e1295ed6c2949
  5e54cc05947f18c8af/ artwork.jpg?rev=2-
  2739352689 --data-binary @artwork.jpg -H
  "Content-Type: image/jpg“

Antwort:

{"ok":true,"id":"6e1295ed6c29495e54cc05947f1
8c8af","rev":"2-2739352689"}

Die API - Dokumente

- Abruf eines Anhangs:

http://127.0.0.1:5984/albums/6e1295ed6c29495e54cc05947f18c8af/artwork.jpg

Die API - Dokumente

- Löschen eines Dokumentes/Anhanges:

DELETE

http://127.0.0.1:5984/somedatabase/some_doc?rev=1582603387

Antwort

{"ok":true,"rev":"1582603387"}

Funktionen – Design Dokumente

- Design Dokument:
  Normale Dokumente, deren ID mit „_design/“ beginnt, enthalten Code für Anwendungen in JavaScript

- Query Server:
  Führt die Funktionen der Design Dokumente aus

Funktionen – Design Dokumente

Beispiel:

```
{
  "_id" : "_design/example",
  "views" : {
    "foo" : {
      "map" : "function(doc){ emit(doc._id, doc._rev)}"
    }
  }
}
```

Aufruf:

http://127.0.0.1:5984/basic/_design/example/_view/foo

Funktionen – Design Dokumente

- Antwort:

{"total_rows":1,"offset":0,"rows":[
{"id":"acef9ec9ffe29733de784473c70000c7","key":"acef9ec9ffe29733de784473c70000c7","value":"5-bd5f698adcea2ab57aaf66af5c13d86b"}
]}

Funktionen - Views

Views sind die SQL SELECTs:

- Filtern von Dokumenten
- Dateien in einer bestimmten Reihenfolge anzeigen
- Erstellen von Indizes, um Dokumente anhand Wert oder Struktur zu finden
- Nutzen von Indizes für Herstellung von Beziehungen
- Ausführung verschiedener Berechnungen

Funktionen - Views

Beispiel:

```
function(doc) {
  if(doc.date && doc.title) {
    emit(doc.date, doc.title);
  }
}
```

Ausgabe:

Key	Value
"2009/01/15 15:52:20"	"Hello World"
"2009/01/30 18:04:11"	"Biking"
"2009/02/17 21:13:39"	"Bought a Cat"

Funktionen - Validierung

Zum sicherstellen das User nur Dokumente speichern, die Gültig sind (Optional)

- Drei Formen für die Validierung
  1. Inhalt
  2. Struktur
  3. Anfragender-User

- How to:

1. Hinzufügen der „validate_doc_update“ Funktion im Design Dokument
2. Implementieren der Validierung (in JS)

```
function(newDoc, oldDoc, userCtx) {
  throw({forbidden : 'no way'});
}
```

Funktionen - Show

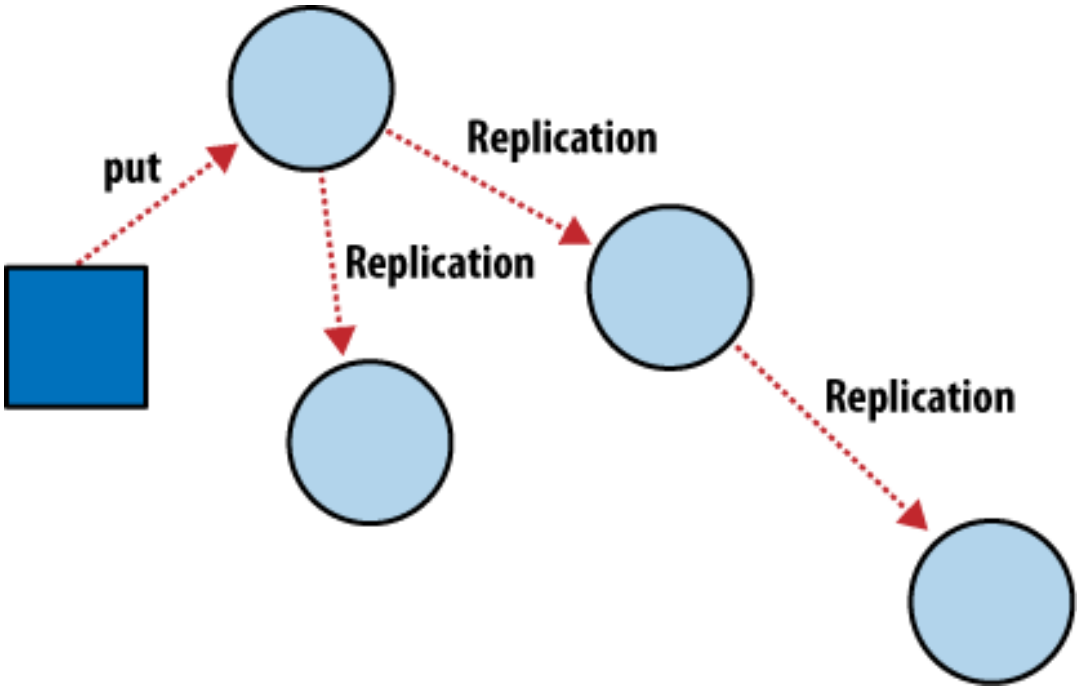
Benötigt um JSON Objekte z.B.: als HTML Objekte zu rendern

- Damals Aufgabe eines Application-Servers
  - Ruby on Rails
- CouchDB beinhaltet diese Funktionalitäten bereits

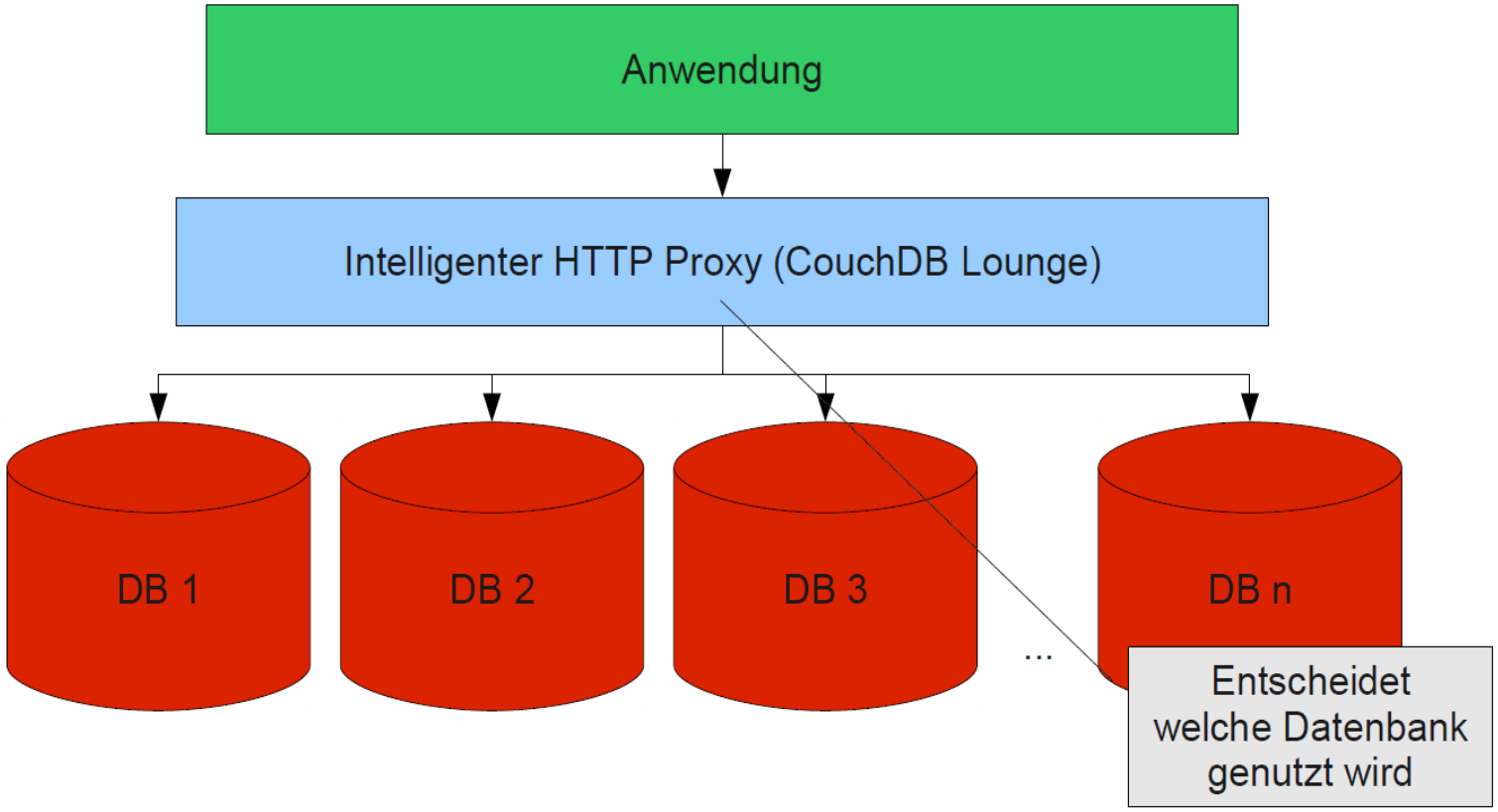
```
function(doc, req) {
  return '<h1>' + doc.title + '</h1>';
}
```

Replikation

Synchronisieren zwischen Datenbanken:



Clustering


Anwendung
Intelligenter HTTP Proxy (CouchDB Lounge)
DB 1
DB 2
DB 3
...
DB n
Entscheidet welche Datenbank genutzt wird

Einsatzgebiete

- Facebook Anwendungen
  - Horoscope
- Webseiten / Webanwendungen
  - Meebo.com bei Umfrage System
  - Swinger , Präsentationsprogramm
- Klassische Softwareprojekte
  - Ubuntu Karmic nutzt CouchDB um Adressen und Lesezeichen zu synchronisieren
- Mobile Anwendungen
  - SpreadLyrics – „song lyrics sharing application“ für Android

-> Viele mehr: CouchDB in the wild

Live Demo

ENDE

Vielen Dank für Eure Aufmerksamkeit

Quellen

- http://guide.couchdb.org/editions/1/de/index.html
- http://www.pro-linux.de/artikel/2/1446/1,einfuehrung.html
- http://www.computerwoche.de/software/software-infrastruktur/2489786/index7.html
- http://de.wikipedia.org/wiki/CouchDB
- http://karl.glatz.biz/files/couchdb-praesentation.pdf