



CouchDB

relax

CouchDB ist ...

keine relationale Datenbank

eine dokument-orientierte Datenbank

für das Web gemacht

skalierbar

Was ist ein Dokument?

Rechnung, Brief, Visitenkarte, ...

Struktur folgt nicht aus Typ:

Visitenkarte A enthält Name, Adresse und Webseite

Visitenkarte B enthält Name, E-Mail und Mobilnr.

natürliche und semi-strukturierte Daten

Dokument-orientiert?

Keine Tabellen mit Zeilen und Spalten

Kein festes Schema für Daten

Daten werden in semi-strukturierten Dokumenten gespeichert

CouchDB Dokumente

In frühen Versionen: XML

Jetzt: JSON

kompakt

portabel

leicht aus nativen Objekten erzeugbar

Beispiel Dokument

```
{  
  "_id": "098F6BCD4621D373CADE4E832627B4F6",  
  "_rev": 4134847,  
  "type": "Rechnung",  
  "Bedienung": "Max Mustermann",  
  "Posten": [  
    {"Name": "Tagesgericht", "Preis": 5.95},  
    {"Name": "Getränk", "Preis": 2.50}  
  ],  
  "verminderte MwSt": false  
}
```

Arbeiten mit Dokumenten via REST

Erstellen HTTP POST /db/

Lesen HTTP GET /db/D0C1DFF

Ersetzen HTTP PUT /db/D0C1DFF

Löschen HTTP DELETE /db/D0C1DFF

HTTP Proxies/Caches können verwendet werden

Varnish

squid

...

PHPillow

```
<?php  
$doc = new myBlogDocument();  
$doc->title = 'New blog post';  
$doc->text = 'Hello world.';  
$doc->save();  
$id = $doc->_id;  
  
$doc = myBlogDocument::fetchById($id);
```

Views

Filtern, sortieren und aggregieren

Map/Reduce

Default Engine: Javascript

Aktualisierung bei Abfrage

Ad-hoc views: `POST /db/_slow_view`

Example Design Document

```
{ _id: "_design/rechnungen",  
  "language": "text/javascript",  
  "views":  
  {  
    "all": {"map": "<map function all>"}  
    "gesamt_umsatz": {  
      "map": "<map function umsatz>",  
      "reduce": "<reduce function>"  
    }  
  }  
}
```

Example view result (all)

GET /db/_view/rechnungen/all

Key	Value
[2008, 12, 10, 19, 17, 23]	{_id:...}
[2008, 12, 22, 21, 43, 40]	{_id:...}
[2009, 1, 3, 22, 10, 45]	{_id:...}
[2009, 1, 5, 20, 56, 19]	{_id:...}

View Function all

```
function(doc) {  
  if (doc.type == "rechnung")  
    emit(doc.date, doc);  
}
```

Example reduce result

GET /db/_view/rechnungen/gesamt_umsatz

Key	Value
null	12345

Example reduce result

GET

/db/_view/rechnungen/gesamt_umsatz?group_level=1

Key	Value
[2008]	10000
[2009]	2345

View Functions: Gesamtumsatz

```
map: function(doc) {  
    if (doc.type == "rechnung")  
        for (var posten in doc.Posten)  
            emit(doc.date, posten.Preis);  
}  
  
reduce: function (keys, values, rereduce) {  
    return sum(values);  
}
```

Replikation

Pull und Push (bi-direktional)

Inkrementell

POST /_replicate

```
{  
  "source": "URI/lokaler Name",  
  "target": "URI/lokaler Name"  
}
```

Einschließlich Applikationslogik - Views

Offline arbeiten

Partielle Replikation von großen Datenbeständen

Applikation kann offline weiter genutzt werden

Änderungen werden später zurück gespielt

Konfliktlösung manuell oder automatisch durch
Applikation

Load Balancing

Master-Slave Replikation

Regelmäßige Synchronisation

HTTP Load Balancer (z.B. nginx)

Konflikte

Konflikt-Flag

Alle Revisionen werden gespeichert

Eine „winning“ Revision – identisch auf allen
Instanzen bei mehreren Konflikten

Konfliktdokumente werden repliziert

Compaction

Zukunftssicher

Cluster Of Unreliable Commodity Hardware DB

In Erlang geschrieben

Lock-freie Architektur

MVCC (Multiversion Concurrency Control)

ACID (Atomicity, Consistency, Isolation,
Durability)

Als verteiltes System geplant – anders als
RDBMS

Projektgeschichte

Ideen aus Lotus Notes

Damien Katz hat CouchDB zunächst in C++
begonnen

Jetzt Open Source Apache Projekt

Von IBM unterstützt

Futon ...

Vielen Dank!

Fragen?

links:

CouchDB: <http://couchdb.org>

PHPillow: <http://kore-nordmann.de/projects/phpillow/>

Vielen Dank an Jan Lehnardt <http://jan.prima.de/>