12.–15.09.2010
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Komm aufs Sofa

NOSQL am Beispiel von Apache CouchDB

Frank Pientka

Materna GmbH, Dortmund

Agenda

- Welche Datenbank braucht das Internet?
  - Das CAP-Theorem
  - Was ist NOSQL?
  - Map-Reduce-Algorithmus
- REST-Architektur, JSON-Dokumente
  - Wie Dokumente speichern?
  - Wie Dokumente verarbeiten?
- Apache CouchDB
  - Architektur
  - Funktionen
  - Verwendung
  - Beispiel
- Fazit

Über den Web 2.0-Wolken

Die Architektur des Internets verändert sich!

- CLOUD, IaaS, PaaS
- HTML5: lokaler Speicher, SimpleDB API
- Verteilter, paralleler Speicher (Map&Reduce)
- Ressourcenorientierung (REST)
- Dynamische Oberflächen (AJAx)
- Anwendungen durch Benutzer erstellt

N★SQL

Eine Ära geht zu Ende…

- *"The relational model of the 70s is not necessarily the answer"* (Stonebraker)

"The End of an Architectural Era", 2007, Michael Stonebraker et al.

NotOnlySQL-Datenbanken

- Spaltenorientiert: Hadoop/ Hbase, Cassandra, Hypertable
- Dokumentenorientiert: Riak, MongoDB, CouchDB
- Schlüssel/Wertorientiert: Simple DB, Azure Table Storage, Redis, Dynamo, BigTable
- Graphorientiert: GIS, Neo4j, GraphDB

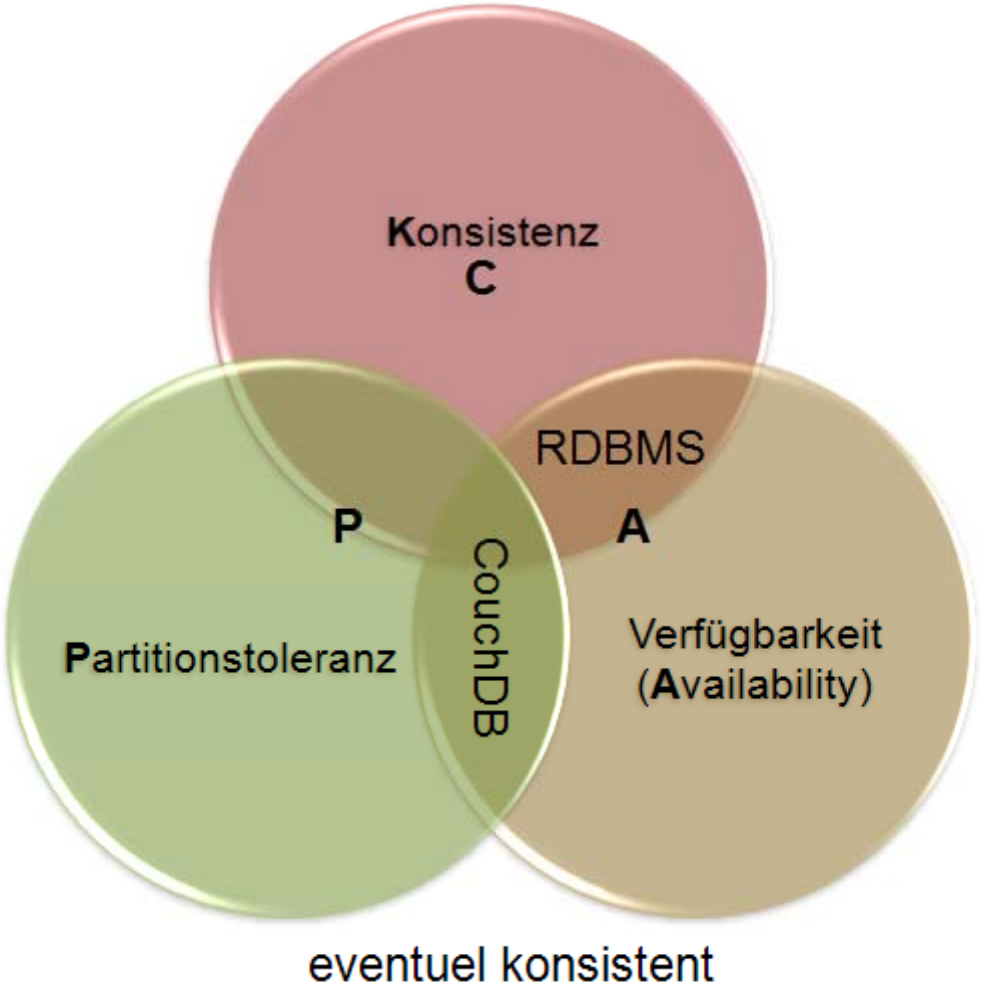
Gab es da nicht:

- 60er hierarchisch: IMS, LDAP, XML (Tamino, Xindice)
- 70er netzwerkartig: CODASYL
- 80er Hash-DB: xDBM Ken Thompson, BerkleyDB
- 80er relational: ORACLE, DB2, mySQL
- 90er objektorientiert: db4o, Versant

➔ hybride Datenbanken SQL:2008


NO
SQL
Now
What

CAP-Theorem: nur 2-aus3 (Brewer 2000)



NoSQL und Brewers CAP-Theorem

http://blog.nahurst.com/visual-guide-to-nosql-systems

ACID versus BASE Transaktionen

ACID:

- **A**tomic
- **C**onsistent
- **I**solated Transactions cannot interfere with each other.
- **D**urable

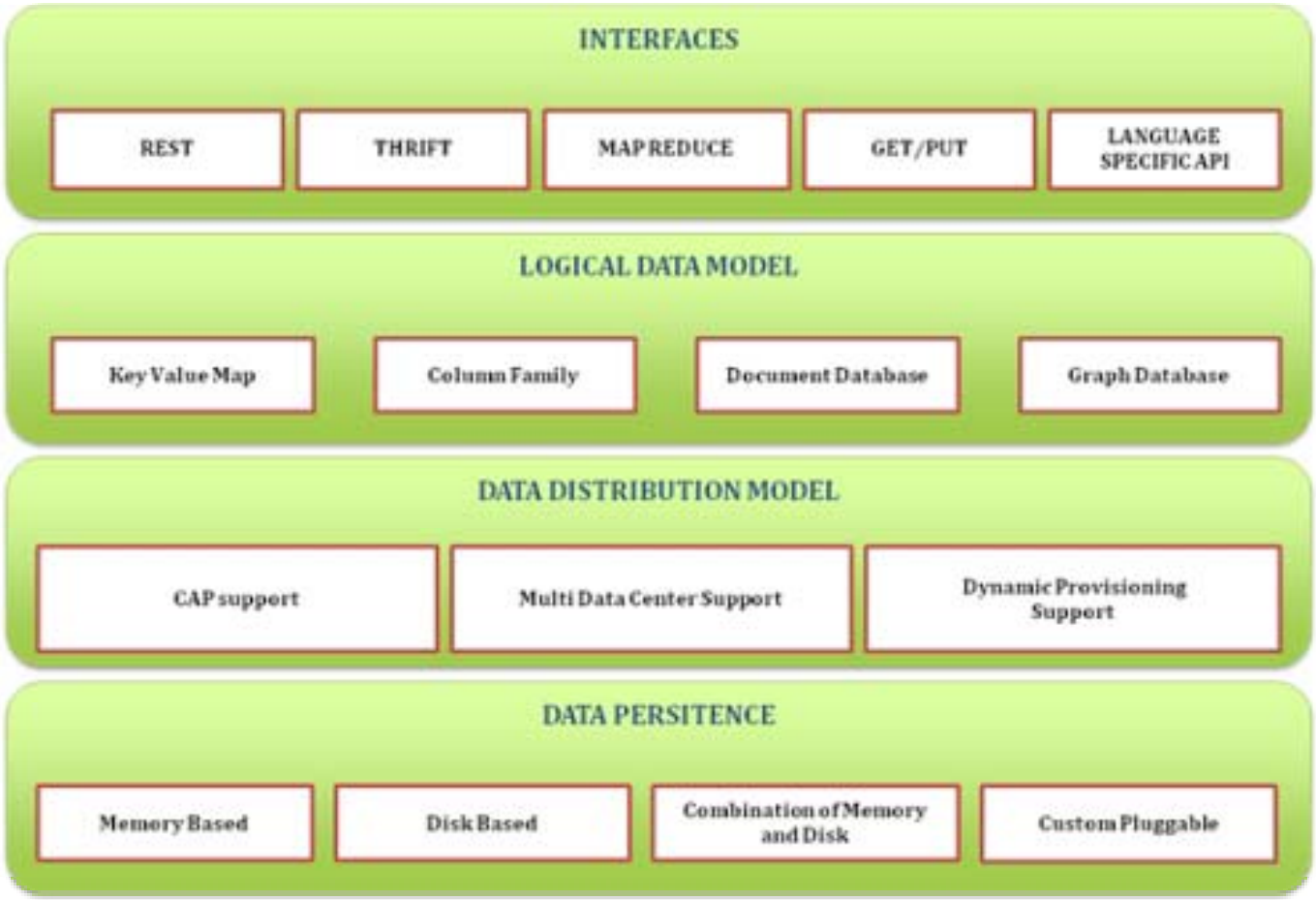
BASE:

- **B**asic **A**vailability
- **S**oft-state
- **E**ventual consistency

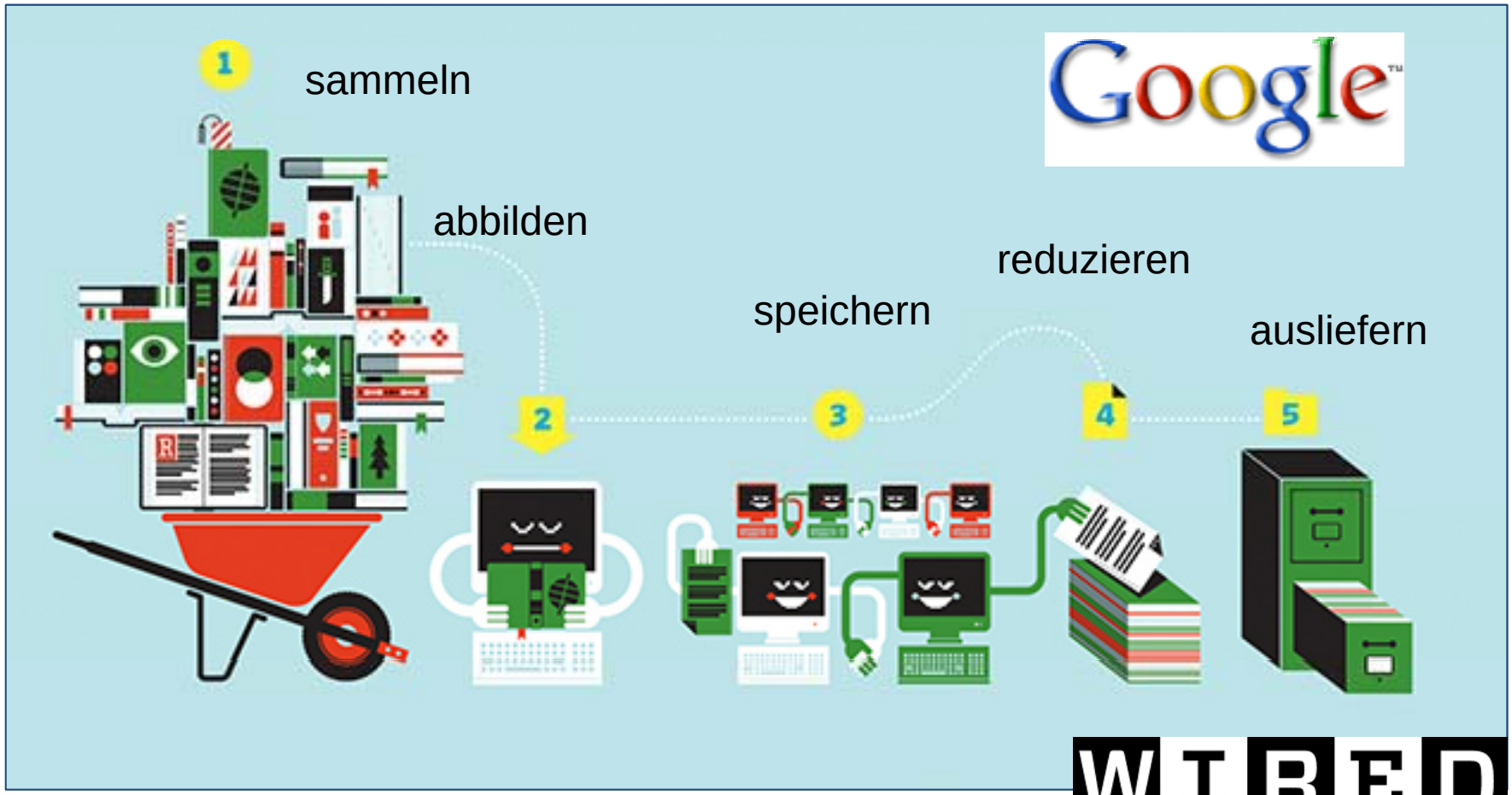
Verfügbarkeit wichtiger als Konsistenz für schwächere Transaktionsanforderungen

”Your Coffee Shop doesn’t use Two-Phase-Commit“ Gregor Hohpe 2005

4 Eigenschaften von NOSQL-Systemen



Sorting the World with MR



Quelle: (WIRED MAGAZINE 16.07)

Das Map&Reduce-Verfahren

in 2 Phasen parallel ausgeführt, wie Pipes&Filter (UNIX)

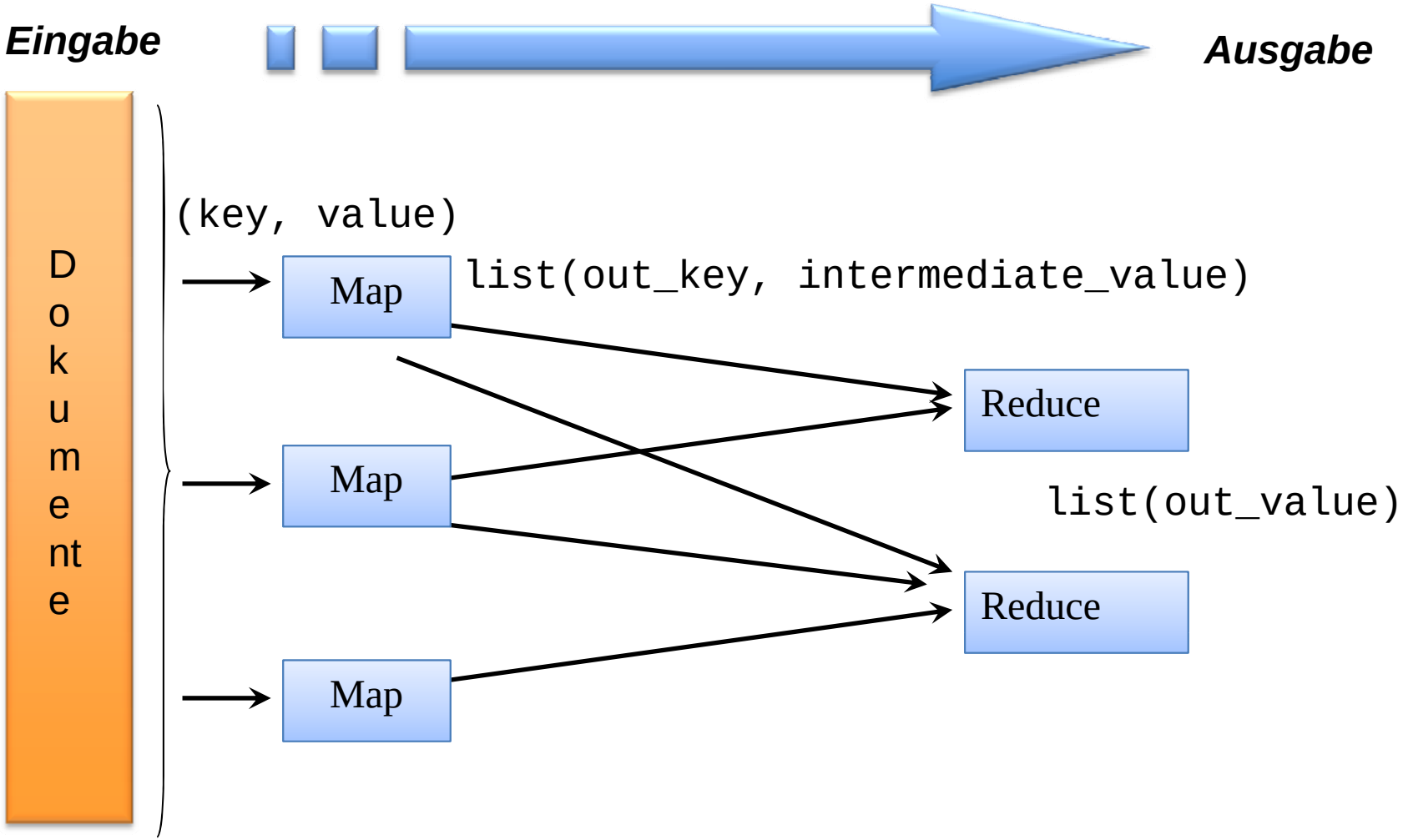
1. Mapper-Funktion

```
map (in_key, in_value) ->
   list(out key,
   intermediate_value)
```

2. Reduce-Funktion berechnete Werte. Aggregatfunktion, wie COUNT, SUM, AVG, MIN, MAX:

```
reduce (out key,
list(intermediate_value)) ->
list(out value)
```

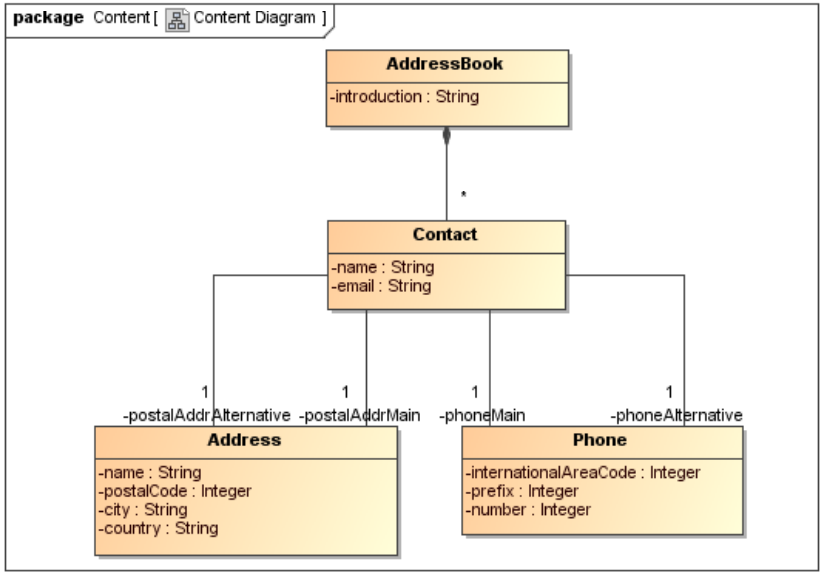
MapReduce-Verfahren



Agenda

- Welche Datenbank braucht das Internet?
  - Das CAP-Theorem
  - Was ist NOSQL?
  - Map-Reduce-Algorithmus
- REST-Architektur, JSON-Dokumente
  - Wie Dokumente speichern?
  - Wie Dokumente verarbeiten?
- Apache CouchDB
  - Architektur
  - Funktionen
  - Verwendung
  - Beispiel
- Fazit

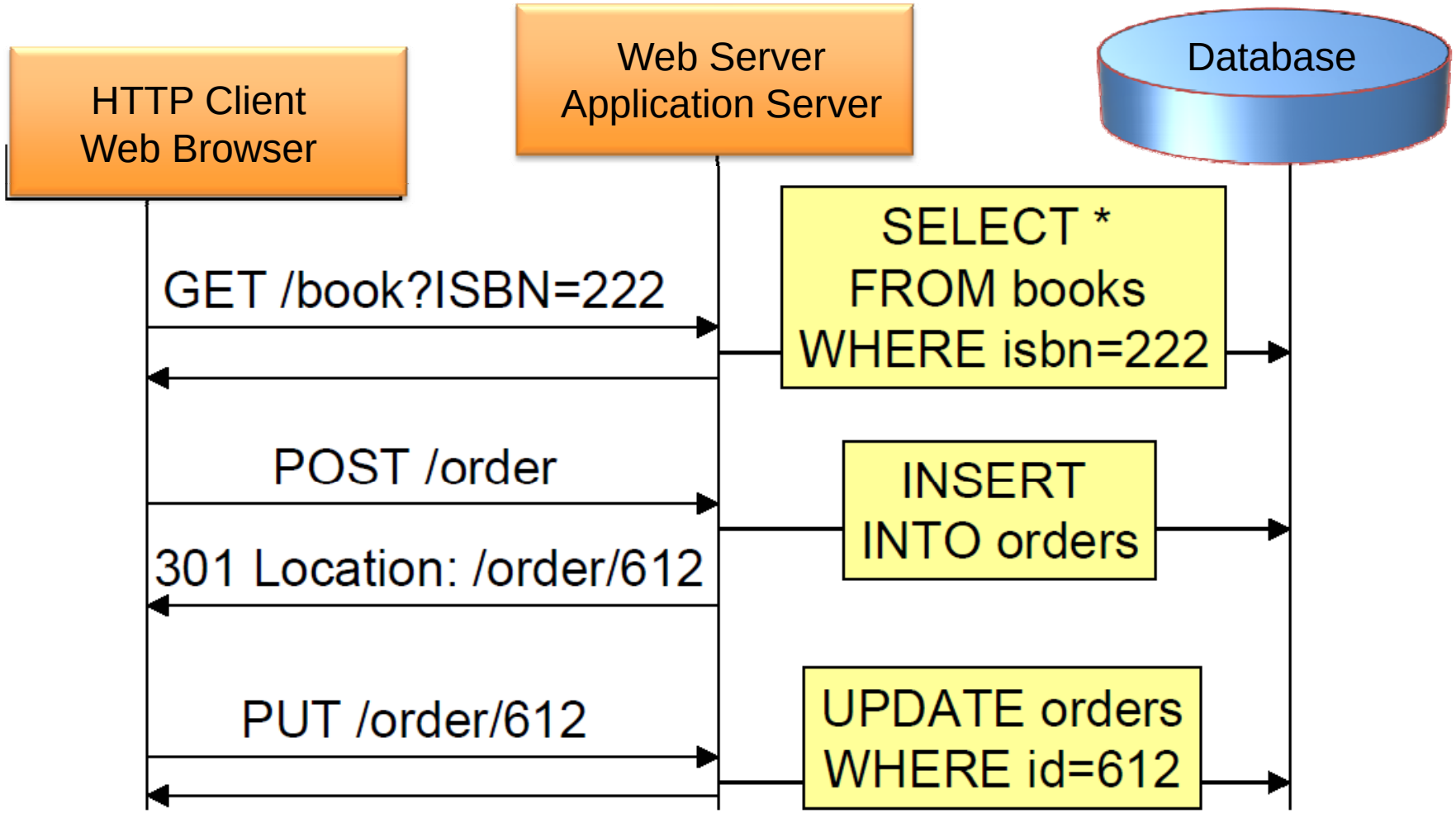
Daten sind oft Dokumente



Standardisierung + Modellierung
Doch Strukturen können sich oft ändern!

„Grau ist alle Theorie und grün des Lebens goldner Baum“ (Faust)

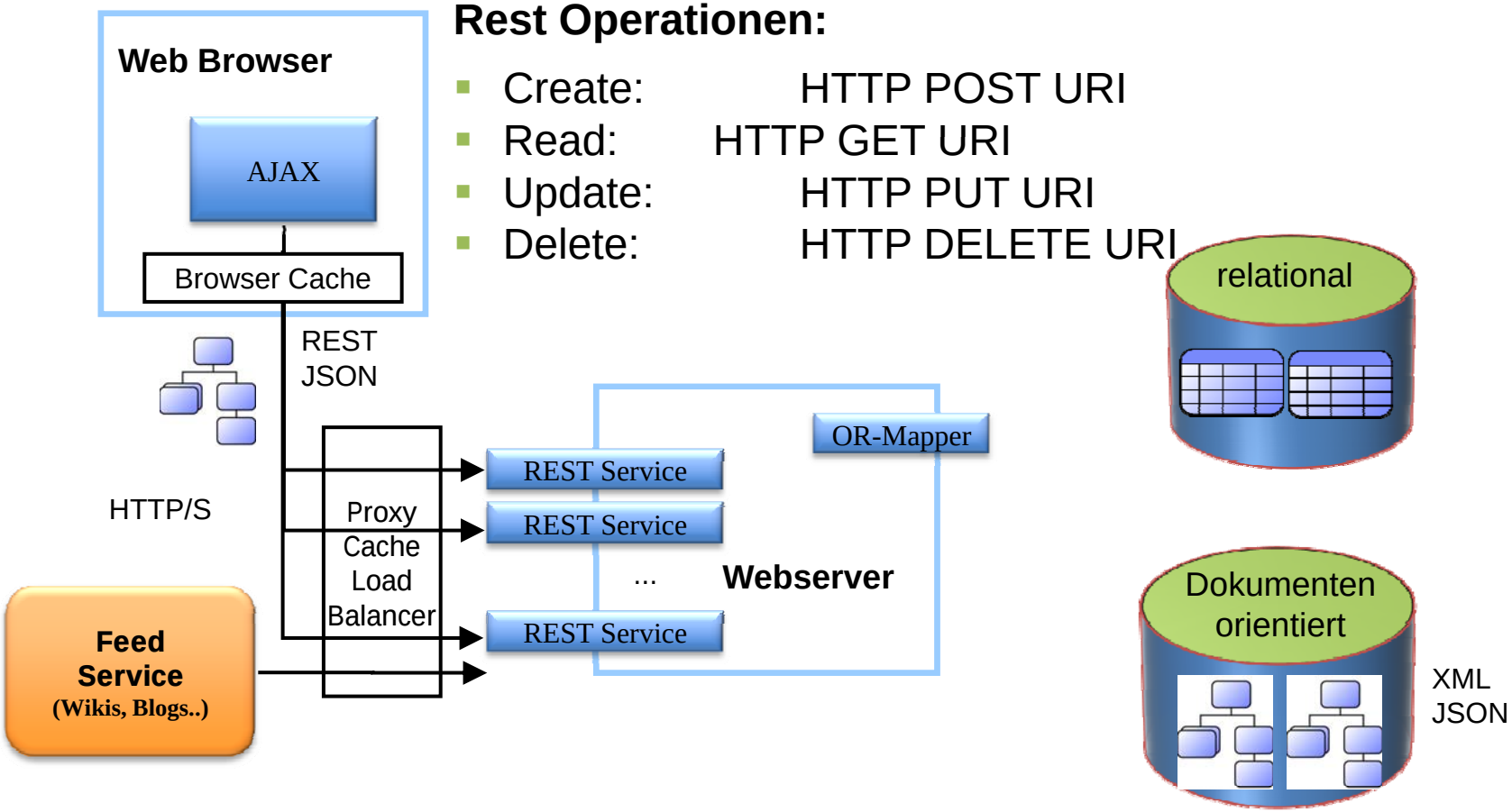
Die Restverarbeitung



REST-Operationen: HTTP-API

- **Create**
- HTTP PUT /db/my_doc_id
- **Read**
- HTTP GET /db/my_doc_id
- **Update**
- HTTP PUT /db/my_doc_id
- **Delete**
- HTTP DELETE /db/my_doc_id

Web 2.0 Architektur



Rest Operationen:

- Create: HTTP POST URI
- Read: HTTP GET URI
- Update: HTTP PUT URI
- Delete: HTTP DELETE URI

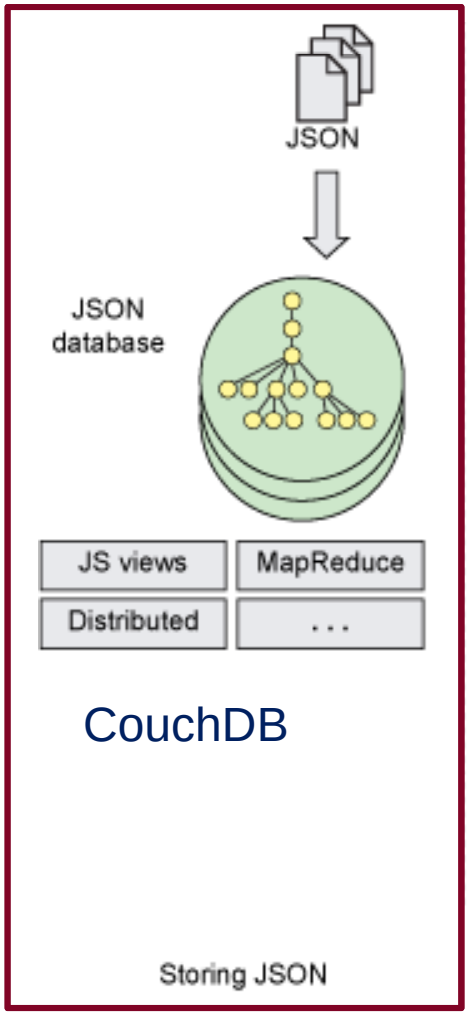
www.json.org RFC-4627

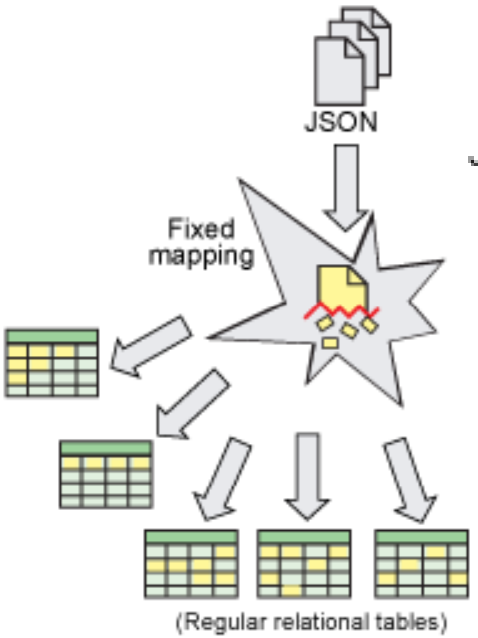
- Beliebtes Austauschformat im WEB
- einfach mit vielen Sprachen verwendbar und in XML umwandelbar
- verbraucht weniger Bandbreite und CPU, als XML
- schemafrei
- Verarbeitung, Speicherung nicht standardisiert

jCard-Vorschlag

```
{"person" : {"name" : "Frank Pientka"},
"business":[{"name":" dipl. Inf. Frank Pientka"},
 {"org":"Materna GmbH"}],
"contact_details" : [
    {"phone" : "+49 231 5599 8854"},
    {"fax" : "+49 231 5599-272 "},
    {"email" : "frank.pientka@materna.de"},
    {"adr":{"street-address":"Vosskuhle 37",
 "region":"NRW",
 "locality":"Dortmund",
 "postal-code":"44141",
 "country-name":"Germany"}},
 "url":"http:www.materna.de”  ]}
```

Wie JSON-Dokumente speichern?





XML-Datentyp

Datenbanken mit Java+REST/WS

- JSR-311: JAX-RS 1.1
  - Jersey
  - Restlet
  - RESTEasy
  - CXF, WINK
- Datenbank
  - sqlREST
  - EclipseLink DBWS
  - Eclipse STP REST Support

Dokumente in der Datenbank

```
{"my_key" : "my_value"}
{"my key" : "another value",
"different_key" : "another value"}
{"my_array_key" : ["value 1", "value 2",
 "etc"]}
{"my_hash_key" : { "internal_key" :
 "internal value" }
```

CouchDB-Dokument sind im JSON-Format

Wie Dokumente verarbeiten?

- JsonML (JSON Markup Language)
- Browser-Side Templating (JBST)
- JSON (JavaScript Object Notation)
- JSON-REST, JSONQuery, JSON-Processor

Map-Reduce: 1958 in LISP (John McCarthy)

```
map: function(doc)
 {emit(null,doc);}
Reduce: function (key, values,
 rereduce)
{return
 aggregationsum(values);}
```

Dokumente abfragen

Nur das Feld `username` der Dokumente, die als type `user` haben ausgeben:

```
map: function(doc){
 if(doc.type == "user"){emit( doc.username,
 doc );}
}
```

Anzahl der gleichen `username-Werte` mit der Reduce-Funktion ausgeben:

```
function(key, values, rereduce)
{return values.length;}
```

SQL: `SELECT username, count(*) FROM users GROUP BY username`

Dokumente gruppieren

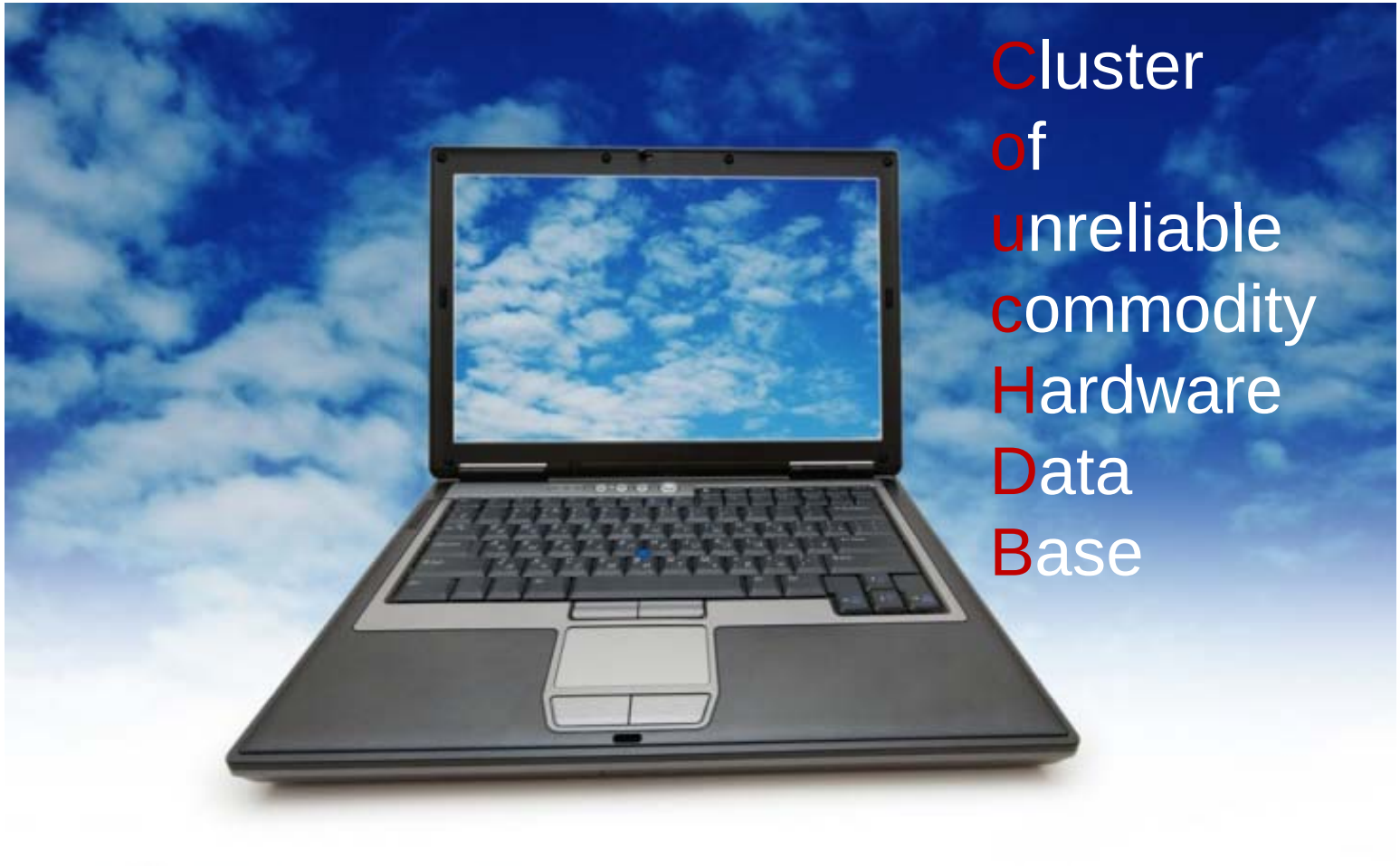
Dokumente gruppiert nach Anhängen: GROUP BY

```
map: function(doc) {
  if (doc. attachments) {
    emit("with attachment", 1);
  }
  else {
    emit("without attachment", 1);
  }
}
reduce: function(keys, values) {
   return sum(values);
}
```

Agenda

- Welche Datenbank braucht das Internet?
  - Das CAP-Theorem
  - Was ist NOSQL?
  - Map-Reduce-Algorithmus
- REST-Architektur, JSON-Dokumente
  - Wie Dokumente speichern?
  - Wie Dokumente verarbeiten?
- Apache CouchDB
  - Architektur
  - Funktionen
  - Verwendung
  - Beispiel
- Fazit

Über den Wolken…




Couchio
proudly presents
Apache
CouchDB
1.0
"The best thing since sliced bread"
Jan LEHNARDT
Noah SLATER
Christopher LENZ
J Chris ANDERSON
Paul J DAVIS
Dr. Adam KOCOLOSKI
Mark HAMMOND
Jason DAVIES
Benoît CHESNEAU
Filipe MANANA
Directed and developed by Damien KATZ – In cinemas across the world from July 14 2010 – www.couch.io/get

Apache CouchDB 1.0

- dokumenteorientierte, schemafreie, verteilte Datenbank mit Versionierung seit 2004
- Skalierbar: Abfragen mit dem Map&Reduce-Verfahren
- Anwendungen und Daten in der DB können repliziert werden
- In Erlang geschrieben
- Über REST-HTTP-API ansprechbar
- Ausfalltollerant mit MVCC
- NICHT: Ersatz für RDBMS, ODBMS

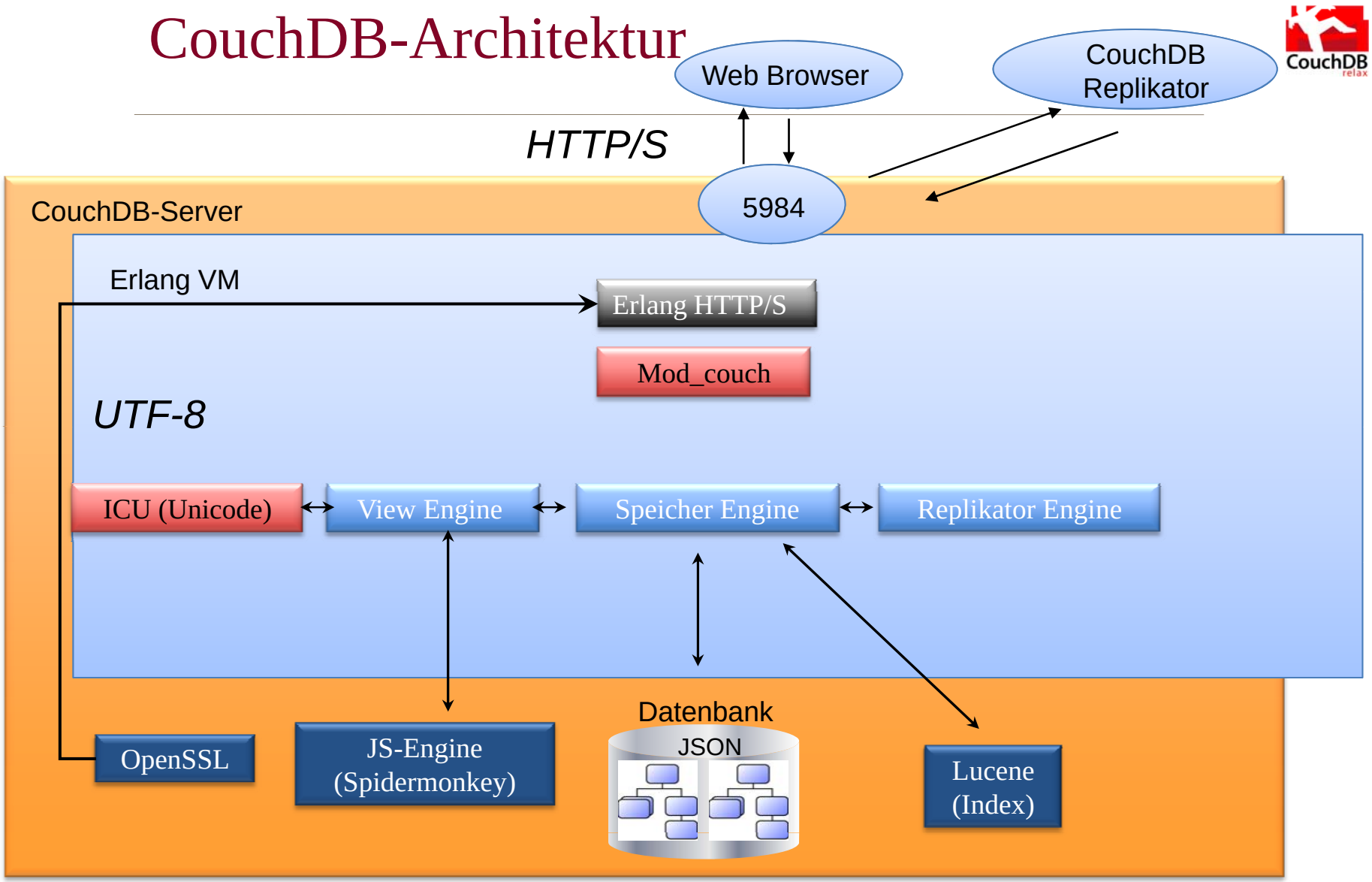
Apache CouchDB Merkmale

- Multi-Version Concurrency Control (MVCC) mit Konfliktvermeidung ohne Sperren (immer konsistent) ID+Version
- Dokumente sind mit b-Bäumen inkrementell, indiziert (DocID)
- automatische Komprimierung
- Views (temporär, permanent)
- inkrementelle Replikation
- „offline fähig“

Apache CouchDB Konzepte

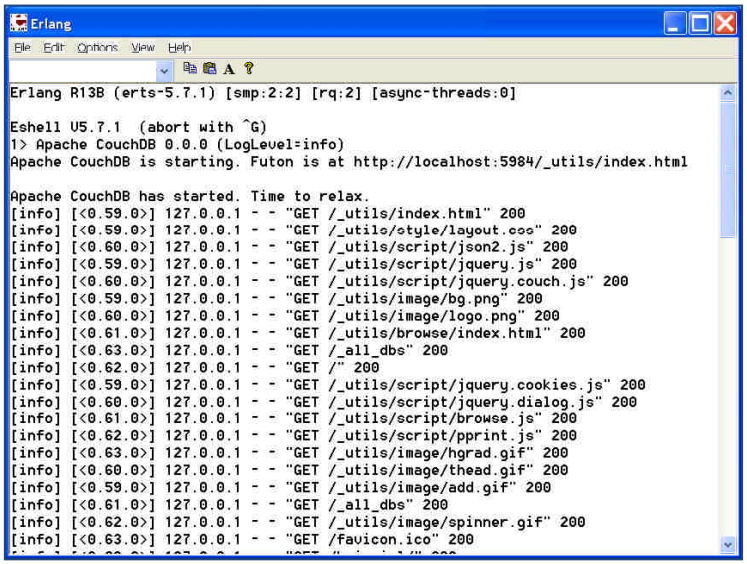
- Dokumente
  - Schema-frei, semi-strukturiert mit ID
  - JSON
- Sichten
  - dynamische Abfragen (Aggregate, Join, Report)
- Verteilt
  - Inkrementelle, bidirektionale Peer-Replikation

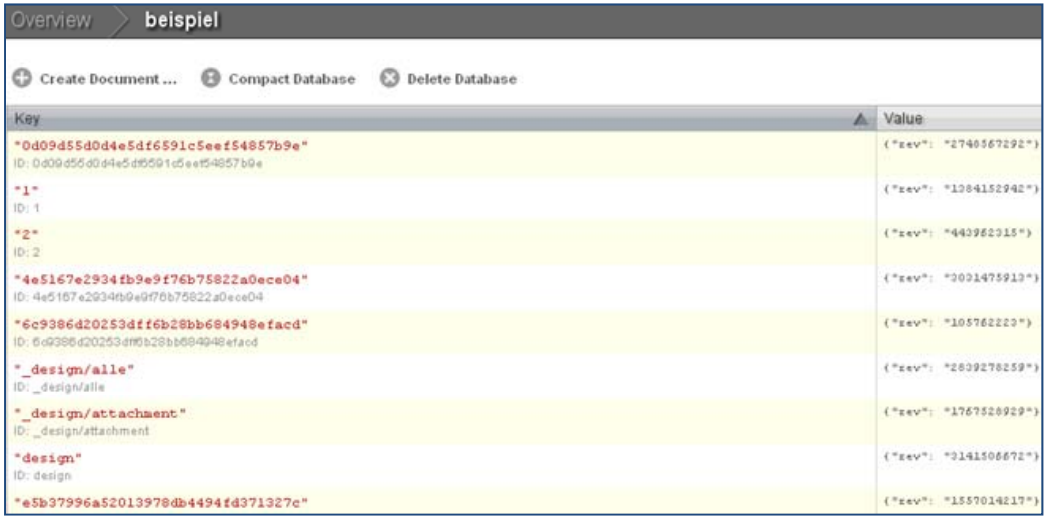
CouchDB-Architektur



Apache CouchDB Umgebung

Erlang/OTP-Interpreter R14B (BSD, Linux, Solaris, Linux, VxWorks, Windows, MAC OS x)
Kommandozeile, Browser
Verwaltungs-GUI: Futon





Design Dokumente

- Werden indiziert bei Aufruf
- Gespeicherte View mit folgender Konvention:
  - ID muss mit _design/ beginnen
  - Das view-Attribut hat MAP und evtl. Reduce-Funktion

Design Dokumente

```
{" id":" design\/application"," rev"
 :"2693121204","views":{
   "viewname": {
 "map": "function(doc){emit(null,
 doc); } " ,
"reduce": "function(keys, values)
 {return sum(values);}"
   }}}
curl -X GET
 http://127.0.0.1:5984/test suite db
 /_design/application
```

REST-API mit curl -X

Information zur Datenbank ausgeben oder erstellen:

```
GET http://127.0.0.1:5984/test_suite_db/
```

Alle Dokumente in der Datenbank anzeigen:

```
GET http://127.0.0.1:5984/db/_all_docs
```

Dokument anzeigen:URI */{dbname}/{docid}/{viewname}*

```
GET http://127.0.0.1:5984/db/dic_id
```

Versionen zu Dokument anzeigen

```
GET http://127.0.0.1:5984/db/dic_id?revs=true
GET
  http://127.0.0.1:5984/db/dic id?revs info=true
```

REST-API mit curl -X

BULK_LOAD

```
POST http://127.0.0.1:5984/test_suite_db/_bulk_docs
-d'{"docs":[{"key":"baz","name":"bazzel"},
  {"key":"bar","name":"barry"}]}'
```

BULK_READ

```
{dbname}/_all_docs?include_docs=true
{dbname}/_all_docs?include_docs=true&startkey="aa"&endkey="
  zz"
{dbname}/_all_docs?startkey="doc2"&limit=2&descending=true
{dbname}/ all docs by seq
```

CouchDB HTTP-API-Referenz

Bereiche:

- Authentication:
- Database document methods:
- Database methods:
- Server configuration:
- Server-level methods:
- Special design documents:
- Special design document handlers:
- Special non-replicating documents:

http://wiki.apache.org/couchdb/Complete_HTTP_API_Reference

CoucDB-API-Implementierungen

- JS #3
- PHP #4
- JAVA, Android SDK # 4
- Groovy/Grails #2
- GWT #2
- .NET #3
- Ruby #4
- Erlang #4
- Perl #5
- Python #3
- Scala #1
- Clojure #2
- Common Lisp #2

Replikationsaufruf

```
curl -X POST http://localhost:5984/ replicate
-d '{"source":"http://localhost:5984/remote-db",
"target":"local-db"}'
```

Mit Futon Webkonsole

Map-,Reduce-Funktionen

CouchDB relax

Versionskonflikte: SELECT * FROM WHERE

```
function(doc){if(doc. conflicts){
emit(null,null);}}
```

Alle Dokumente mit Anhängen: SELECT * FROM

```
function(doc) { if (doc._attachments) {
 emit(doc._id, null); } }
```

Zähle Anzahl Anhänge: SELECT COUNT(field) FROM

```
map:function(doc) { if (doc._attachments) {
 emit("with attachment", 1); } else {
 emit("without attachment", 1); } }
reduce: function(keys, values) { return
 sum(values);}
```

SQL und CouchDB gegenübergestellt

SQL	CouchDB
festes Schema	dynamisches Schema
Tabellen von Daten, Menge, Zeilen	Sammlung von Dokumenten variabler Struktur, Multisets
normalisiert	denormalisiert
Objekte über mehrere Tabellen verteilt	Dokumente beschreiben sich selbst
Zum Verarbeiten der Objekte muss Schema bekannt sein	Zum Verarbeiten muss nur Dokumentenname bekannt sein
dynamische Abfragen mit statischem Schema	statische Abfragen mit dynamischem Schema

Couch-Beispiele für SQL-Umsteiger
http://books.couchdb.org/relax/reference/views-for-sql-jockeys

Beispiel CouchDB4J mit Java

```
Session s = new Session("localhost",5984);
Database db = s.getDatabase("foodb");
Document doc = db.getDocument("documentid1234");
doc.put("foo","bar");
db.save(doc);
Document newdoc = new Document();
doc.put("too","baz");
db.save(doc);
ViewResult result = db.getAllDocuments();
 for (Document d: result.getResults()) {
     System.out.println(d.getId()); }
 ViewResult resultAdHoc = db.adhoc(
     "function (doc) { if (doc.foo=='bar') {
         return doc; }}");
    }
```

Car-Beispiel Divan mit C#: Objekt speichern

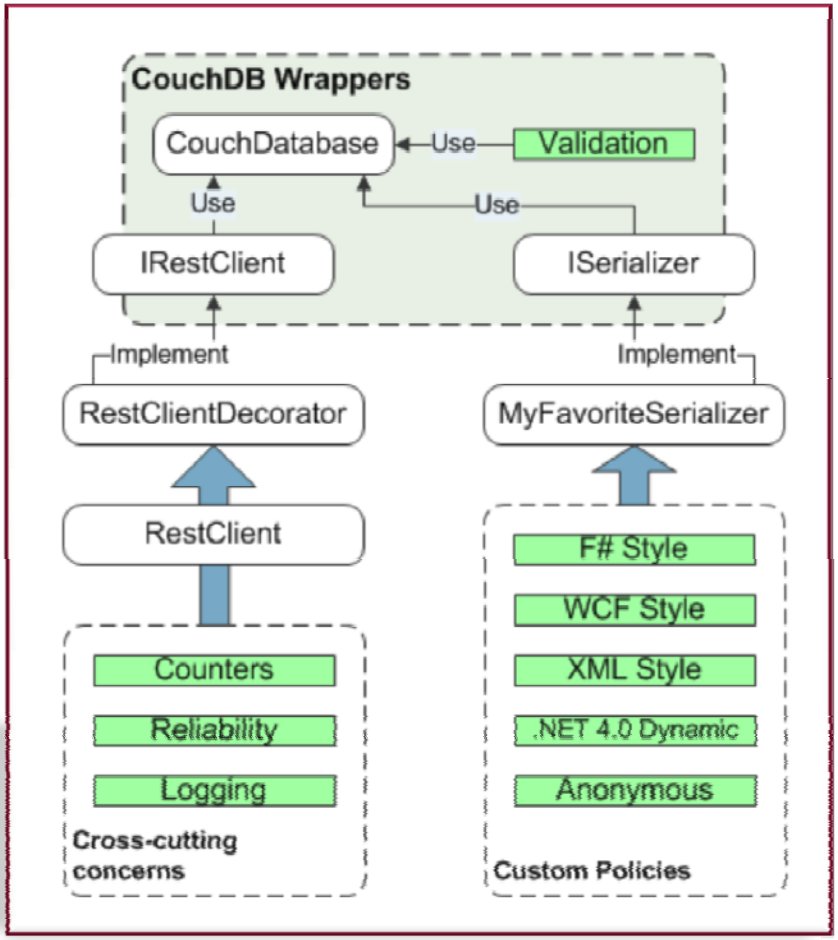
```
private class Car : CouchDocument
{ public string Make;
public string Model;
public int HorsePowers;
…}
...
var server = new CouchServer("localhost", 5984);
var db = server.GetDatabase("trivial");
Car car = new Car("Saab", "93", 170);
db.SaveDocument(car);
// Load all Cars as CouchJsonDocuments
db.GetAllDocuments().First().ToString());
...
```

Car-Beispiel Divan mit C#: Objekt lesen über REST-API

```
curl -X PUT http://127.0.0.1:5984/trivial
curl -X POST http://127.0.0.1:5984/trivial/ -d
' {"docType": "car", "Make": "Saab",
"Model": "93", "Hps": 170}'
curl -X GET
  http://127.0.0.1:5984/trivial/_all_docs?include_docs=true
```

Benötigt .NET 2.0, JSON.NET 3.5 und NUnit 2.4.
Alternativen: Relax, SharpCouch

CouchDB-.NET Adapter NCouch



Beispiel zu Fuß

Beispieldatenbank *artikel_db* anlegen:

curl -X PUT http://127.0.0.1:5984/artikel_db

{"ok":true}

Artikel anlegen:

curl -X POST http://127.0.0.1:5984/artikel_db -d
'{"article":"Komm aufs Sofa„„
"magazin":„kaffeklatsch",
authors":["Pientka"]},

Artikel anzeigen:

curl -X GET http://127.0.0.1:5984/artikel_db/_id

Tipps&Tricks

- LINUX, MacOS X hat beste Unterstützung
- Einschränkungen bei WINDOWS (Keine Komprimierung Download unter www.couchone.com/get, Fehler IE-JS in Futon und Testsuite)
- Reservierte Feldnamen beginnen mit _
- Bei Replikation: keine Autoincrement-ID
- Besser GUID oder ID aus fachlichen Feldern
- config.ini: Indizes per Skript neu erzeugen, Sicherheit für Admin, Benutzer
- Raw Collation Views, native in Erlang statt mit JS
- 1.0.1, 0.11.2 einsetzen

Authorisierung, Authentifizierung

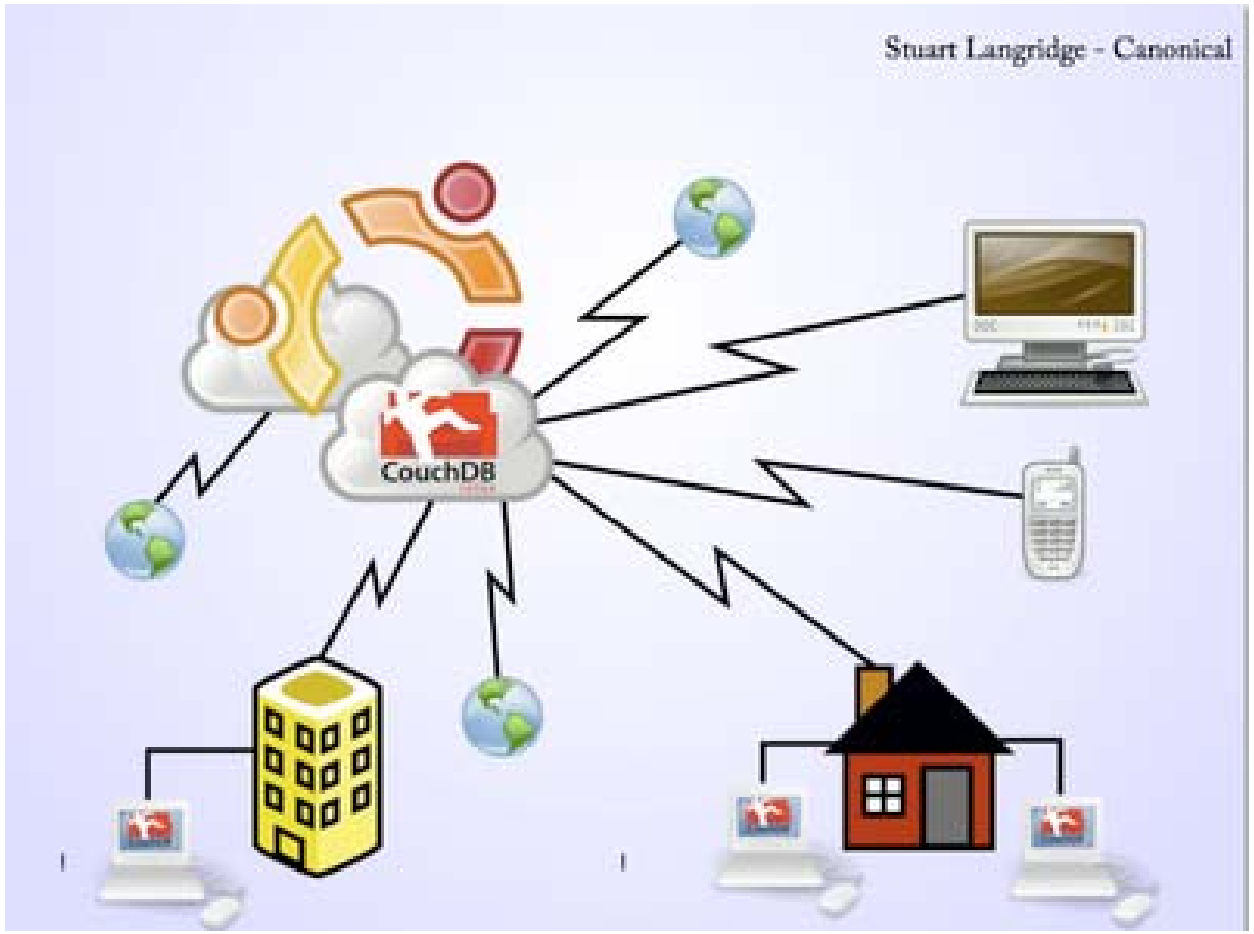
- Benutzer-Rollen:
  - database readers
  - database admins
  - server admins
- Authentifizierung:
- Oauth
- Cookie timestamped
- Default (HTTP basic authentication)

CouchDB in freier Wildbahn

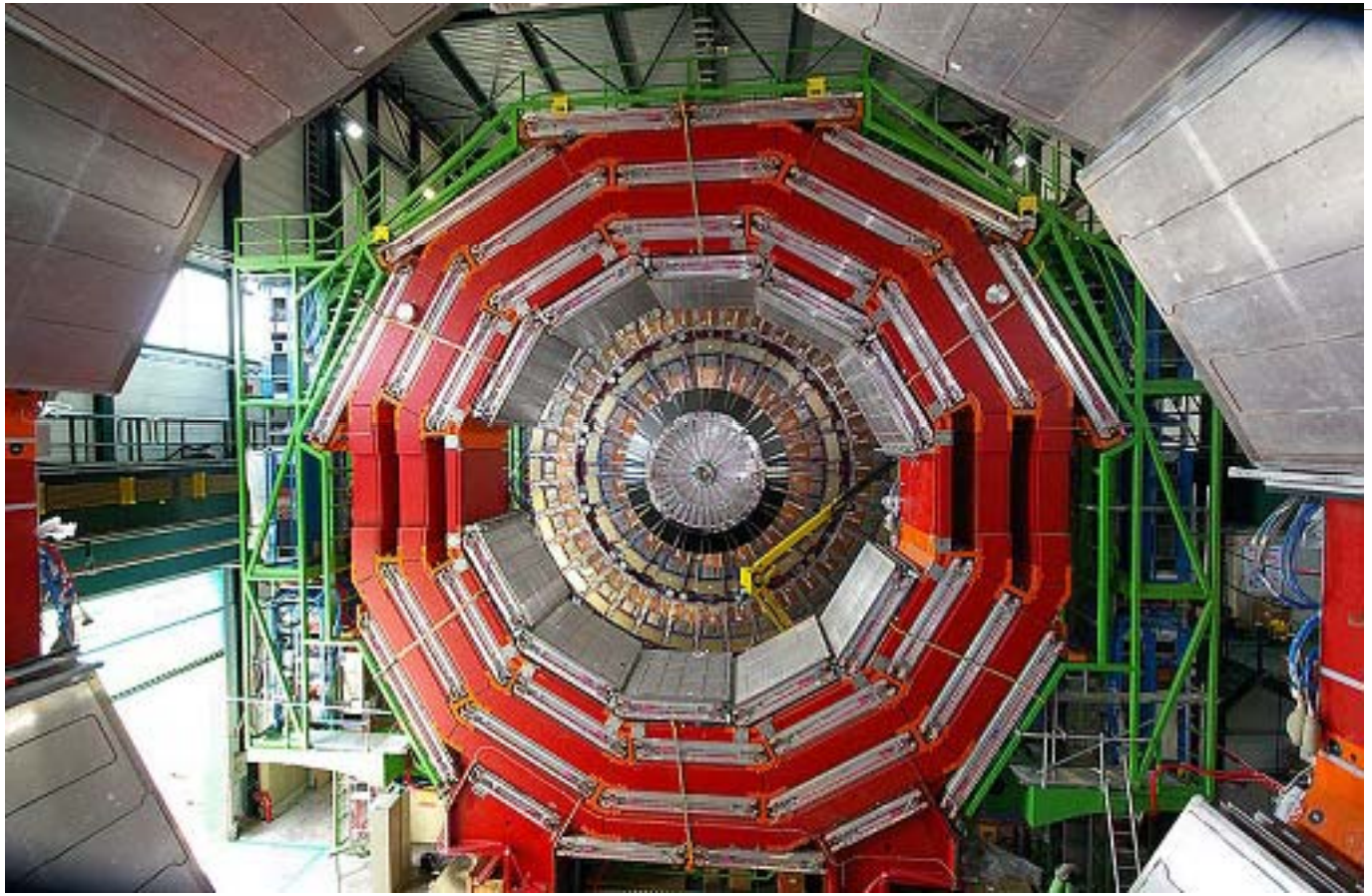
- Hosting: hosting.couch.io
- Ubuntu One: Evolution (Synchronisation von Kontakten)
- Get Things Done with Quickly
- BigCouch-Cluster (Cloudia)
- Fallstudie: Migration von mySQL zu CouchDB: johnpwood.net/tag/couchdb-case-study
- Beliebt bei Ruby, PHP und JS-Programmierern
- Webseiten: wego.com, bbc.co.uk, hg.toolness.com/browser-couch

CouchDB Ubuntu One: Cloud, PC, Smartphone

CouchDB auf über 10 million PC und in der Cloud mit Ubuntu 9.10



CouchDB in freier Wildbahn: CERN Large Hadron Collider



Agenda

- Welche Datenbank braucht das Internet?
  - Das CAP-Theorem
  - Was ist NOSQL?
  - Map-Reduce-Algorithmus
- REST-Architektur, JSON-Dokumente
  - Wie Dokumente speichern?
  - Wie Dokumente verarbeiten?
- Apache CouchDB
  - Architektur
  - Funktionen
  - Verwendung
  - Beispiel
- Fazit

Fazit CouchDB

CouchDB relax

Geeignet für

- einfach strukturierte schemalose Webdokumente
- AJAJ-Architektur
- Dezentral gespeicherte Daten
- Enterprise 2.0: mehr Leser, als Schreiber, EDA/CEP
- Einfache Auswertung großer Dokumentenbestände
- Verteilte Webanwendungen mit HTML5

Hindernisse:

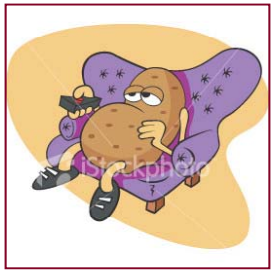
- Schlechte Dokumentation, wenig Best-Practicies
- Konkurrenz (Riak, MongoDB)

Fazit

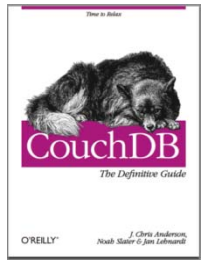
- Relationale Datenbanken dominieren mit hybriden Erweiterungen weiterhin
- Bei neuen verteilen Webarchitekturen bietet NOSQL Vorteile
  - Replikation: Offline/Online
  - Universelle REST-Schnittstelle
  - Ausfalltolleranz und lineare Skalierbarkeit: BASE vs ACID, CAP
  - OpenSource passt gut zu Cloud
  - Aus der Praxis für Anwendungen der neuen Generation (mobil, verteilt)
- Kombination sinnvoll, wenn SQL nicht immer die Antwort ist
- Polyglotte Persistenz: Sprachvielfalt auch bei Datenbanken
- Nicht alles lässt sich de-/normalisieren
- Referenzielle Integrität ist für Web 2.0 weniger wichtig

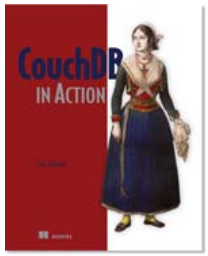
Fazit

- REST-basierte Anwendungen haben andere Architekturanforderungen
- MR
  - gut verstanden (Theorie, Pattern)
  - bessere Testunterstützung nötig
  - Teil von einigen auch kommerziellen Datensystemen
- Spezialisierte Datenbanken werden fürs Web benötigt (~mySQL-Ersatz)
- NOSQL ist OpenSource (aus der Praxis für die Praxis)
- RDBMS und MR ergänzen sich, doch es gibt zu viele SQL-CouchPotatoes



Infos CouchDB





- *Apache CouchDB: couchdb.apache.org*
- *CouchDB in Action: Christopher Chandler, Manning, 2009*
- *CouchDB: The Definitive Guide, Chris Anderson, Noah Slater, Jan Lehnardt, O'Reilly, 2009*
- *Beginning CouchDB: Joe Lennon, Apress, 2009*
- CouchDB - kurz & gut ,Mario Scheliga, 2010 , O'Reilly
- planet.couchdb.org
- blog.couch.io/
- www.couchone.com/get
- Anwendungen mit CouchDB, JavaScript, HTML5: couchapp.org
- What's New in CouchDB 1.0 www.youtube.com/watch?v=4jtBQf41Ppc
- CouchDB, Database Pro 05/ 09, Frank Pientka

12.–15.09.2010
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Vielen Dank!

Frank Pientka

Materna GmbH, Dortmund