CouchDB: JSON, HTTP & MapReduce

Bradley Holt (http://bradley-holt.com/)
@BradleyHolt (http://twitter.com/BradleyHolt)
About Me
Co-Founder and Technical Director
from Vermont
CouchDB Basics
Cluster Of Unreliable Commodity Hardware

Document-oriented, schema-less

Shared nothing, horizontally scalable

Runs on the Erlang OTP platform

Peer-based, bi-directional replication

RESTful HTTP API

Queries are done against MapReduce “views”, or indexes
When You Might Consider CouchDB

You’ve found yourself denormalizing your SQL database for better performance.

Your domain model is a “fit” for documents (e.g. a CMS).

Your application is read-heavy.

You need a high level of concurrency and can give up consistency in exchange.

You need horizontal scalability.

You want your database to be able to run anywhere, even on mobile devices, and even when disconnected from the cluster.
Trade-Offs

No ad hoc queries. You need to know what you’re going to want to query ahead of time. For example, SQL would be a better fit for business intelligence reporting.

No concept of “joins”. You can relate data, but watch out for consistency issues.

Transactions are limited to document boundaries.

CouchDB trades storage space for performance.
Other Alternatives to SQL

MongoDB
http://www.mongodb.org/

Redis
http://redis.io/

Cassandra
http://cassandra.apache.org/

Riak
http://www.basho.com/

HBase (a database for Hadoop)
http://hbase.apache.org/
Don’t be so quick to get rid of SQL! There are *many* problems for which an SQL database is a good fit. SQL is very powerful and flexible query language.
JSON Documents

{"title":"CouchDB: The Definitive Guide"}
JSON (JavaScript Object Notation) is a human-readable and lightweight data interchange format.

Data structures from many programming languages can be easily converted to and from JSON.
A JSON object is a collection of key/value pairs.

JSON values can be strings, numbers, booleans (false or true), arrays (e.g. ["a", "b", "c"]), null, or another JSON object.
A “Book” JSON Object

```json
{
  "_id":"978-0-596-15589-6",
  "title":"CouchDB: The Definitive Guide",
  "subtitle":"Time to Relax",
  "authors": [
    "J. Chris Anderson",
    "Jan Lehnardt",
    "Noah Slater"
  ],
  "publisher":"O'Reilly Media",
  "released":"2010-01-19",
  "pages":272
}
```
RESTful HTTP API

curl -iX PUT http://localhost:5984/mydb
Representational State Transfer (REST) is a software architecture style that describes distributed hypermedia systems such as the World Wide Web.
HTTP is distributed, scalable, and cacheable.
Everyone speaks HTTP.
Create a Database

$ curl -iX PUT http://localhost:5984/mydb

HTTP/1.1 201 Created
Location: http://localhost:5984/mydb

{"ok":true}
Create a Document

$ curl -iX POST http://localhost:5984/mydb
-H "Content-Type: application/json"
-d '{"_id":"42621b2516001626"}'

HTTP/1.1 201 Created
Location: http://localhost:5984/mydb/42621b2516001626

{
    "ok":true,
    "id":"42621b2516001626",
    "rev":"1-967a00df5e02add41819138abb3284d"
}
Read a Document

$ curl -iX GET http://localhost:5984/mydb/42621b2516001626

HTTP/1.1 200 OK
Etag: "1-967a00dff5e02add41819138abb3284d"

{
    "_id":"42621b2516001626",
    "_rev":"1-967a00dff5e02add41819138abb3284d"
}
When updating a document, CouchDB requires the correct document revision number as part of its *Multi-Version Concurrency Control* (MVCC). This form of *optimistic concurrency* ensures that another client hasn't modified the document since it was last retrieved.
Update a Document

$ curl -iX PUT http://localhost:5984/mydb/42621b2516001626
-H "Content-Type: application/json"
-d '{
  "_id":"42621b2516001626",
  "_rev":"1-967a00dff5e02add41819138abb3284d",
  "title":"CouchDB: The Definitive Guide"
}'

HTTP/1.1 201 Created

{
  "ok":true,
  "id":"42621b2516001626",
  "rev":"2-bbd27429fd1a0daa2b946cbacb22dc3e"
}
Conditional GET

$ curl -iX GET http://localhost:5984/mydb/42621b2516001626
-H 'If-None-Match: "2-bbd27429fd1a0daa2b946cbacbc22dc3e"

HTTP/1.1 304 Not Modified
Etag: "2-bbd27429fd1a0daa2b946cbacbc22dc3e"
Content-Length: 0
Delete a Document

$ curl -iX DELETE http://localhost:5984/mydb/42621b2516001626
-H 'If-Match: "2-bbd27429fd1a0daa2b946cbacb22dc3e"

HTTP/1.1 200 OK

{
  "ok":true,
  "id":"42621b2516001626",
  "rev":"3-29d2ef6e0d3558a3547a92dac51f3231"
}
Read a Deleted Document

$ curl -iX GET http://localhost:5984/mydb/42621b2516001626

HTTP/1.1 404 Object Not Found

{
    "error":"not_found",
    "reason":"deleted"
}
Read a Deleted Document

$ curl -iX GET http://localhost:5984/mydb/42621b2516001626
?rev=3-29d2ef6e0d3558a3547a92dac51f3231

HTTP/1.1 200 OK
Etag: "3-29d2ef6e0d3558a3547a92dac51f3231"

{
  "_id":"42621b2516001626",
  "_rev":"3-29d2ef6e0d3558a3547a92dac51f3231",
  "_deleted":true
}
Fetch Revisions

$ curl -iX GET http://localhost:5984/mydb/42621b2516001626?rev=3-29d2ef6e0d3558a3547a92dac51f3231&revs=true

HTTP/1.1 200 OK

{ // …
  "_revisions":{
    "start":3,
    "ids":[
      "29d2ef6e0d3558a3547a92dac51f3231",
      "bbd27429fd1a0daa2b946cbacb22dc3e",
      "967a00dff5e02add41819138abb3284d"
    ]
  }
}
Do not rely on older revisions. Revisions are used for concurrency control and for replication. Old revisions are removed during database compaction.
MapReduce Views

function(doc) { if (doc.title) { emit(doc.title); } }
MapReduce consists of *Map* and *Reduce* steps which can be distributed in a way that takes advantage of the multiple processor cores found in modern hardware.
## Overview

### Create Database...

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Number of Documents</th>
<th>Update Seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>_users</td>
<td>4.1 KB</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Showing 1-1 of 1 databases
Create a Database
Map and Reduce are written as JavaScript functions that are defined within views. *Temporary views* can be used during development but should be saved permanently to *design documents* for production. Temporary views can be very slow.
Add the First Document

{
    "_id": "978-0-596-15589-6",
    "title": "CouchDB: The Definitive Guide",
    "subtitle": "Time to Relax",
    "authors": [
        "J. Chris Anderson",
        "Jan Lehnardt",
        "Noah Slater"
    ],
    "publisher": "O'Reilly Media",
    "released": "2010-01-19",
    "pages": 272
}
One-To-One Mapping
Map Book Titles

```javascript
function(doc) { // JSON object representing a doc to be mapped
    if (doc.title) { // make sure this doc has a title
        emit(doc.title); // emit the doc’s title as the key
    }
}
```
The emit function accepts two arguments. The first is a key and the second a value. Both are optional and default to null.
“Titles” Temporary View

View Code

function(doc) {
    if (doc.title) {
        emit(doc.title);
    }
}

Warning: Please note that temporary views are not suitable for use in production, as they are really slow for any database with more than a few dozen documents. You can use a temporary view to experiment with view functions, but switch to a permanent view before using them in an application.

Key | Value
--- | ---
"CouchDB: The Definitive Guide" | null
ID: 978-0-596-15589-6 |
Showing 1-1 of 1 row

--- Previous Page | Rows per page: 10 | Next Page ---
### “Titles” Temporary View

<table>
<thead>
<tr>
<th>key</th>
<th>id</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;CouchDB: The Definitive Guide&quot;</td>
<td>&quot;978-0-596-15589-6&quot;</td>
<td>null</td>
</tr>
</tbody>
</table>
Add a Second Document

{
  "_id": "978-0-596-52926-0",
  "title": "RESTful Web Services",
  "subtitle": "Web services for the real world",
  "authors": [
    "Leonard Richardson",
    "Sam Ruby"
  ],
  "publisher": "O'Reilly Media",
  "released": "2007-05-08",
  "pages": 448
}
# "Titles" Temporary View

[Image of a webpage showing a view code and a table with two rows.]

---

**View Code**

```javascript
function(doc) {
  if (doc.title) {
    emit(doc.title);
  }
}
```

**Warning:** Please note that temporary views are not suitable for use in production, as they are really slow for any database with more than a few dozen documents. You can use a temporary view to experiment with view functions, but switch to a permanent view before using them in an application.

---

### Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;RESTful Web Services&quot;</td>
<td>null</td>
</tr>
<tr>
<td>ID: 978-0-596-52926-0</td>
<td></td>
</tr>
<tr>
<td>&quot;CouchDB: The Definitive Guide&quot;</td>
<td>null</td>
</tr>
<tr>
<td>ID: 978-0-596-15589-6</td>
<td></td>
</tr>
</tbody>
</table>

Showing 1-2 of 2 rows
**“Titles” Temporary View**

<table>
<thead>
<tr>
<th>key</th>
<th>id</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;CouchDB: The Definitive Guide&quot;</td>
<td>&quot;978-0-596-15589-6&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;RESTful Web Services&quot;</td>
<td>&quot;978-0-596-52926-0&quot;</td>
<td>null</td>
</tr>
</tbody>
</table>
One-To-Many Mapping
Add a “formats” Field to Both Documents

{
  // ...
  "formats": [
    "Print",
    "Ebook",
    "Safari Books Online"
  ]
}
Add a Third Document

{
   "id":"978-1-565-92580-9",
   "title":"DocBook: The Definitive Guide",
   "authors":[
      "Norman Walsh",
      "Leonard Muellner"
   ],
   "publisher":"O'Reilly Media",
   "formats":[
      "Print"
   ],
   "released":"1999-10-28",
   "pages":648
}
function(doc) { // JSON object representing a doc to be mapped
    if (doc.formats) { // make sure this doc has a formats field
        for (var i in doc.formats) {
            emit(doc.formats[i]); // emit each format as the key
        }
    }
}
function(doc) {
    if (docformats) {
        for (i in docformats) {
            emit(docformats[i]);
        }
    }
}

**Warning**: Please note that temporary views are not suitable for use in production, as they are really slow for any database with more than a few dozen documents. You can use a temporary view to experiment with view functions, but switch to a permanent view before using them in an application.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Safari Books Online&quot;</td>
<td>null</td>
</tr>
<tr>
<td>ID: 978-0-596-52926-0</td>
<td></td>
</tr>
<tr>
<td>&quot;Safari Books Online&quot;</td>
<td>null</td>
</tr>
<tr>
<td>ID: 978-0-596-15589-6</td>
<td></td>
</tr>
<tr>
<td>&quot;Print&quot;</td>
<td>null</td>
</tr>
<tr>
<td>ID: 978-1-565-92580-9</td>
<td></td>
</tr>
<tr>
<td>&quot;Print&quot;</td>
<td>null</td>
</tr>
<tr>
<td>ID: 978-0-596-52926-0</td>
<td></td>
</tr>
<tr>
<td>&quot;Print&quot;</td>
<td>null</td>
</tr>
<tr>
<td>ID: 978-0-596-15589-6</td>
<td></td>
</tr>
<tr>
<td>&quot;Ebook&quot;</td>
<td>null</td>
</tr>
<tr>
<td>ID: 978-0-596-52926-0</td>
<td></td>
</tr>
<tr>
<td>&quot;Ebook&quot;</td>
<td>null</td>
</tr>
<tr>
<td>ID: 978-0-596-15589-6</td>
<td></td>
</tr>
</tbody>
</table>
## “Formats” Temporary View

<table>
<thead>
<tr>
<th>key</th>
<th>id</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Ebook&quot;</td>
<td>&quot;978-0-596-15589-6&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;Ebook&quot;</td>
<td>&quot;978-0-596-52926-0&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;Print&quot;</td>
<td>&quot;978-0-596-15589-6&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;Print&quot;</td>
<td>&quot;978-0-596-52926-0&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;Print&quot;</td>
<td>&quot;978-1-565-92580-9&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;Safari Books Online&quot;</td>
<td>&quot;978-0-596-15589-6&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;Safari Books Online&quot;</td>
<td>&quot;978-0-596-52926-0&quot;</td>
<td>null</td>
</tr>
</tbody>
</table>
When querying a view, the “key” and “id” fields can be used to select a row or range of rows. Optionally, rows can be grouped by “key” fields or by levels of “key” fields.
Map Book Authors

function(doc) {  // JSON object representing a doc to be mapped
    if (doc.authors) {  // make sure this doc has an authors field
        for (var i in doc.authors) {
            emit(doc.authors[i]);  // emit each author as the key
        }
    }
}
Warning: Please note that temporary views are not suitable for use in production, as they are really slow for any database with more than a few dozen documents. You can use a temporary view to experiment with view functions, but switch to a permanent view before using them in an application.
## “Authors” Temporary View

<table>
<thead>
<tr>
<th>key</th>
<th>id</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;J. Chris Anderson&quot;</td>
<td>&quot;978-0-596-15589-6&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;Jan Lehnardt&quot;</td>
<td>&quot;978-0-596-15589-6&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;Leonard Muellner&quot;</td>
<td>&quot;978-1-565-92580-9&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;Leonard Richardson&quot;</td>
<td>&quot;978-0-596-52926-0&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;Noah Slater&quot;</td>
<td>&quot;978-0-596-15589-6&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;Norman Walsh&quot;</td>
<td>&quot;978-1-565-92580-9&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;Sam Ruby&quot;</td>
<td>&quot;978-0-596-52926-0&quot;</td>
<td>null</td>
</tr>
</tbody>
</table>
Reduce
## Built-in Reduce Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>_count</code></td>
<td>Returns the number of mapped values in the set</td>
</tr>
<tr>
<td><code>_sum</code></td>
<td>Returns the sum of the set of mapped values</td>
</tr>
<tr>
<td><code>_stats</code></td>
<td>Returns numerical statistics of the mapped values in the set including the sum, count, min, and max</td>
</tr>
</tbody>
</table>
Count
Format Count, Not Grouped

```
function(doc) {
  if (docformats) {
    for (var i in docformats) {
      emit(docformats[i]);
    }
  }
}
```

Warning: Please note that temporary views are not suitable for use in production, as they are really slow for any database with more than a few dozen documents. You can use a temporary view to experiment with view functions, but switch to a permanent view before using them in an application.

```
null 7
```

Showing 1-1 of unknown rows
Format Count, Not Grouped

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>7</td>
</tr>
</tbody>
</table>
Format Count, Grouped

Map Function:
```javascript
function(doc) {
  if (doc.formats) {
    for (var i in doc.formats) {
      emit(doc.formats[i]);
    }
  }
}
```

Reduce Function (optional):
```
_count
```

**Warning:** Please note that temporary views are not suitable for use in production, as they are really slow for any database with more than a few dozen documents. You can use a temporary view to experiment with view functions, but switch to a permanent view before using them in an application.

**Key**
- "Safari Books Online": 2
- "Print": 3
- "Ebook": 2

Showing 1-3 of unknown rows
## Format Count, Grouped

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Ebook&quot;</td>
<td>2</td>
</tr>
<tr>
<td>&quot;Print&quot;</td>
<td>3</td>
</tr>
<tr>
<td>&quot;Safari Books Online&quot;</td>
<td>2</td>
</tr>
</tbody>
</table>
Sum of Pages by Format
## Sum of Pages by Format

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Ebook&quot;</td>
<td>720</td>
</tr>
<tr>
<td>&quot;Print&quot;</td>
<td>1368</td>
</tr>
<tr>
<td>&quot;Safari Books Online&quot;</td>
<td>720</td>
</tr>
</tbody>
</table>
Stats

sum, count, minimum, maximum, sum over all square roots
Stats of Pages by Format

```javascript
function(doc) {
    if (docformats) {
        for (var i in docformats) {
            emit(docformats[i], doc.pages);
        }
    }
}
```

**Warning:** Please note that temporary views are not suitable for use in production, as they are really slow for any database with more than a few dozen documents. You can use a temporary view to experiment with view functions, but switch to a permanent view before using them in an application.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Safari Books Online&quot;</td>
<td>{sum: 720, count: 2, min: 272, max: 448, sumsqr: 274688}</td>
</tr>
<tr>
<td>&quot;Print&quot;</td>
<td>{sum: 1368, count: 3, min: 272, max: 648, sumsqr: 694592}</td>
</tr>
<tr>
<td>&quot;Ebook&quot;</td>
<td>{sum: 720, count: 2, min: 272, max: 448, sumsqr: 274688}</td>
</tr>
</tbody>
</table>

Showing 1-3 of unknown rows

---

Welcome to Admin Party!
Everyone is admin. Fix this

Futon on Apache CouchDB 1.0.1
# Stats of Pages by Format

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Print&quot;</td>
<td>{&quot;sum&quot;:1368,&quot;count&quot;:3,&quot;min&quot;:272,&quot;max&quot;:648,&quot;sumsqr&quot;:694592}</td>
</tr>
<tr>
<td>&quot;Safari Books Online&quot;</td>
<td>{&quot;sum&quot;:720,&quot;count&quot;:2,&quot;min&quot;:272,&quot;max&quot;:448,&quot;sumsqr&quot;:274688}</td>
</tr>
</tbody>
</table>
Custom Reduce Functions
The built-in Reduce functions should serve your needs most, if not all, of the time. If you find yourself writing a custom Reduce function, then take a step back and make sure that one of the built-in Reduce functions won't serve your needs better.
Reduce Function Skeleton

```javascript
function(keys, values, rereduce) {
}
```
Parameters

Keys: An array of mapped key and document IDs in the form of [key,id] where id is the document ID.

Values: An array of mapped values.

Rereduce: Whether or not the Reduce function is being called recursively on its own output.
Count Equivalent

```javascript
function(keys, values, rereduce) {
    if (rereduce) {
        return sum(values);
    } else {
        return values.length;
    }
}
```
function(keys, values, rereduce) {
    return sum(values);
}
MapReduce Limitations

Full-text indexing and ad hoc searching

- couchdb-lucene
  https://github.com/rnewson/couchdb-lucene
- ElasticSearch and CouchDB
  https://github.com/elasticsearch/elasticsearch/wiki/Couchdb-integration

Geospatial indexing and search (two dimensional)

- GeoCouch
  https://github.com/vmx/couchdb
- Geohash (e.g. dr5rusx1qkvvr)
  http://geohash.org/
Querying Views
You can query for all rows, a single contiguous range of rows, or even rows matching a specified key.
Add a Fourth Document

{
  "_id": "978-0-596-80579-1",
  "title": "Building iPhone Apps with HTML, CSS, and JavaScript",
  "subtitle": "Making App Store Apps Without Objective-C or Cocoa",
  "authors": [
    "Jonathan Stark"
  ],
  "publisher": "O'Reilly Media",
  "formats": [
    "Print",
    "Ebook",
    "Safari Books Online"
  ],
  "released": "2010-01-08",
  "pages": 192
}
Map Book Releases

```javascript
function(doc) {
  if (doc.released) {
    emit(doc.released.split("-")), doc.pages);
  }
}
```
Save the “Releases” View
“Releases”, Exact Grouping
"Releases", Exact Grouping

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>[&quot;2007&quot;,&quot;05&quot;,&quot;08&quot;]</td>
<td>{&quot;sum&quot;:448,&quot;count&quot;:1,&quot;min&quot;:448,&quot;max&quot;:448,&quot;sumsqr&quot;:200704}</td>
</tr>
<tr>
<td>[&quot;2010&quot;,&quot;01&quot;,&quot;08&quot;]</td>
<td>{&quot;sum&quot;:192,&quot;count&quot;:1,&quot;min&quot;:192,&quot;max&quot;:192,&quot;sumsqr&quot;:36864}</td>
</tr>
<tr>
<td>[&quot;2010&quot;,&quot;01&quot;,&quot;19&quot;]</td>
<td>{&quot;sum&quot;:272,&quot;count&quot;:1,&quot;min&quot;:272,&quot;max&quot;:272,&quot;sumsqr&quot;:73984}</td>
</tr>
</tbody>
</table>
“Releases”, Level 1 Grouping
**“Releases”, Level 1 Grouping**

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;1999&quot;</td>
<td>{&quot;sum&quot;:648,&quot;count&quot;:1,&quot;min&quot;:648,&quot;max&quot;:648,&quot;sumsqr&quot;:419904}</td>
</tr>
<tr>
<td>&quot;2007&quot;</td>
<td>{&quot;sum&quot;:448,&quot;count&quot;:1,&quot;min&quot;:448,&quot;max&quot;:448,&quot;sumsqr&quot;:200704}</td>
</tr>
<tr>
<td>&quot;2010&quot;</td>
<td>{&quot;sum&quot;:464,&quot;count&quot;:2,&quot;min&quot;:192,&quot;max&quot;:272,&quot;sumsqr&quot;:110848}</td>
</tr>
</tbody>
</table>
“Releases”, Level 2 Grouping

```javascript
function(doc) {
    if (doc.released) {
        emit(doc.released.split('-'), doc.pages);
    }
}
```

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>[&quot;2010&quot;, &quot;01&quot;]</td>
<td><code>{sum: 464, count: 2, min: 192, max: 272, sumsq: 110848}</code></td>
</tr>
<tr>
<td>[&quot;2007&quot;, &quot;05&quot;]</td>
<td><code>{sum: 448, count: 1, min: 448, max: 448, sumsq: 200704}</code></td>
</tr>
</tbody>
</table>
### “Releases”, Level 2 Grouping

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>[&quot;1999&quot;,&quot;10&quot;]</td>
<td>{&quot;sum&quot;:648,&quot;count&quot;:1,&quot;min&quot;:648,&quot;max&quot;:648,&quot;sumsqr&quot;:419904}</td>
</tr>
<tr>
<td>[&quot;2007&quot;,&quot;05&quot;]</td>
<td>{&quot;sum&quot;:448,&quot;count&quot;:1,&quot;min&quot;:448,&quot;max&quot;:448,&quot;sumsqr&quot;:200704}</td>
</tr>
<tr>
<td>[&quot;2010&quot;,&quot;01&quot;]</td>
<td>{&quot;sum&quot;:464,&quot;count&quot;:2,&quot;min&quot;:192,&quot;max&quot;:272,&quot;sumsqr&quot;:110848}</td>
</tr>
</tbody>
</table>
PHP Libraries for CouchDB

PHPCouch
http://www.phpcouch.org/

PHPillow
http://arbitracker.org/phpillow.html

PHP CouchDB Extension (PECL)
http://www.topdog.za.net/php_couchdb_extension

Doctrine2 ODM
Do-It-Yourself

Zend_Http_Client + Zend_Cache

PEAR’s HTTP_Client

PHP cURL

JSON maps nicely to and from PHP arrays using Zend_Json or PHP’s json_ functions
Scaling
Load Balancing

Send POST, PUT, and DELETE requests to a write-only master node

Setup continuous replication from the master node to multiple read-only nodes

Load balance GET, HEAD, and OPTIONS requests amongst read-only nodes

- Apache HTTP Server (mod_proxy)
- nginx
- HAProxy
- Varnish
- Squid
Clustering (Partitioning/Sharding)

Lounge
http://tilgovi.github.com/couchdb-lounge/
  • Proxy, partitioning, and sharding

BigCouch
https://github.com/cloudant/bigcouch
  • Clusters modeled after Amazon’s Dynamo approach

Pillow
https://github.com/khellan/Pillow
  • “…a combined router and rereducer for CouchDB.”
What else?
Authentication

Basic Authentication

Cookie Authentication

OAuth
Authorization

Server Admin

Database Reader

Document Update Validation
http://wiki.apache.org/couchdb/Document_Update_Validation
Hypermedia Controls

Show Functions

List Functions

See:
http://wiki.apache.org/couchdb/Formatting_with_Show_and_List
Replication

Peer-based & bi-directional

Documents and modified fields are incrementally replicated.

All data will be *eventually consistent*.

Conflicts are flagged and can be handled by application logic.

Partial replicas can be created via JavaScript filter functions.
Hosting

CouchOne (now CouchBase)
http://www.couchone.com/

Cloudant
https://cloudant.com/
Distributing

CouchApp
http://couchapp.org/
  • Applications built using JavaScript, HTML5, CSS and CouchDB

Mobile
• Android
  http://www.couchone.com/android
CouchDB Resources

*CouchDB: The Definitive Guide*
by J. Chris Anderson, Jan Lehnardt, and Noah Slater
(O’Reilly)
978-0-596-15589-6

*Writing and Querying MapReduce Views in CouchDB*
by Bradley Holt (O’Reilly)
978-1-449-30312-9

*Scaling CouchDB*
by Bradley Holt (O’Reilly)
063-6-920-01840-7

CouchDB Wiki
http://wiki.apache.org/couchdb/

*Beginning CouchDB*
by Joe Lennon (Apress)
978-1-430-27237-3
Questions?
Thank You

Bradley Holt (http://bradley-holt.com/)
@BradleyHolt (http://twitter.com/BradleyHolt)