

CouchDB and GeoCouch

Volker Mische

Erlang Factory Lite Munich
January 31st 2011
Munich

About me

- Volker "vmx" Mische, volker@couchone.com, @vmische
- Creator of GeoCouch
- Open source developer, Erlang and JavaScript fan
- Proponent of all things open and geo
- Raver



Document

```
1 {
2   "_id": "Station-2942_001",
3   "_rev": "3-77f17a55f6ab11f7f6668e63a75f2281",
4
5   "name": "Station-2942",
6   "date": "2011-01-31",
7   "loc": [48.136944, 11.575278],
8   "temperature": -3,
9   "rainfall": 5.0
10 }
```

RESTful HTTP API

The four basic functions of a database (CRUD):

Create HTTP **PUT** /db/Station-2942_001

Read HTTP **GET** /db/Station-2942_001

Update HTTP **PUT** /db/Station-2942_001

Delete HTTP **DELETE** /db/Station-2942_001

Views

MapReduce

Map function

```
1 function(doc) {  
2   // emit(keys, value)  
3   emit(...);  
4 }
```


Map function

```
1 function(doc) {  
2   // emit(keys, value)  
3   emit([doc.name, doc.date], doc.rainfall);  
4 }
```

Keys	Value
...	...
Station-1873, 2009-10-19	2
Station-1873, 2009-10-20	0
Station-2942, 2009-07-08	13
Station-2942, 2009-07-09	14
Station-2942, 2009-07-10	17
...	...

Accessing the View

http://localhost:5984/db/_design/stations/_view/rainfall

Keys	Value
...	...
Station-1873, 2009-10-19	2
Station-1873, 2009-10-20	0
Station-2942, 2009-07-08	13
Station-2942, 2009-07-09	14
Station-2942, 2009-07-10	17
...	...

Accessing the View

“Get the rainfall from 8–9 July 2009 of Station-2942”

Keys	Value
...	...
Station-1873, 2009-10-19	2
Station-1873, 2009-10-20	0
Station-2942, 2009-07-08	13
Station-2942, 2009-07-09	14
Station-2942, 2009-07-10	17
...	...

Accessing the View

“Get the rainfall from 8–9 July 2009 of Station-2942”

`http://localhost:5984/db/_design/stations/_view/rainfall?`

`startkey=["Station-2942", "2009-07-08"]&`

`endkey=["Station-2942", "2009-07-09"]`

Keys	Value
...	...
Station-1873, 2009-10-19	2
Station-1873, 2009-10-20	0
Station-2942, 2009-07-08	13
Station-2942, 2009-07-09	14
Station-2942, 2009-07-10	17
...	...

Reduce

Reduce

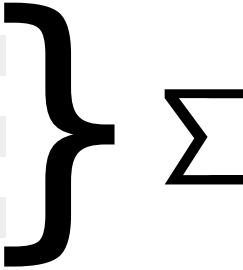
```
1 function(keys, values) {
2   var sum = 0;
3   for each(var val in values) {
4     sum = sum + val;
5   }
6   return sum;
7 }
```

Reduce result

Keys	Value
...	...
Station-1873, 2009-10-19	2
Station-1873, 2009-10-20	0
Station-2942, 2009-07-08	13
Station-2942, 2009-07-09	14
Station-2942, 2009-07-10	17
...	...

Reduce result

Keys	Value
...	...
Station-1873, 2009-10-19	2
Station-1873, 2009-10-20	0
Station-2942, 2009-07-08	13
Station-2942, 2009-07-09	14
Station-2942, 2009-07-10	17
...	...



Reduce result

Keys	Value
...	...
Station-1873, 2009-10-19	2
Station-1873, 2009-10-20	0
Station-2942, 2009-07-08	13
Station-2942, 2009-07-09	14
Station-2942, 2009-07-10	17
...	...

} 27

Fancy Features

Schema Free

Schema Free

```
{
  "id": "Station-2942_001",
  "_rev": "3-77f17a55f6ab11f7f6668e63a75f2281",

  "name": "Station-2942",
  "date": "2011-01-31",
  "loc": [48.136944, 11.575278],
  "temperature": -3,
  "rainfall": 5.0
}
```

Schema Free

```
{  
  "id": "Station-2942_001",  
  "_rev": "3-77f17a55f6ab11f7f6668e63a75f2281",  
  
  "name": "Station-2942",  
  "date": "2011-01-31",  
  "loc": [48.136944, 11.575278],  
  "temperature": -3,  
  "rainfall": 5.0  
  "atmospheric_pressure": 1012  
}
```

Arbitrary input/output

JSON + JavaScript
=
anything

Append-only data structure (Example)

File

**Connection
to existing
Nodes**

**Actual
Tree**

Append-only data structure (Example)

File

**Connection
to existing
Nodes**

**Actual
Tree**



**Position of
the root node**



Newly inserted node




Other nodes

Append-only data structure (Example)

File  1

Connection
to existing
Nodes 

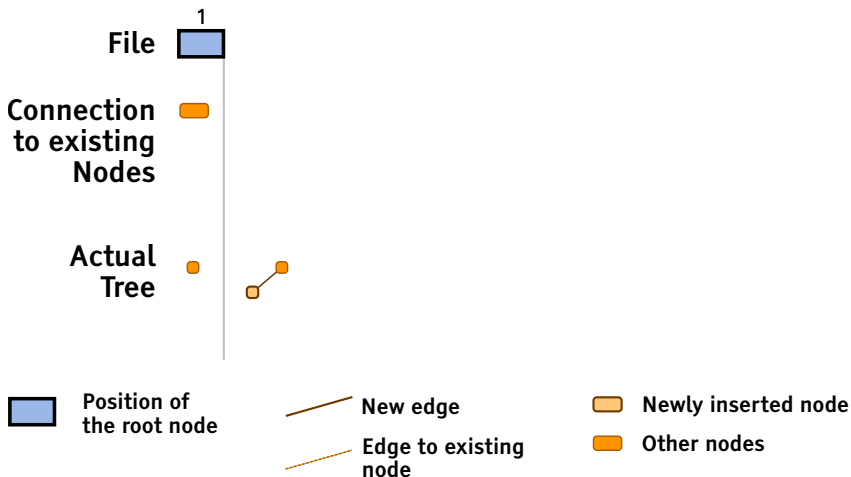
Actual
Tree 

 Position of
the root node

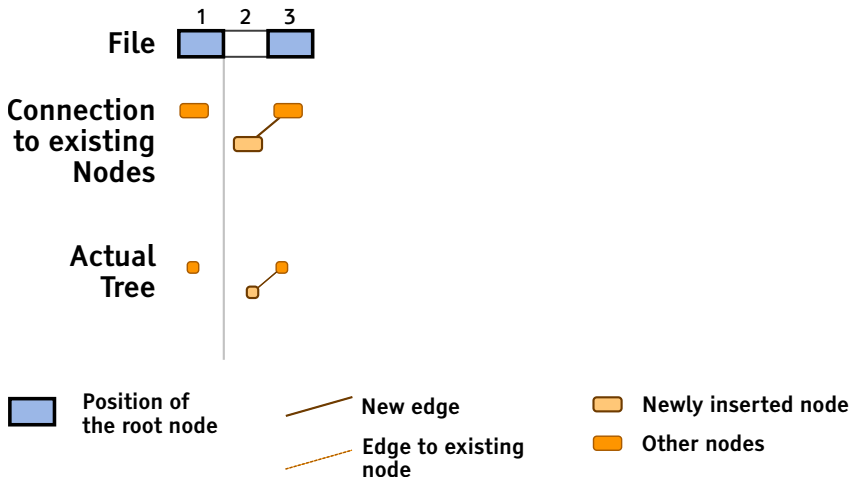
 Newly inserted node

 Other nodes

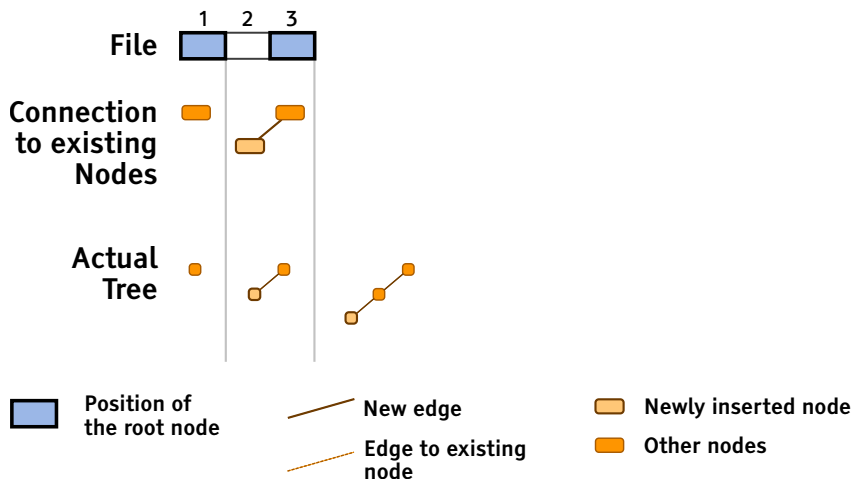
Append-only data structure (Example)



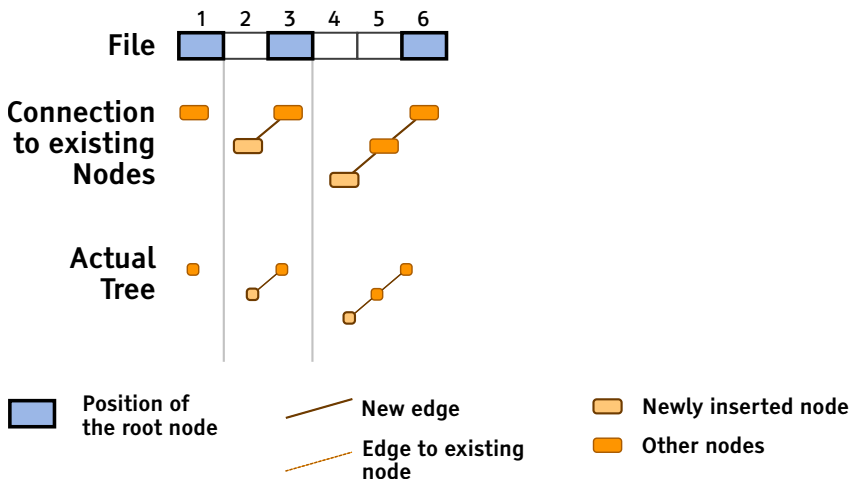
Append-only data structure (Example)



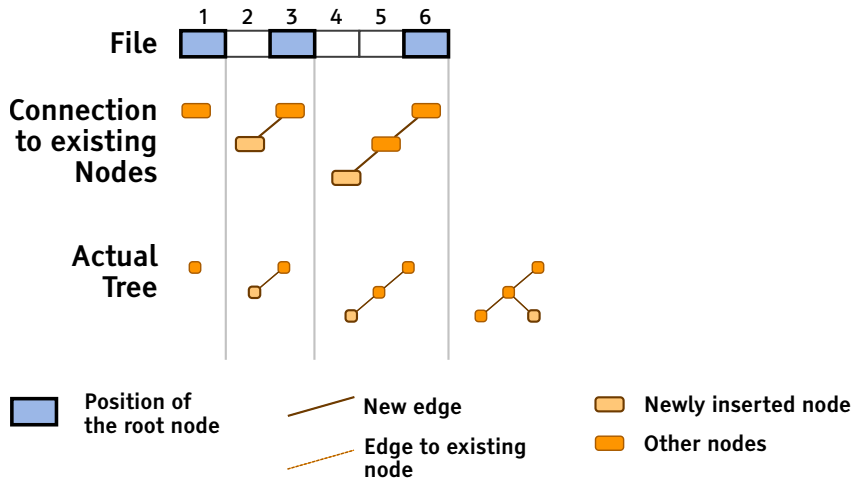
Append-only data structure (Example)



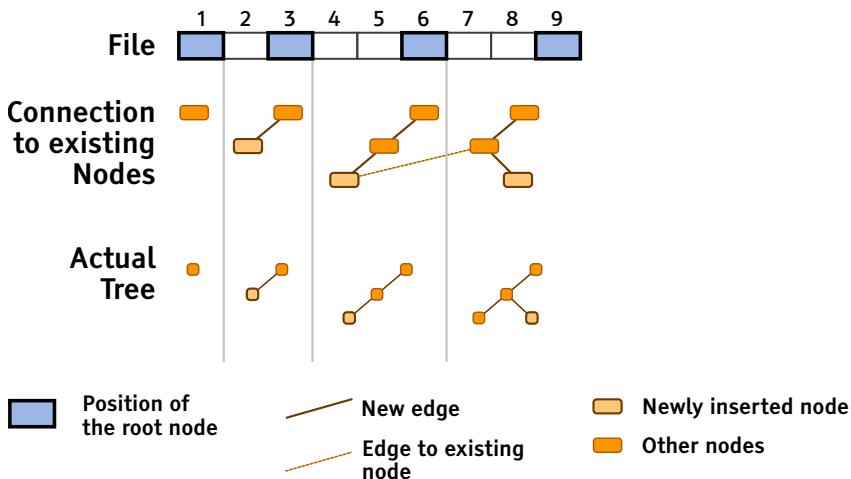
Append-only data structure (Example)



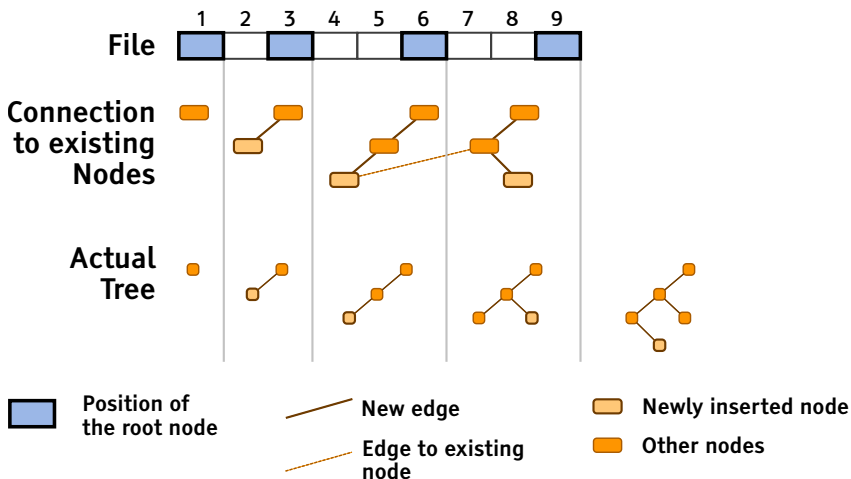
Append-only data structure (Example)



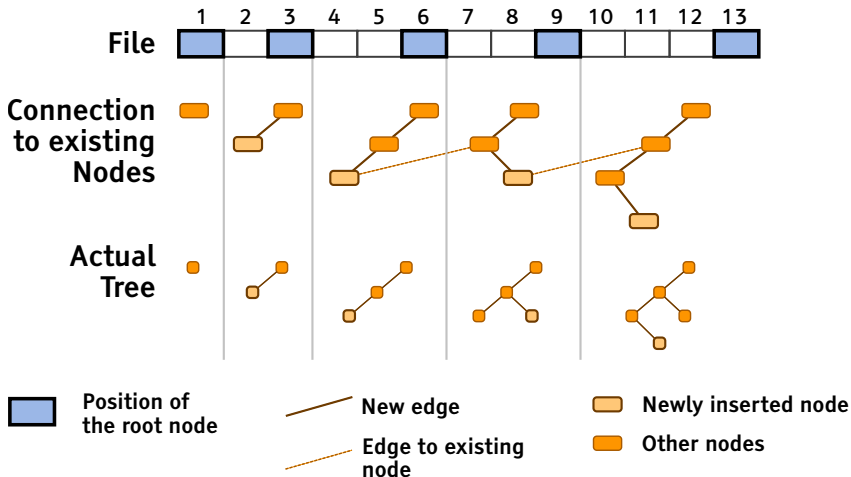
Append-only data structure (Example)



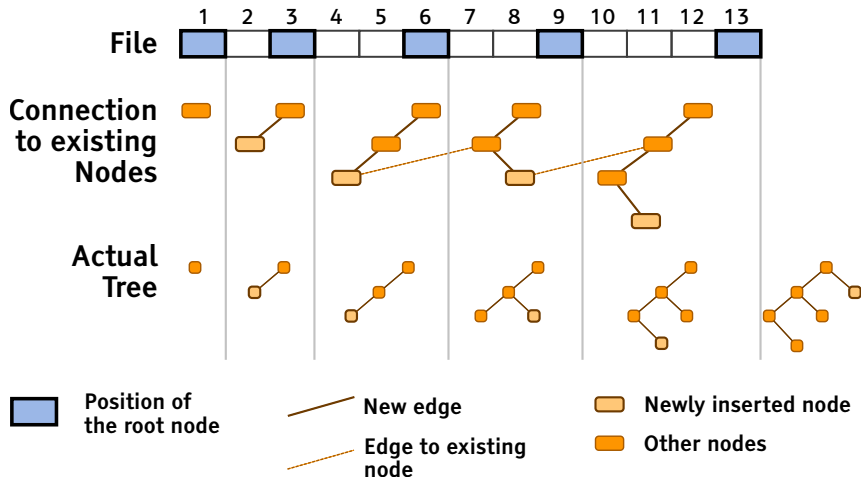
Append-only data structure (Example)



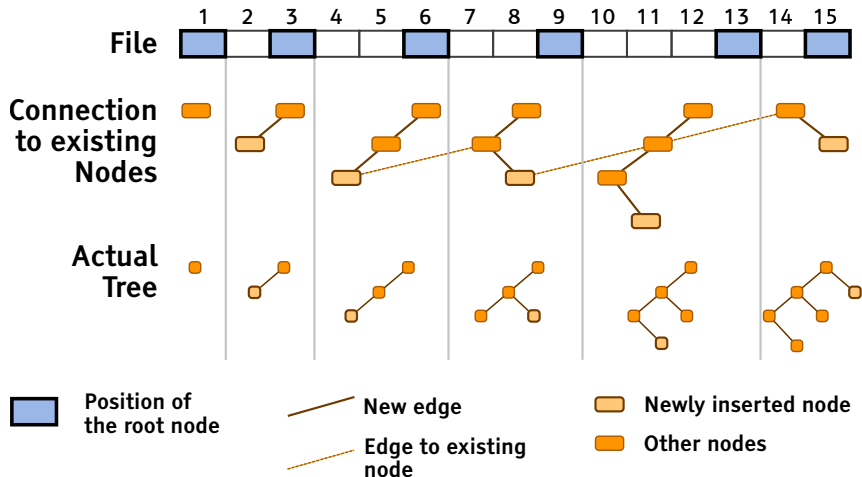
Append-only data structure (Example)



Append-only data structure (Example)



Append-only data structure (Example)



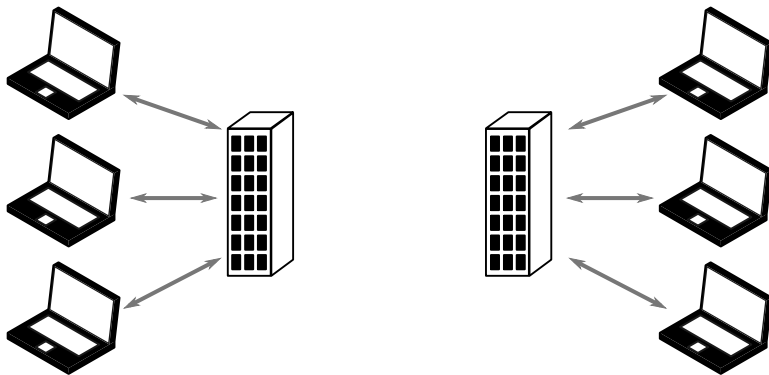
Concurrency/Robustness

- Erlang
- Append-only
- Durability (crash-only design)
- No lock on reads (MVCC)

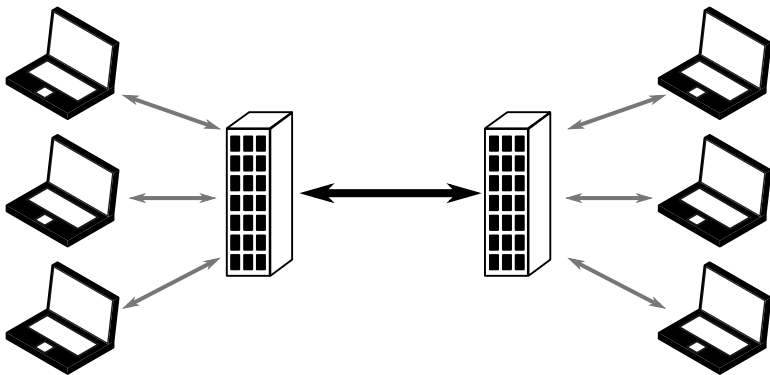
Replication



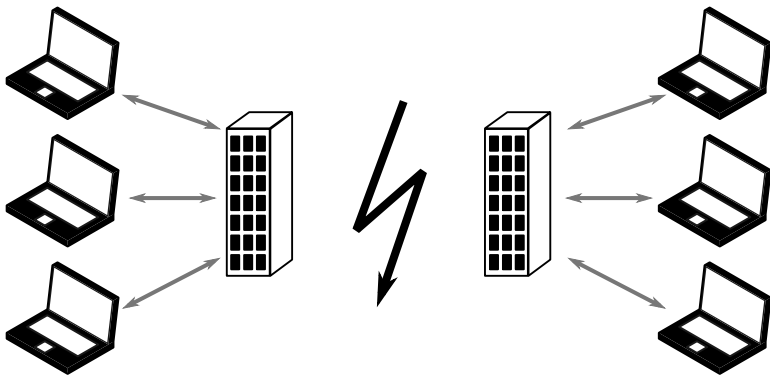
Replication



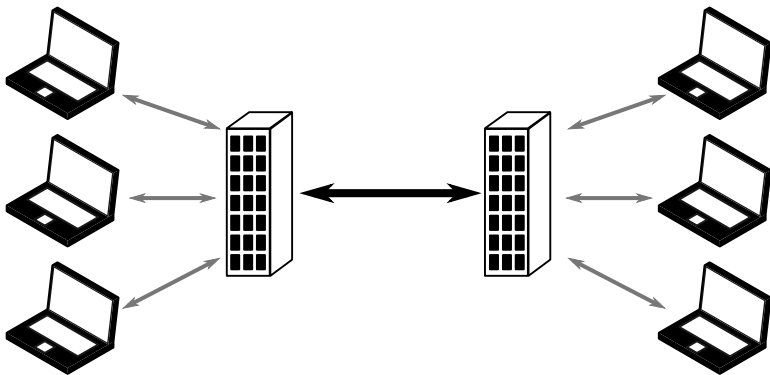
Replication



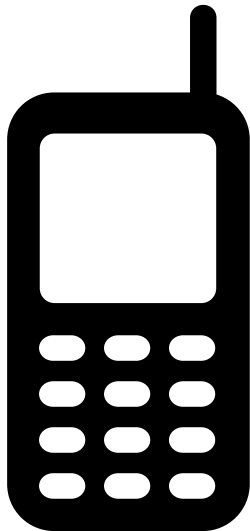
Replication



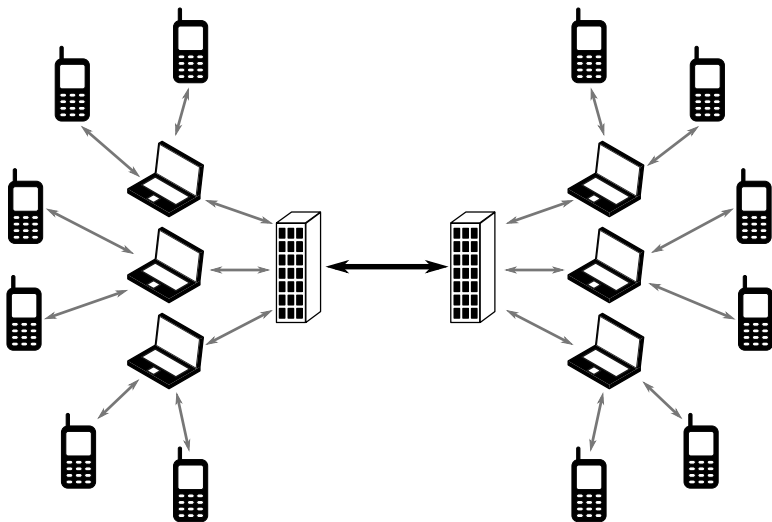
Replication



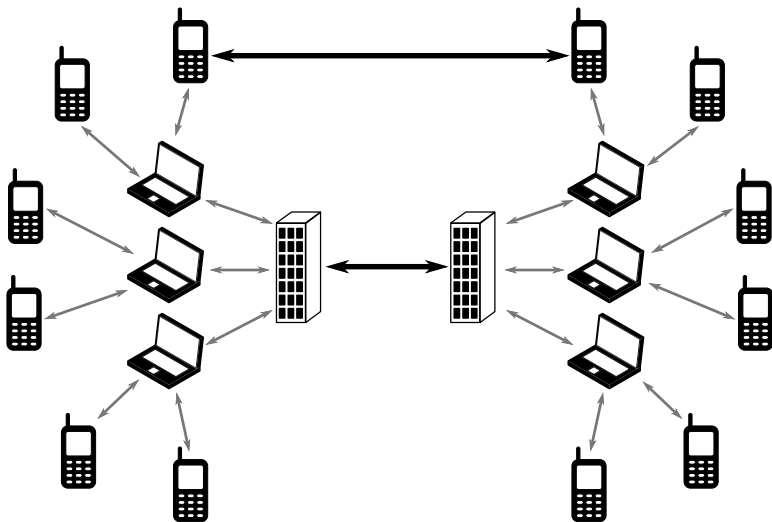
Replication



Replication



Replication





Seth Sawyers (CC-BY/2.0) (verändert)







GeoCouch

What is GeoCouch?

A spatial index for CouchDB

Implementation

- Completely written in Erlang
- No additional dependencies
- R-Tree
- Spatial index is independent of view index

GeoJSON

- Emerged as simple geo standard for the web
- Widely used
- Points, Lines, Polygons (all are supported by GeoCouch)
- Example:

```
{  
  "type": "Point",  
  "coordinates": [10.898333,48.371667]  
}
```

Spatial Index Function

```
1 function(doc) {  
2  
3  
4  
5  
6  
7  
8 }
```

Spatial Index Function

```
1 function(doc) {  
2     if (doc.loc) {  
3  
4  
5  
6  
7     }  
8 }
```

Spatial Index Function

```
1 function(doc) {
2     if (doc.loc) {
3         emit(/* key */
4
5
6         ,/* value */);
7     }
8 }
```


Spatial Index Function

```
1 function(doc) {
2     if (doc.loc) {
3         emit({
4             type: "Point",
5             coordinates: doc.loc
6         }, /* value */);
7     }
8 }
```

Spatial Index Function

```
1 function(doc) {
2   if (doc.loc) {
3     emit({
4       type: "Point",
5       coordinates: doc.loc
6     }, doc._id);
7   }
8 }
```

OpenSearch Geo

OpenSearch Geo

```
http://localhost:5984/db/_design/stations/  
_spatial/points?bbox=-110,-60,-30,15
```

OpenSearch Geo

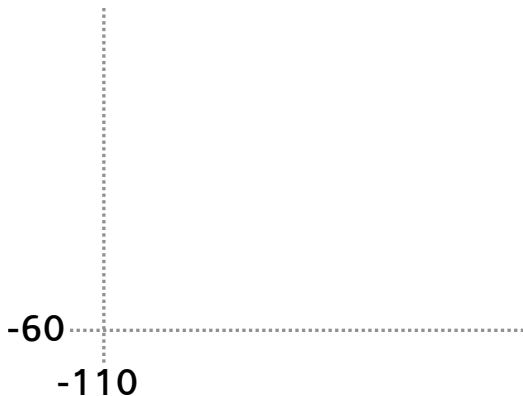
```
http://localhost:5984/db/_design/stations/  
_spatial/points?bbox=-110, -60, -30, 15
```



-110

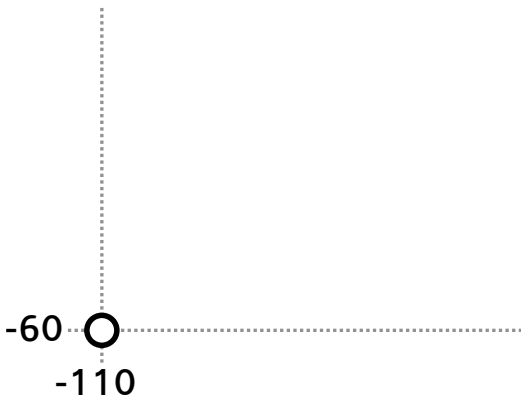
OpenSearch Geo

```
http://localhost:5984/db/_design/stations/  
_spatial/points?bbox=-110,-60,-30,15
```



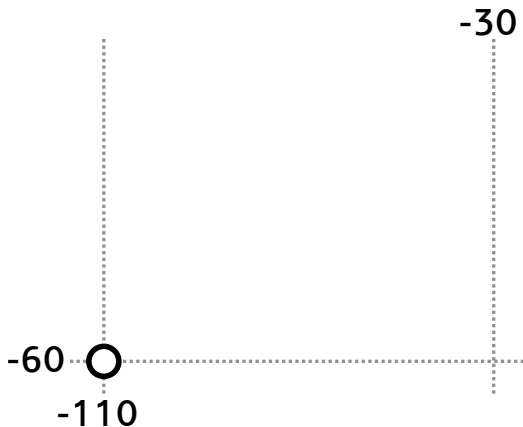
OpenSearch Geo

```
http://localhost:5984/db/_design/stations/  
_spatial/points?bbox=-110,-60,-30,15
```



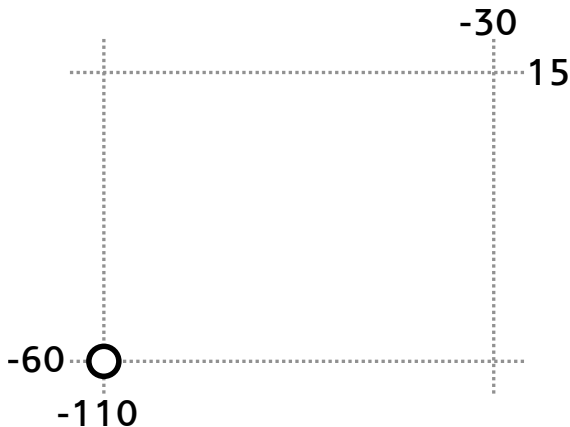
OpenSearch Geo

```
http://localhost:5984/db/_design/stations/  
_spatial/points?bbox=-110,-60,-30,15
```



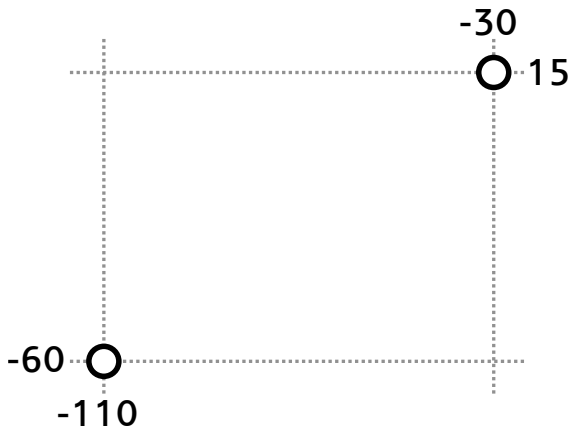
OpenSearch Geo

```
http://localhost:5984/db/_design/stations/  
_spatial/points?bbox=-110,-60,-30,15
```



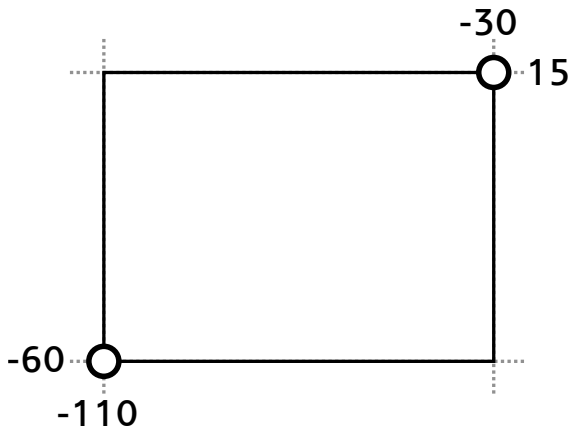
OpenSearch Geo

```
http://localhost:5984/db/_design/stations/  
_spatial/points?bbox=-110,-60,-30,15
```



OpenSearch Geo

`http://localhost:5984/db/_design/stations/
_spatial/points?bbox=-110,-60,-30,15`



Web Mapping Applications

Serving up data

- Base map (e.g. Google, OpenStreetMap)

Source: <http://www.openstreetbrowser.org/>

Serving up data

- Base map (e.g. Google, OpenStreetMap)
- Some overlay (your data)

Source: <http://www.openstreetbrowser.org/>

Typical 3-Tier architecture



Client

JavaScript
(e.g. OpenLayers)



Server

Web Map/Feature Server
(e.g. GeoServer, MapServer)



Database

Geospatial database
(e.g. PostGIS, SpatiaLite)

2-Tier architecture with CouchDB



Client

JavaScript
(e.g. OpenLayers,
code to access CouchDB)



CouchDB
+ GeoCouch

Applications



PDX API

UPDATE 7/2016: Creating your Portland Food Cart Finder iPhone App powered by PDX API! Available for free in the App Store soon. Follow my feeders if you want to stay up to date! [Example](#)

PDX API is a JSON API that provides access to data from CivicApps, the open data initiative by the City of Portland.

Much of the data on CivicApps is in a very raw and hard to use form. PDX API exists to filter the data from CivicApps and provide it in the easiest way possible to application developers wanting to consume the data.

API

`http://pdxapi.com/resources?lat=45.5171&lon=-122.6761`

You must specify the resource, dataset and supply a bounding box. The bounding box pair, y and x , represent the lower left and upper right points of the bounding box, respectively.

You'll need to request the data using library that supports JSONP, such as [jQuery](#).

Drag the map around to explore bike rack locations

Example

jQuery JSONP request to the `http://pdxapi.com/resources/bike_rack_locations`:

```
$.getJSON(
  url: 'http://www.pdxapi.com/resources/bike_rack_locations',
  dataType: 'jsonp',
  data: {
    'lat': '45.5171', 'lon': '-122.6761', 'lat2': '45.5171', 'lon2': '-122.6761'
  },
  success: function(data) {
    // manipulate data
  },
  error: function() {
    //
  }
);
```

Datasets

Datasets currently available through PDX API:

```
bike_rack_locations
bicycles
bicycles_availability
bike_trails
bicycles
cattle
cityhall
geodata
events
locations
```

GeoNotifications for
mltab@gmail.com

[Edit User](#) | [Logout](#)

Contact Methods

- Phone Call to 15038302651
[Edit](#) [Delete](#) [Test](#)
- "orange", a Text Message to 15038302651
[Edit](#) [Delete](#) [Test](#)
- "Chris", a Phone Call to 7342761100
[Edit](#) [Delete](#) [Test](#)
- [Add a contact method](#)

Schedules

- **Bike Parking PDX**
Subscription:
Area:
- Contact Method:
-

Your areas of interest

Long term goals

- Polygon and radius search (→ Erlang geometry library)
- Layer on top (e.g. for clustering, reprojection)
- Sharding

Thanks!

Get in touch with me

- Website: <http://vmx.cx/>
- IRC: vmx @ freenode
- Email: volker@couchone.com
- Jabber: [volker@vmx.cx](xmpp:volker@vmx.cx)
- Twitter: @vmische