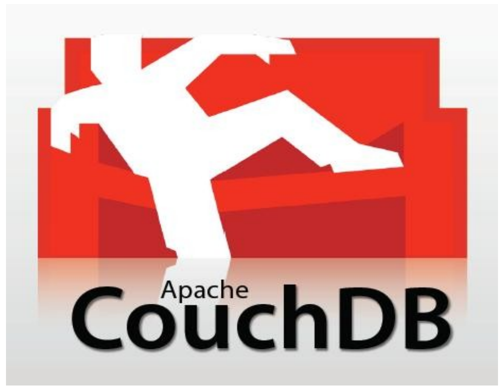
Karlsruher Institut für Technologie

Université Claude Bernard Lyon 1

Data analysis using the CouchDB database

Cécile Kéfélian

+

=


KSETA Freudenstadt workshop, 17/10/2013

Goal of the workshop

Getting an overview of the CouchDB
database and its usefulness
for monitoring and data analysis

- What is CouchDB ?
- What are its benefits ?
- How to get informations for it using views ?
- How to handle Big Data problem ?
- How to use CouchDB from a python script using couchdbkit ?
- What interesting features are offered by CouchDB ?
  - CouchApps
  - Replication

Introduction to CouchDB

CouchDB: ***C**luster **O**f **U**nreliable **C**ommodity **H**ardware **D**ata**B**ase*

Definition from official website:

"CouchDB is an open source document-oriented database.
Store your data with JSON documents.
Access your documents with your web browser, via HTTP.
Query, combine, and transform your documents with JavaScript."

open source

NoSQL

schema-free

CouchApps

document-based

key/value pairs

N-Master replications

views

revision

map/reduce

json-based

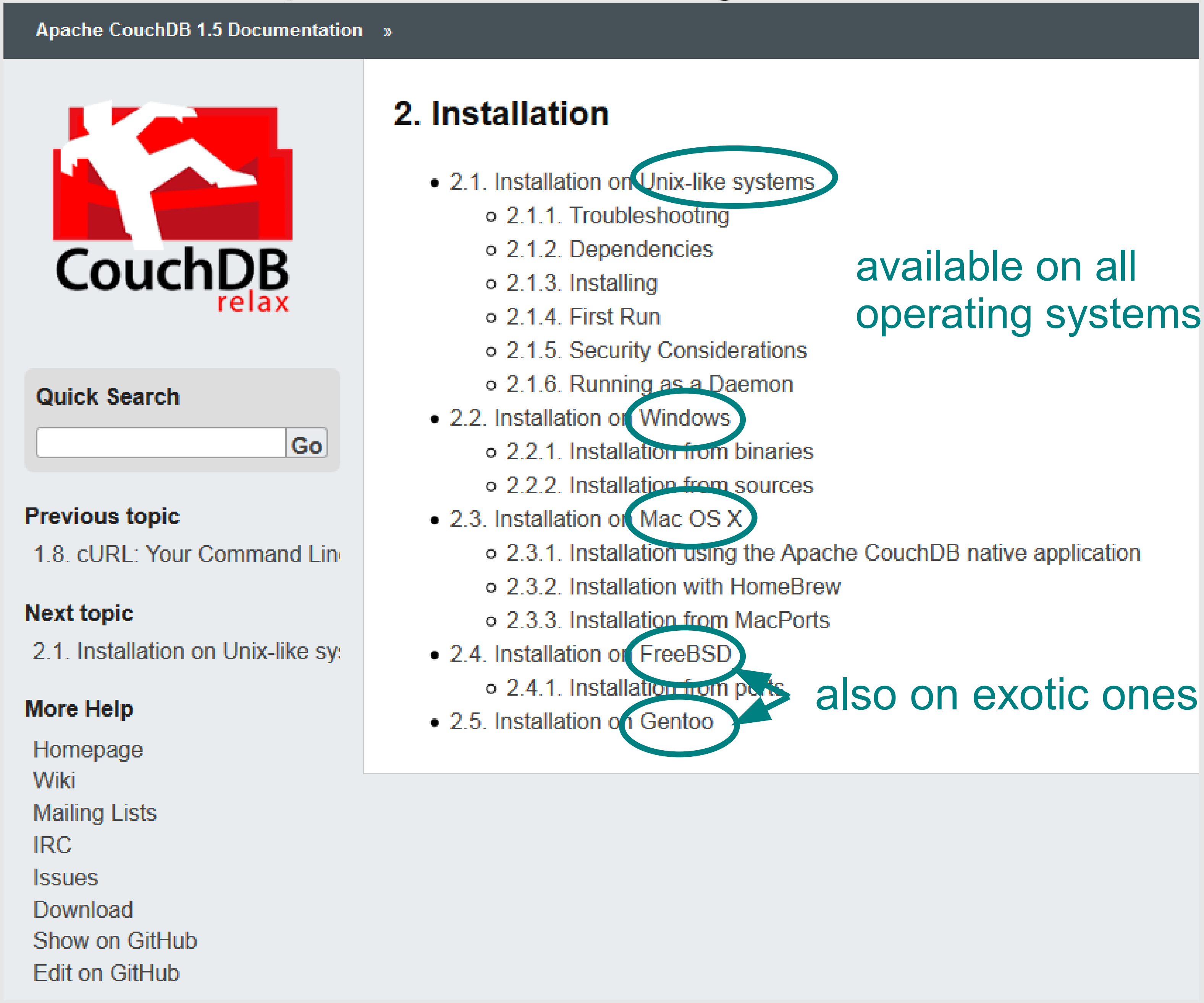
RESTfull API

Infinite applications: films, sms, contacts, cooking recipes, web apps, blogs, websites...
… monitor the detector temperature, store analysis results

Installation of couchDB

→ pre-compiled binaries for all platforms available

http://docs.couchdb.org/en/latest/install/index.html



Apache CouchDB 1.5 Documentation »

2. Installation

- 2.1. Installation on Unix-like systems
  - 2.1.1. Troubleshooting
  - 2.1.2. Dependencies
  - 2.1.3. Installing
  - 2.1.4. First Run
  - 2.1.5. Security Considerations
  - 2.1.6. Running as a Daemon
- 2.2. Installation on Windows
  - 2.2.1. Installation from binaries
  - 2.2.2. Installation from sources
- 2.3. Installation on Mac OS X
  - 2.3.1. Installation using the Apache CouchDB native application
  - 2.3.2. Installation with HomeBrew
  - 2.3.3. Installation from MacPorts
- 2.4. Installation on FreeBSD
  - 2.4.1. Installation from ports
- 2.5. Installation on Gentoo

available on all operating systems

also on exotic ones ;)

Quick Search

Go

Previous topic

1.8. cURL: Your Command Lin

Next topic

2.1. Installation on Unix-like sy

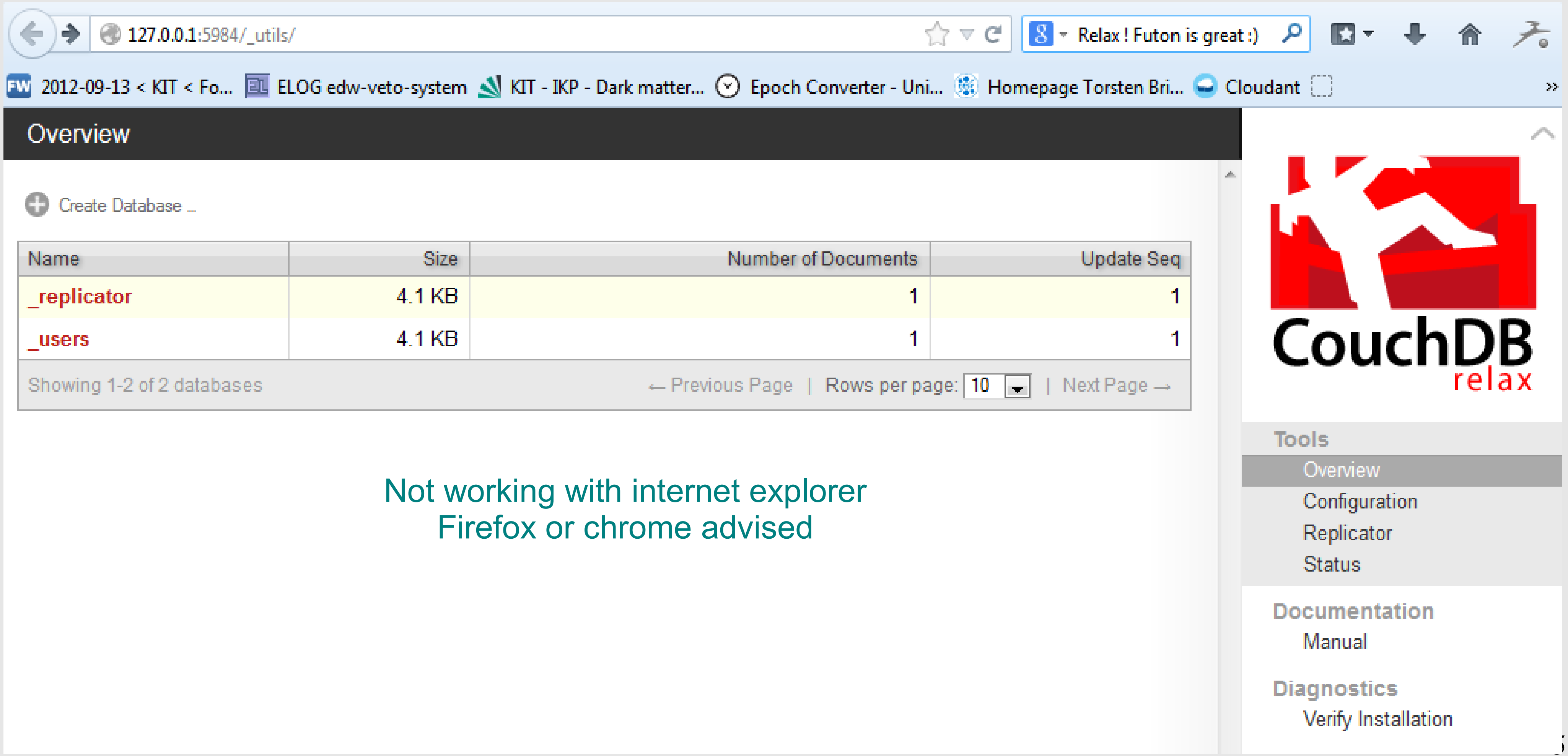
More Help

Homepage
Wiki
Mailing Lists
IRC
Issues
Download
Show on GitHub
Edit on GitHub

How to administrate the CouchDB database

- From creation to replication to data insertion, CouchDB administration can be done via HTTP
CouchDB is a RESTful API → the 4 HTTP methods GET,POST,PUT and DELETE can be used
→ Terminal + command line utility to throw around HTTP requests (like curl)
http://docs.couchdb.org/en/latest/intro/tour.html

- Futon (web build-in administration interface)
→ load Futon in your browser: http://127.0.0.1:5984/_utils/



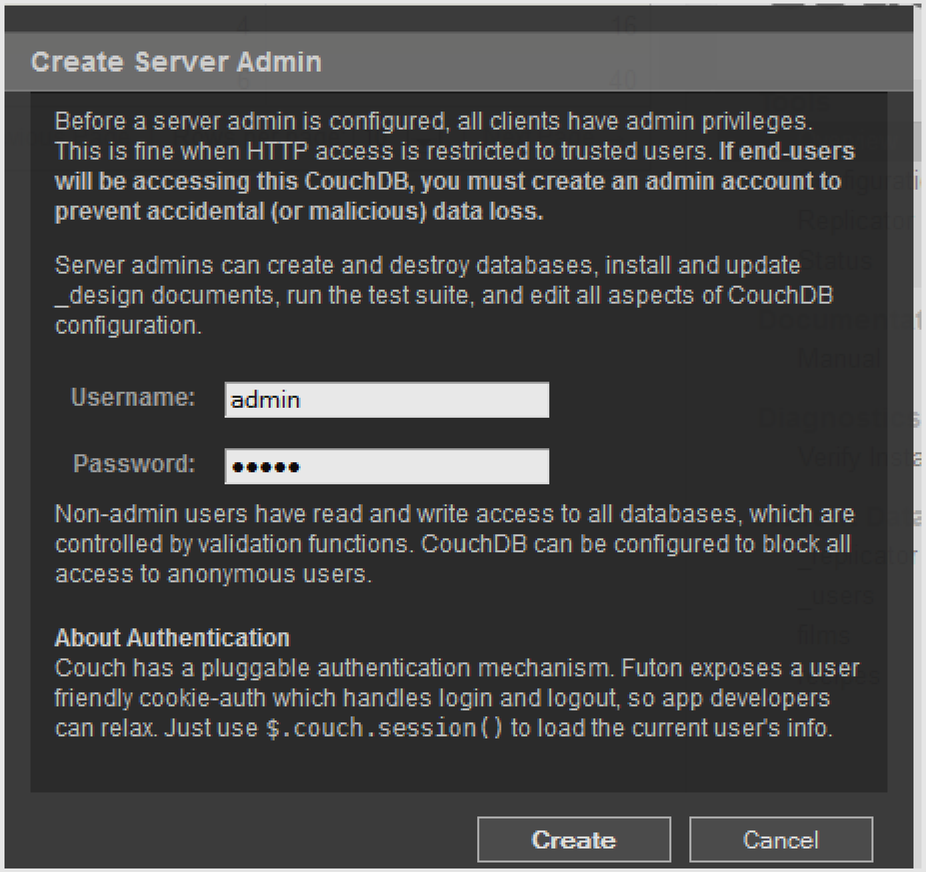
Name	Size	Number of Documents	Update Seq
_replicator	4.1 KB	1	1
_users	4.1 KB	1	1

Showing 1-2 of 2 databases ← Previous Page | Rows per page: 10 | Next Page →

Not working with internet explorer
Firefox or chrome advised

Admin rights

By default, CouchDB gives
every user admin rights
on all databases.



Create Server Admin

Before a server admin is configured, all clients have admin privileges. This is fine when HTTP access is restricted to trusted users. **If end-users will be accessing this CouchDB, you must create an admin account to prevent accidental (or malicious) data loss.**

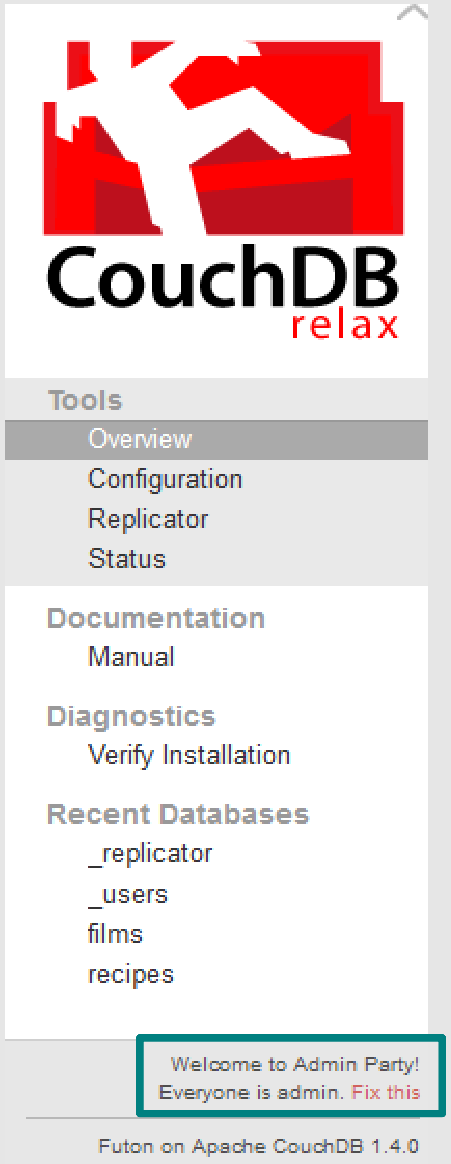
Server admins can create and destroy databases, install and update _design documents, run the test suite, and edit all aspects of CouchDB configuration.

Username: admin

Password: •••••

Non-admin users have read and write access to all databases, which are controlled by validation functions. CouchDB can be configured to block all access to anonymous users.

About Authentication
Couch has a pluggable authentication mechanism. Futon exposes a user friendly cookie-auth which handles login and logout, so app developers can relax. Just use `$.couch.session()` to load the current user's info.



Welcome to Admin Party!
Everyone is admin. Fix this

Futon on Apache CouchDB 1.4.0

MySQL vs NoSQL database

SQL (**S**tructured **Q**uery **L**anguage): programming language designed for managing data held in a relational database.

MySQL

- Support the SQL
- Relational database (collection of tables of data items, described and organized according to the relational model)
- Collection of tables of data items to be defined up-front
- Relationship between tuples have to be defined
- Specific protocol used to communicate with the db

➡ **up-front defined structure**

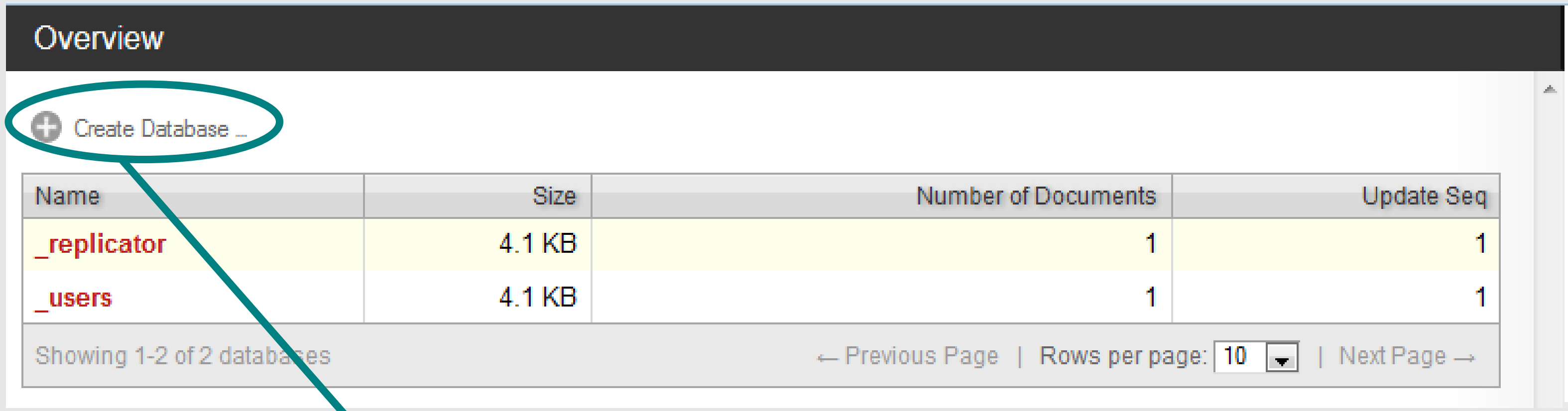
NoSQL
CouchDB, MongoDB

- Do not support the SQL
- Document-based database
- Collection of self-contained documents which can differ from each other (document not stored in a defined table)
- No relationships have to be defined
- HTTP protocol used to communicate with the db

➡ **schema-free**

Let's create a database containing the list of the films you watched :)

Creating a database

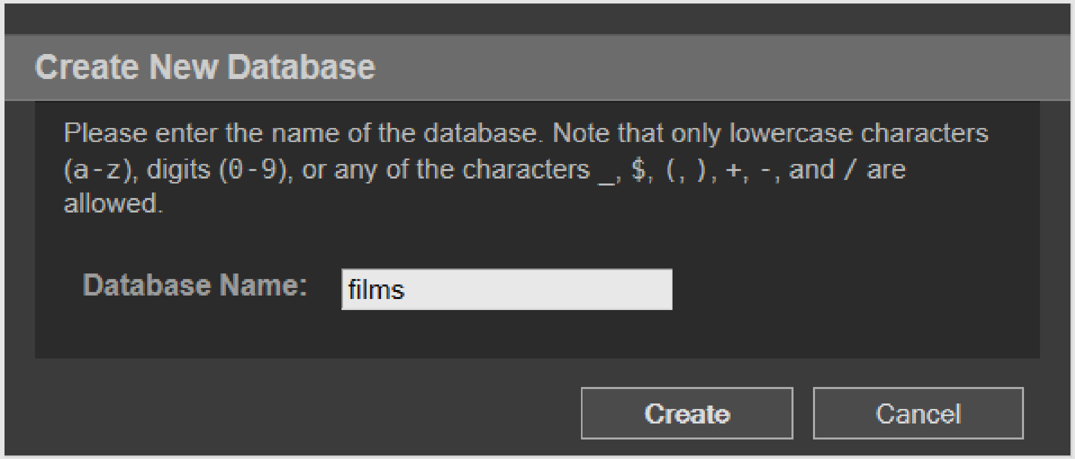


Overview

Create Database ...

Name	Size	Number of Documents	Update Seq
_replicator	4.1 KB	1	1
_users	4.1 KB	1	1

Showing 1-2 of 2 databases ← Previous Page | Rows per page: 10 | Next Page →

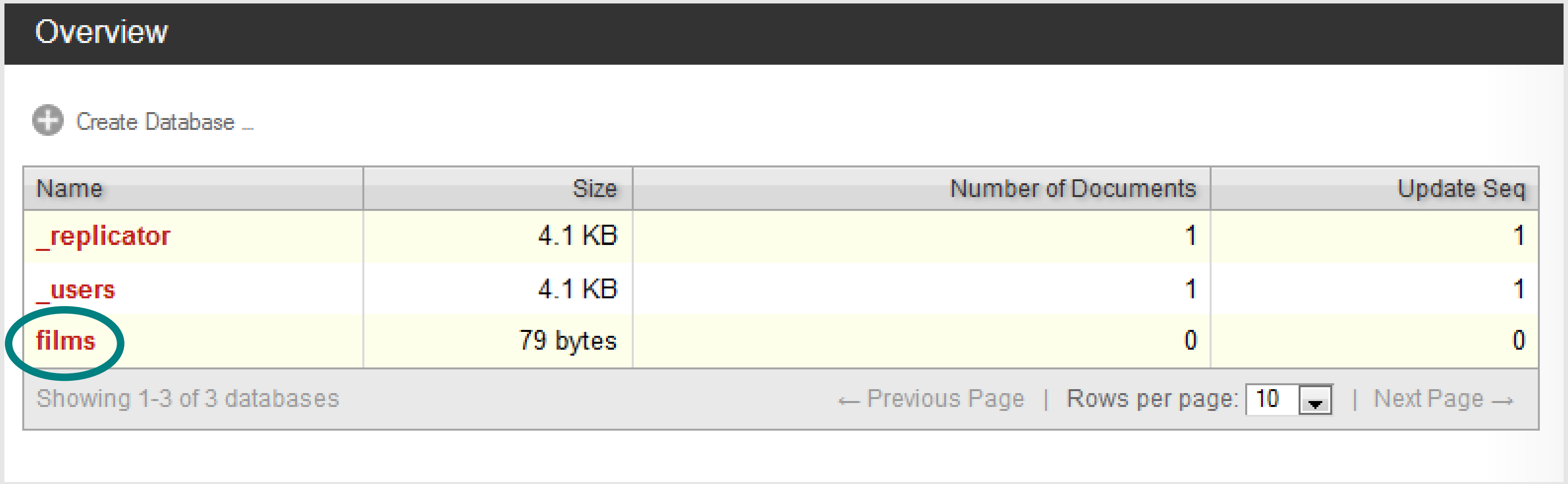


Create New Database

Please enter the name of the database. Note that only lowercase characters (a-z), digits (0-9), or any of the characters _, $, (,), +, -, and / are allowed.

Database Name: films

Create Cancel



Overview

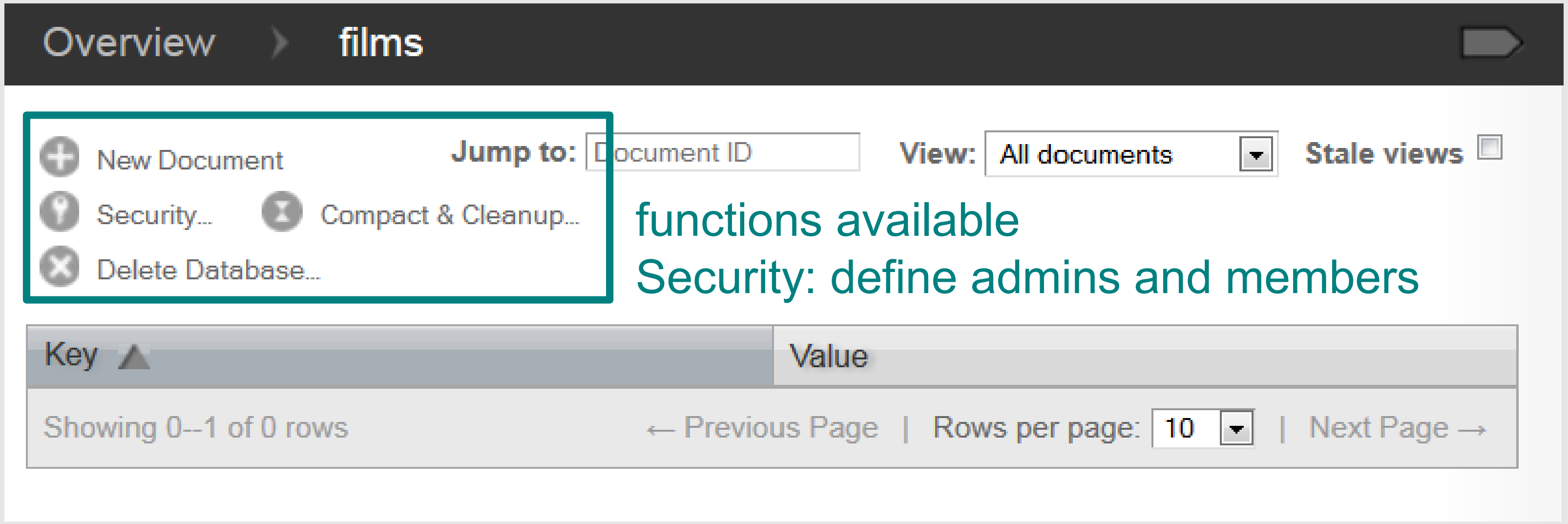
Create Database ...

Name	Size	Number of Documents	Update Seq
_replicator	4.1 KB	1	1
_users	4.1 KB	1	1
films	79 bytes	0	0

Showing 1-3 of 3 databases ← Previous Page | Rows per page: 10 | Next Page →

Creating a document

Database empty at the moment...



Overview › films

New Document
Security...
Compact & Cleanup...
Delete Database...

Jump to: Document ID

View: All documents

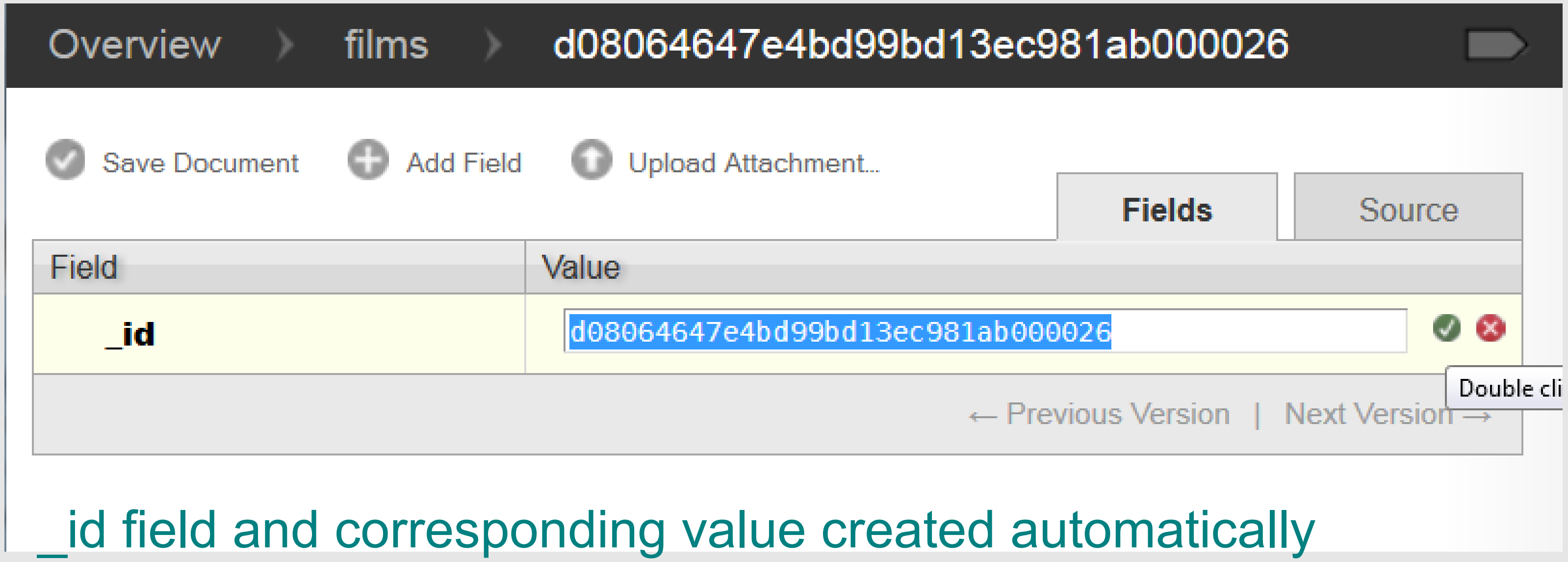
Stale views

Key ▲	Value

Showing 0--1 of 0 rows ← Previous Page | Rows per page: 10 | Next Page →

functions available
Security: define admins and members

After clicking on new document...



Overview › films › d08064647e4bd99bd13ec981ab000026

Save Document
Add Field
Upload Attachment...

Fields | Source

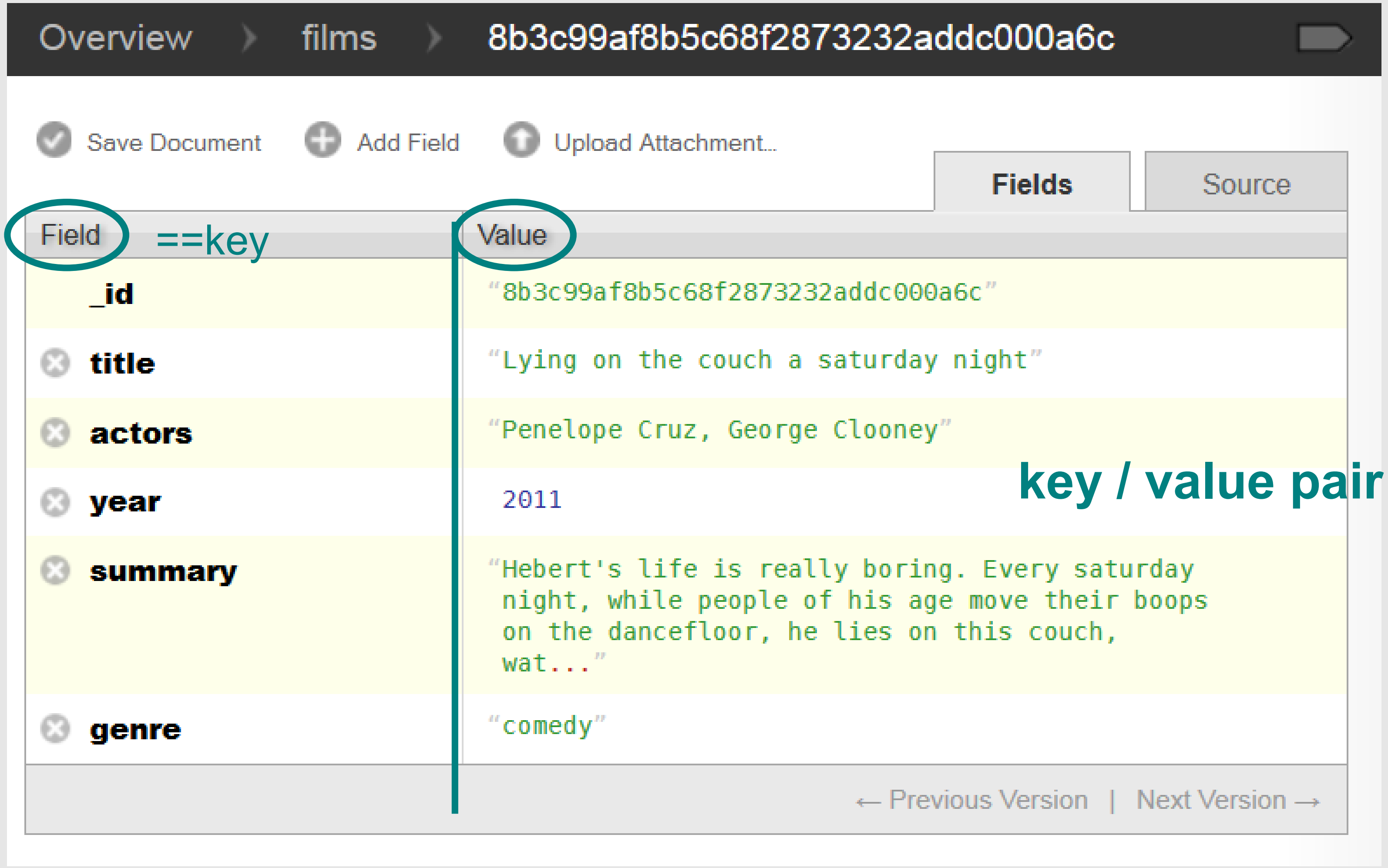
Field	Value
_id	d08064647e4bd99bd13ec981ab000026

Double cli

← Previous Version | Next Version →

_id field and corresponding value created automatically
→ unique value identifying the document

Document structure


Overview
films
8b3c99af8b5c68f2873232addc000a6c
Save Document
Add Field
Upload Attachment...
Fields
Source
Field
==key
Value
key / value pair structure
← Previous Version | Next Version →

Field	Value
_id	"8b3c99af8b5c68f2873232addc000a6c"
title	"Lying on the couch a saturday night"
actors	"Penelope Cruz, George Clooney"
year	2011
summary	"Hebert's life is really boring. Every saturday night, while people of his age move their boops on the dancefloor, he lies on this couch, wat..."
genre	"comedy"

Field (=key) → string

Value → JSON (JavaScript Object Notation) object :

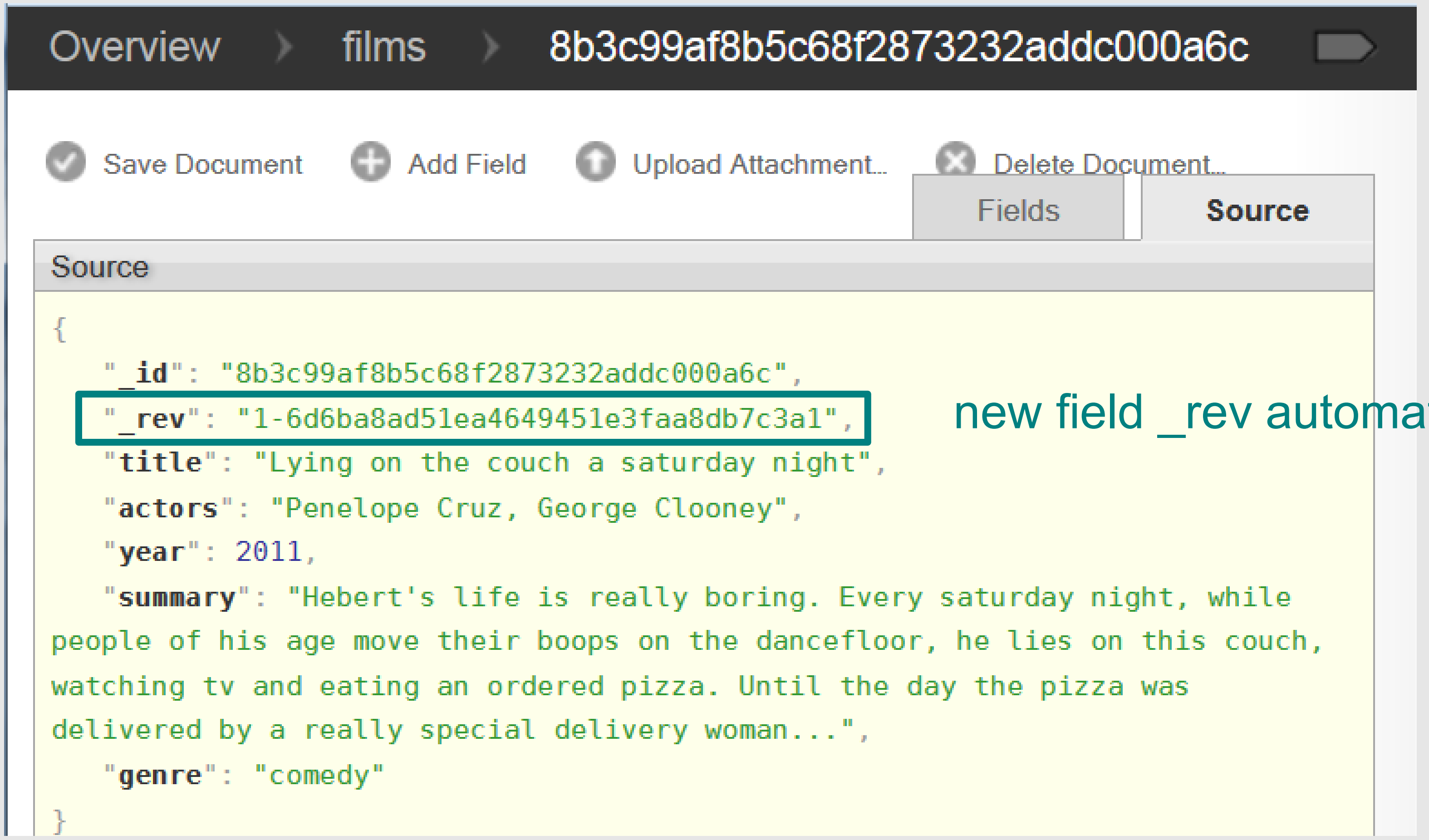
- number (either integer or float)
- string
- boolean (true/false value)
- array
- object (a set of key/value pairs)

Use to format the content and structure of the data and responses

CouchDB also supports attachments.

Document revision

After saving...

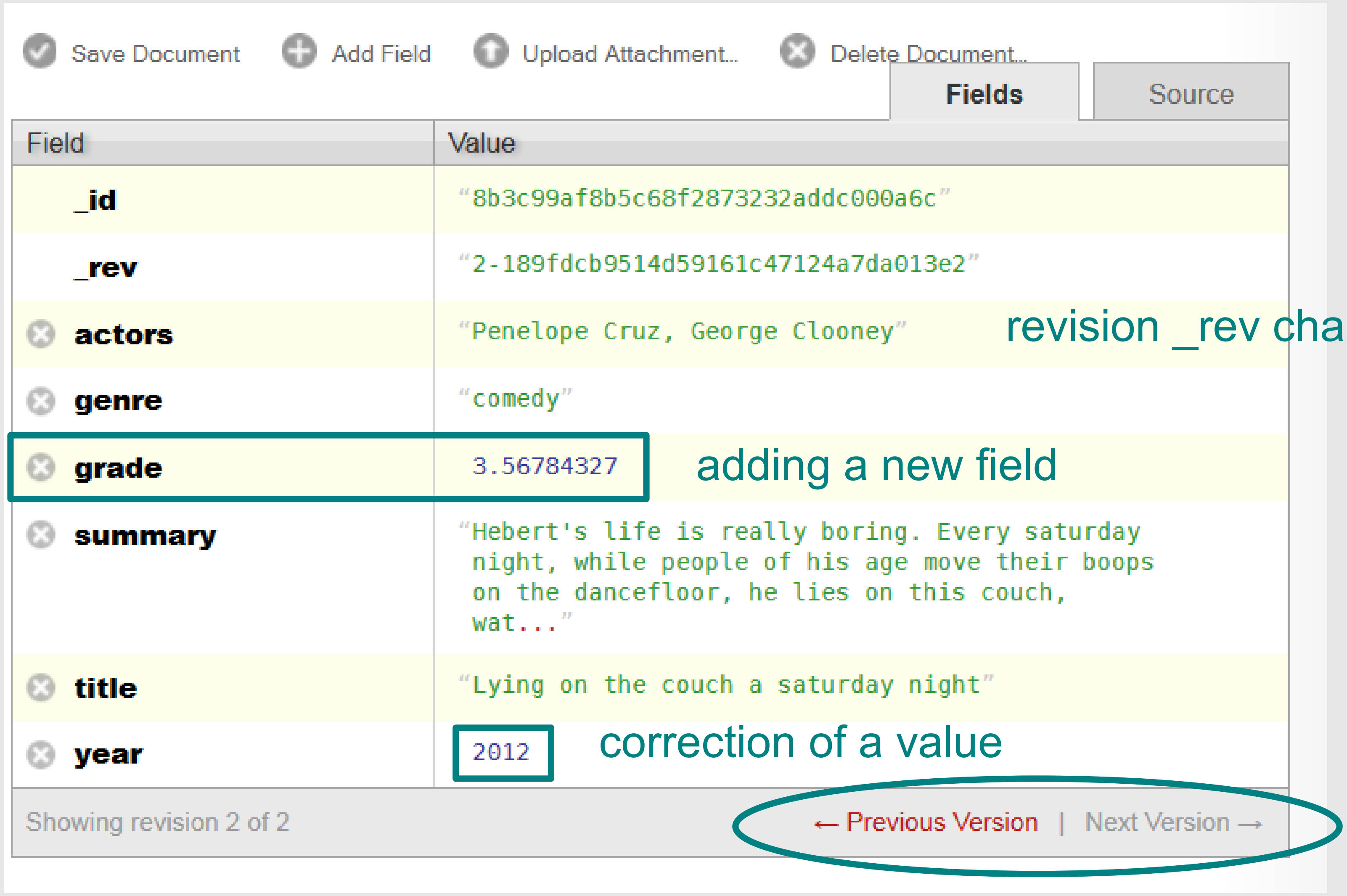


```
{
   "_id": "8b3c99af8b5c68f2873232addc000a6c",
   "_rev": "1-6d6ba8ad51ea4649451e3faa8db7c3a1",
   "title": "Lying on the couch a saturday night",
   "actors": "Penelope Cruz, George Clooney",
   "year": 2011,
   "summary": "Hebert's life is really boring. Every saturday night, while
people of his age move their boops on the dancefloor, he lies on this couch,
watching tv and eating an ordered pizza. Until the day the pizza was
delivered by a really special delivery woman...",
   "genre": "comedy"
}
```

new field _rev automatically added

Document revision

Each time the document is modified (key/value pair added or modified) and saved, a new _rev value is given to the document



Field	Value
_id	"8b3c99af8b5c68f2873232addc000a6c"
_rev	"2-189fdcb9514d59161c47124a7da013e2"
actors	"Penelope Cruz, George Clooney"
genre	"comedy"
grade	3.56784327
summary	"Hebert's life is really boring. Every saturday night, while people of his age move their boops on the dancefloor, he lies on this couch, wat..."
title	"Lying on the couch a saturday night"
year	2012

Showing revision 2 of 2

revision _rev changes after saving

adding a new field

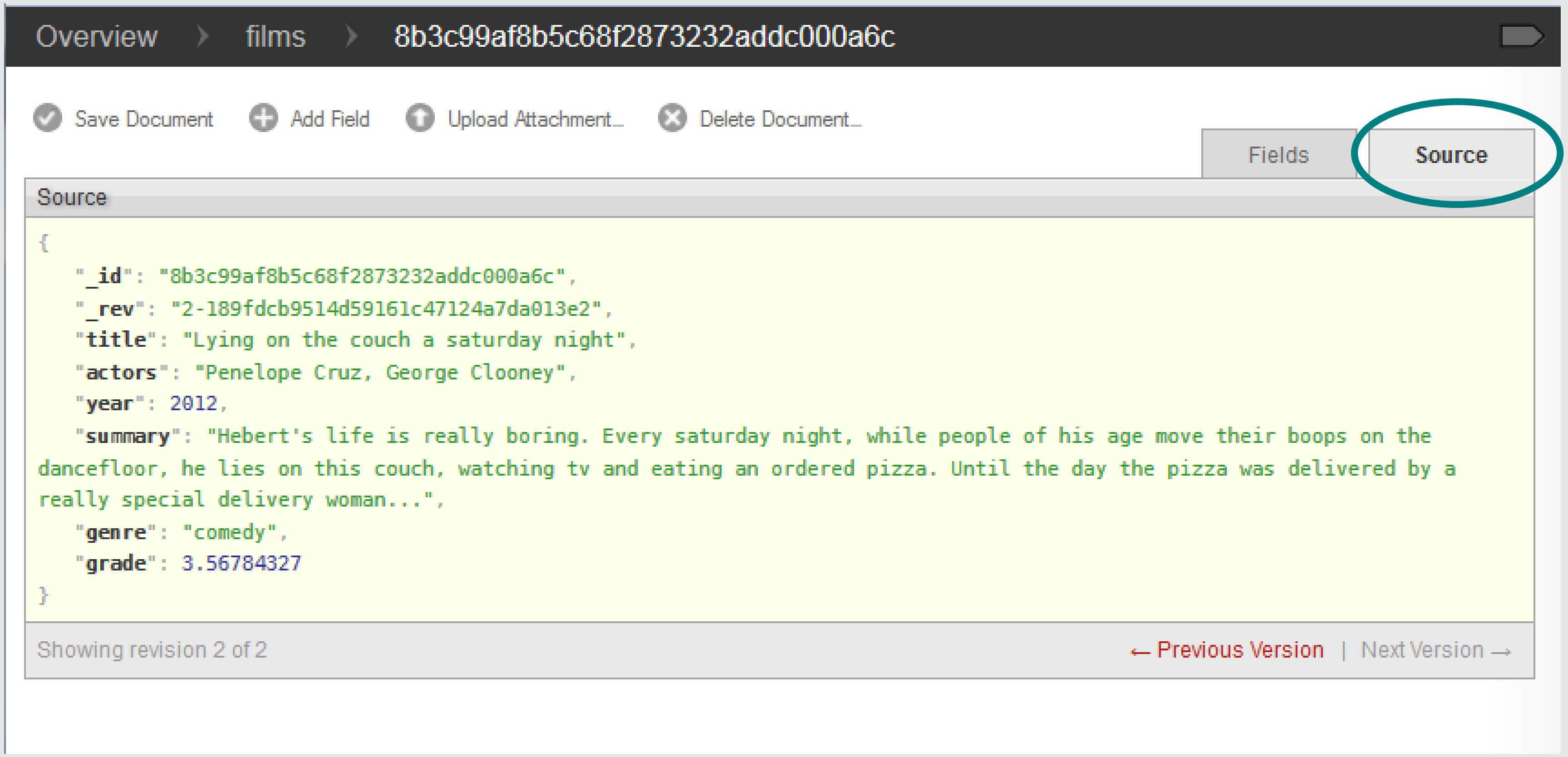
correction of a value

previous version accessible !

All the document revisions can be deleted by clicking on Compact&Cleanup

JSON-based document storage

After clicking on source:

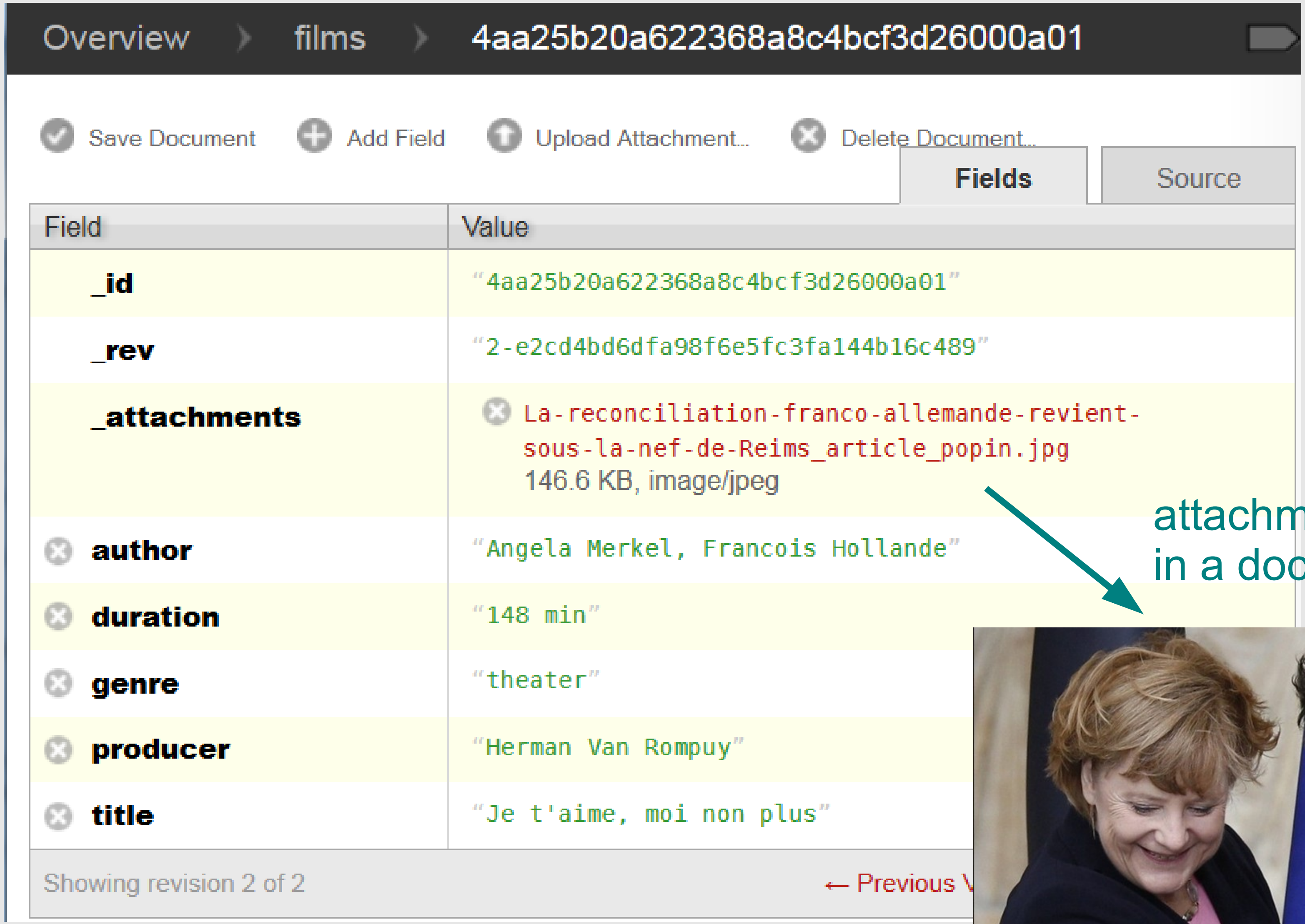


```
{
   "_id": "8b3c99af8b5c68f2873232addc000a6c",
   "_rev": "2-189fdcb9514d59161c47124a7da013e2",
   "title": "Lying on the couch a saturday night",
   "actors": "Penelope Cruz, George Clooney",
   "year": 2012,
   "summary": "Hebert's life is really boring. Every saturday night, while people of his age move their boops on the
dancefloor, he lies on this couch, watching tv and eating an ordered pizza. Until the day the pizza was delivered by a
really special delivery woman...",
   "genre": "comedy",
   "grade": 3.56784327
}
```

- Futon interprets the key/value pairs as JSON objects.
- By clicking on source, the underlying JSON document is displayed

Schema-free database

We can add a document with different key/value pair in the “films” database
→ documents of a given db do not necessarily have the same structure



Field	Value
_id	“4aa25b20a622368a8c4bcf3d26000a01”
_rev	“2-e2cd4bd6dfa98f6e5fc3fa144b16c489”
_attachments	La-reconciliation-franco-allemande-revient-sous-la-nef-de-Reims_article_popin.jpg 146.6 KB, image/jpeg
author	“Angela Merkel, Francois Hollande”
duration	“148 min”
genre	“theater”
producer	“Herman Van Rompuy”
title	“Je t'aime, moi non plus”

Showing revision 2 of 2 ← Previous V

attachments possible in a document :) :) :)

Useful JSON syntax

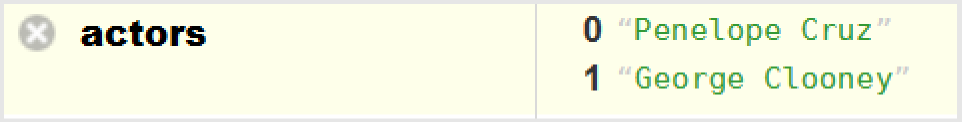
Value → JSON (JavaScript Object Notation) object :

- number (either integer or float)
- string
- boolean (true/false value)
- array
- set of key/value pairs

$\}$ Which syntax should be use for the interpreter to recognize the object type ?

- JSON arrays:

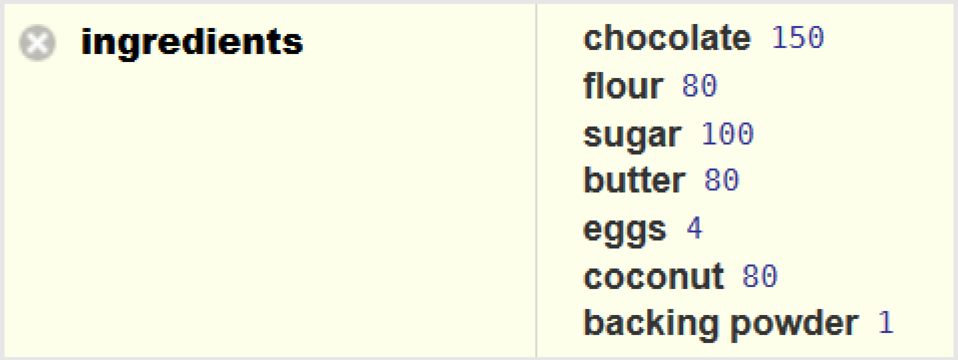
```
"actors": [
      "Penelope Cruz",
      "George Clooney"
  ],
```



```
actors    0 "Penelope Cruz"
          1 "George Clooney"
```

- JSON set of key/value pairs:

```
{
  "chocolate": 150,
  "flour": 80,
  "sugar": 100,
  "butter": 80,
  "eggs": 4,
  "coconut": 80,
  "backing powder": 1
}
```



```
ingredients    chocolate 150
               flour 80
               sugar 100
               butter 80
               eggs 4
               coconut 80
               backing powder 1
```

open source **NoSQL** **schema-free**

CouchApps

document-based

key/value pairs

N-Master replications

Apache CouchDB

views

revision

map/reduce

json-based

RESTfull API

Now, let's...

Get informations
from the database

Getting useful informations using views

• CouchDB is schema-free i.e. unstructured by nature
→ difficult to use in real-world applications
Use views to give structure to the data

• Two kind of view existing:

- permanent views (stored in design document): iterate over every document and build a list of documents with specific fields → improve the performance

- temporary views: executed on command but ressource-intensive and become slower as the amount of data stored in the db increases

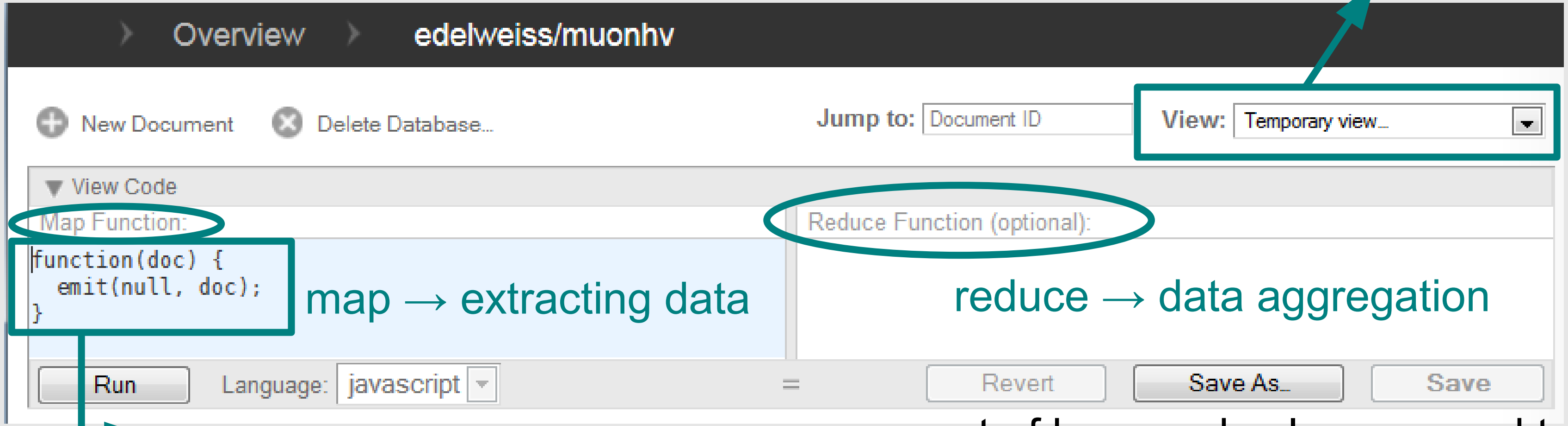
• Views based on Map/Reduce principle and using JavaScript functions

• Views are not updated after a document is saved but when it is run
→ first run can last long if there are many documents in the db

Getting useful informations using views

- Views based on the Map/Reduce principle

go to temporary view to write a view



```
function(doc) {
  emit(null, doc);
}
```

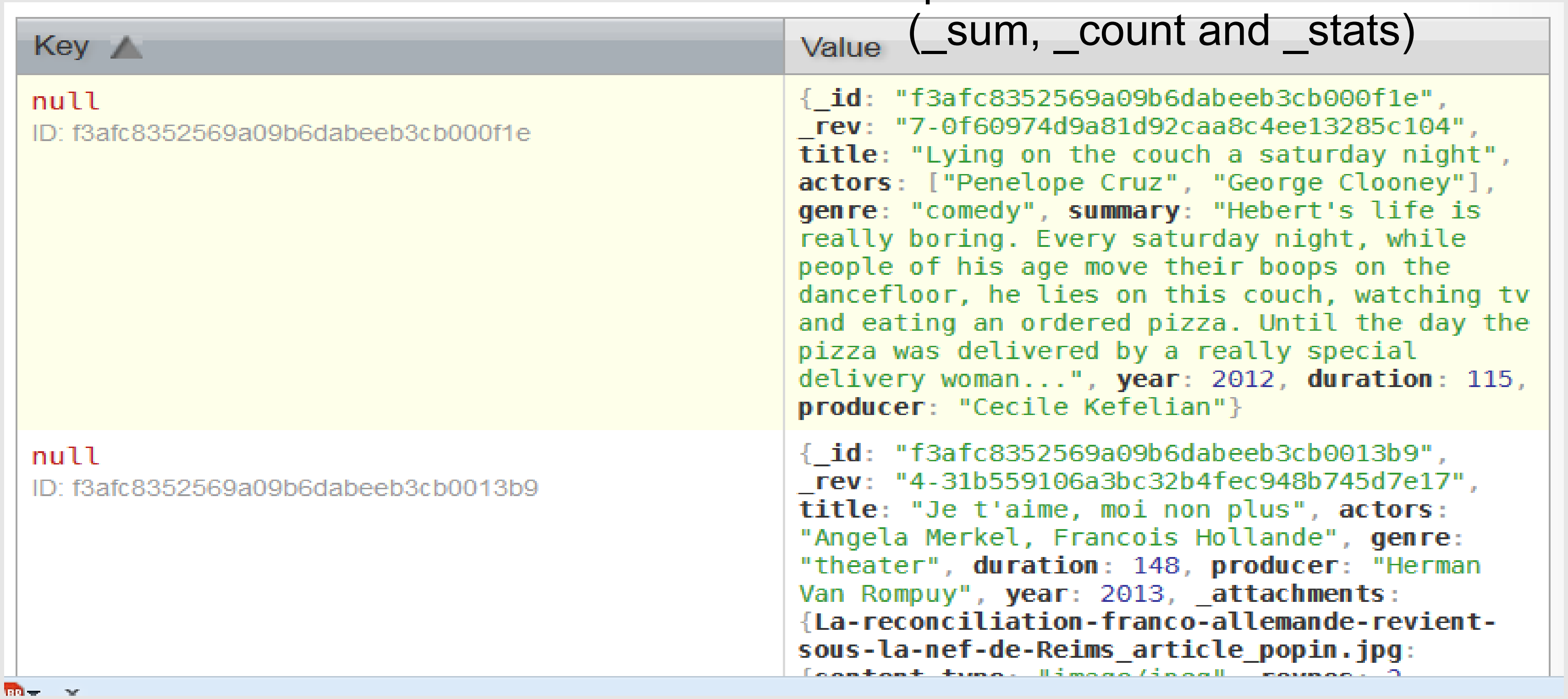
map → extracting data

reduce → data aggregation

simplest map function

- set of keys and values passed to it and combined to a single value
- predefined reduce functions (_sum, _count and _stats)

After clicking on “Run”, view output:



Key	Value
null ID: f3afc8352569a09b6dabeeb3cb000f1e	{_id: "f3afc8352569a09b6dabeeb3cb000f1e", _rev: "7-0f60974d9a81d92caa8c4ee13285c104", title: "Lying on the couch a saturday night", actors: ["Penelope Cruz", "George Clooney"], genre: "comedy", summary: "Hebert's life is really boring. Every saturday night, while people of his age move their boops on the dancefloor, he lies on this couch, watching tv and eating an ordered pizza. Until the day the pizza was delivered by a really special delivery woman...", year: 2012, duration: 115, producer: "Cecile Kefelian"}
null ID: f3afc8352569a09b6dabeeb3cb0013b9	{_id: "f3afc8352569a09b6dabeeb3cb0013b9", _rev: "4-31b559106a3bc32b4fec948b745d7e17", title: "Je t'aime, moi non plus", actors: "Angela Merkel, Francois Hollande", genre: "theater", duration: 148, producer: "Herman Van Rompuy", year: 2013, _attachments: {La-reconciliation-franco-allemande-revient-sous-la-nef-de-Reims_article_popin.jpg:

Map function syntax

Considering a document of the following form:

```
{
   "_id": "f3afc8352569a09b6dabeeb3cb000f1e",
   "_rev": "7-0f60974d9a81d92caa8c4ee13285c104",
   "string": "HelloWolrd",
   “int”:5,
   “output1”:”Couch”,
   “output2”:”DB”
}
```

Example of a map function:

```
function(doc) {
   if(doc.string && doc.int>5  )
     emit(doc.output1, doc.output2);
}
```

What you should know on map syntax:

- indent not compulsory
- doc[key1] <=> doc.key1
- between 2 conditions: &&
- if(doc.key3) => if the document has a field called key3, then continue
- emit() generates the output

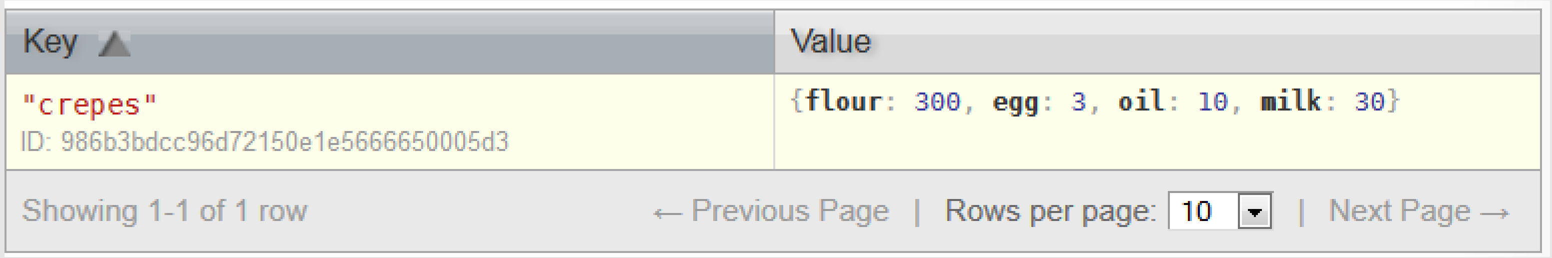
Examples of map functions

- Condition on simple key/value pair

```
function(doc) {
  if((doc.PreparationTime+doc.CookingTime) <20)
    emit(doc.name, doc.ingredients);
}
```

To view recipes with cooking+preparation time < 20 min

Corresponding output:



Key ▲	Value
"crepes" ID: 986b3bdcc96d72150e1e5666650005d3	{flour: 300, egg: 3, oil: 10, milk: 30}

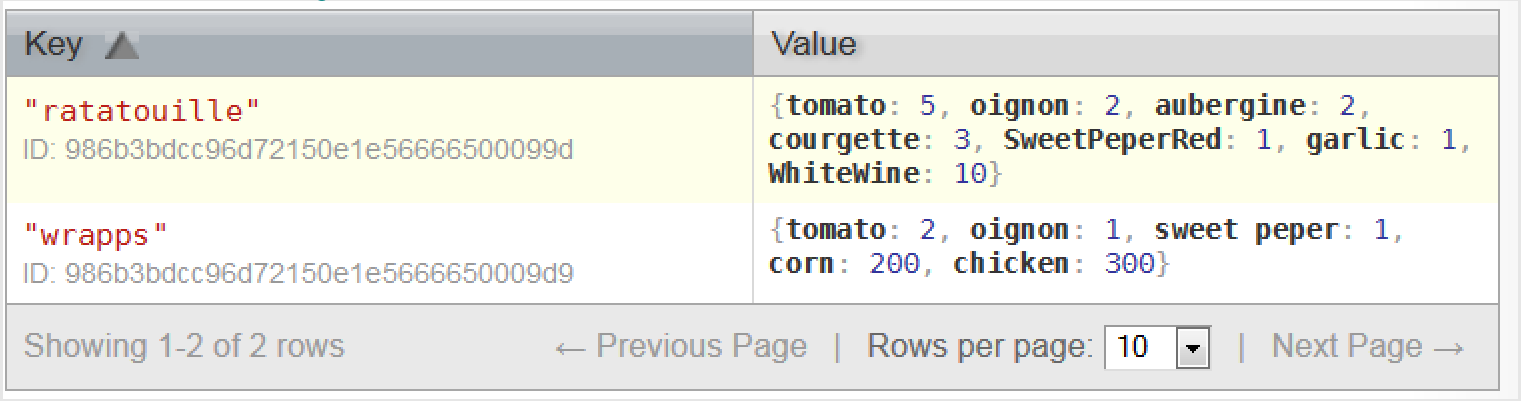
Showing 1-1 of 1 row ← Previous Page | Rows per page: 10 | Next Page →

- Condition on object elements:

```
function(doc) {
  if(doc.ingredients["tomato"]>0)
    emit(doc.name, doc.ingredients);
}
```

To view recipes using tomatoes

Corresponding output:



Key ▲	Value
"ratatouille" ID: 986b3bdcc96d72150e1e56666500099d	{tomato: 5, oignon: 2, aubergine: 2, courgette: 3, SweetPeperRed: 1, garlic: 1, WhiteWine: 10}
"wrapps" ID: 986b3bdcc96d72150e1e5666650009d9	{tomato: 2, oignon: 1, sweet peper: 1, corn: 200, chicken: 300}

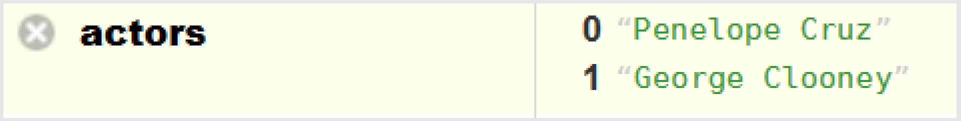
Showing 1-2 of 2 rows ← Previous Page | Rows per page: 10 | Next Page →

```
{
  "pastry brisée": 1,
  "oignon": 7,
  "egg": 5,
  "lardon": 250,
  "butter": 25,
  "liquid creme": 20
}
```

Examples of map functions

- Condition on table of values:

In the document:



actors	0 "Penelope Cruz" 1 "George Clooney"

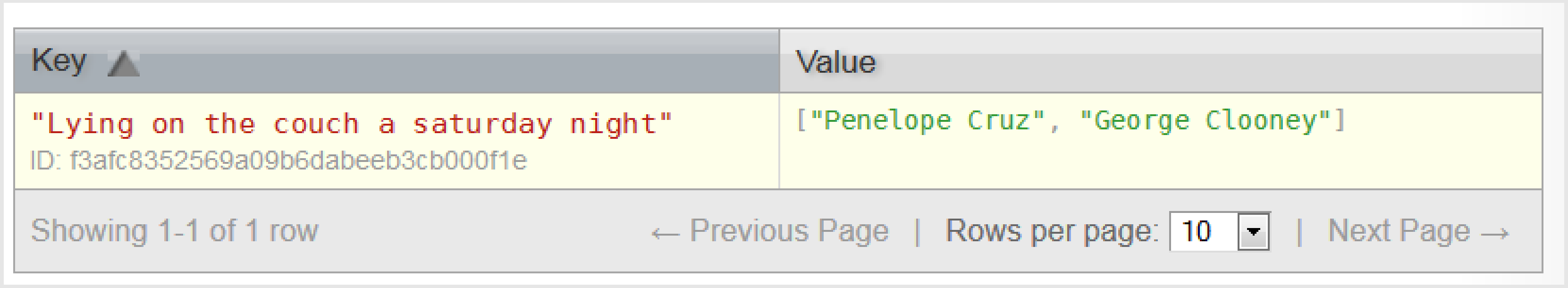
Underlying JSON doc:

```
"actors": [
      "Penelope Cruz",
      "George Clooney"
  ],
```

We need the following map function to query an actor:

```
function(doc) {
 for(var i=0;i<doc["actors"].length;i++){
   if(doc["actors"][i]=="Penelope Cruz")
     emit(doc.title, doc.actors);
 }
}
```

Corresponding output:



Key	Value
"Lying on the couch a saturday night" ID: f3afc8352569a09b6dabeeb3cb000f1e	["Penelope Cruz", "George Clooney"]

Showing 1-1 of 1 row ← Previous Page | Rows per page: 10 | Next Page →

Using CouchDB for physics purposes

Using CouchDB in Physics

We can store:
- DAQ informations: run configuration...
- Slow control (temperature, pressure...)
- Hardware maps
- Informations on detectors
- Energy resolution
- Noise spectra/filters
- Analysis results

Physics application often requires fast-growing db
Problem: CouchDB do not offer "horizonal" scaling i.e. across many servers

Solution:
Going from your small Couch...

... to the BigCouch

- BigCouch was released and is primarily maintained by Cloudant
- Based on CouchDB
- BigCouch allows to create clusters of CouchDBs that are distributed over many servers but appears to the user as one CouchDB instance
- All the CouchDB servers act together to store and retrieve documents, index and serve views, and serve CouchApps.

Cloudant website: https://cloudant.com/

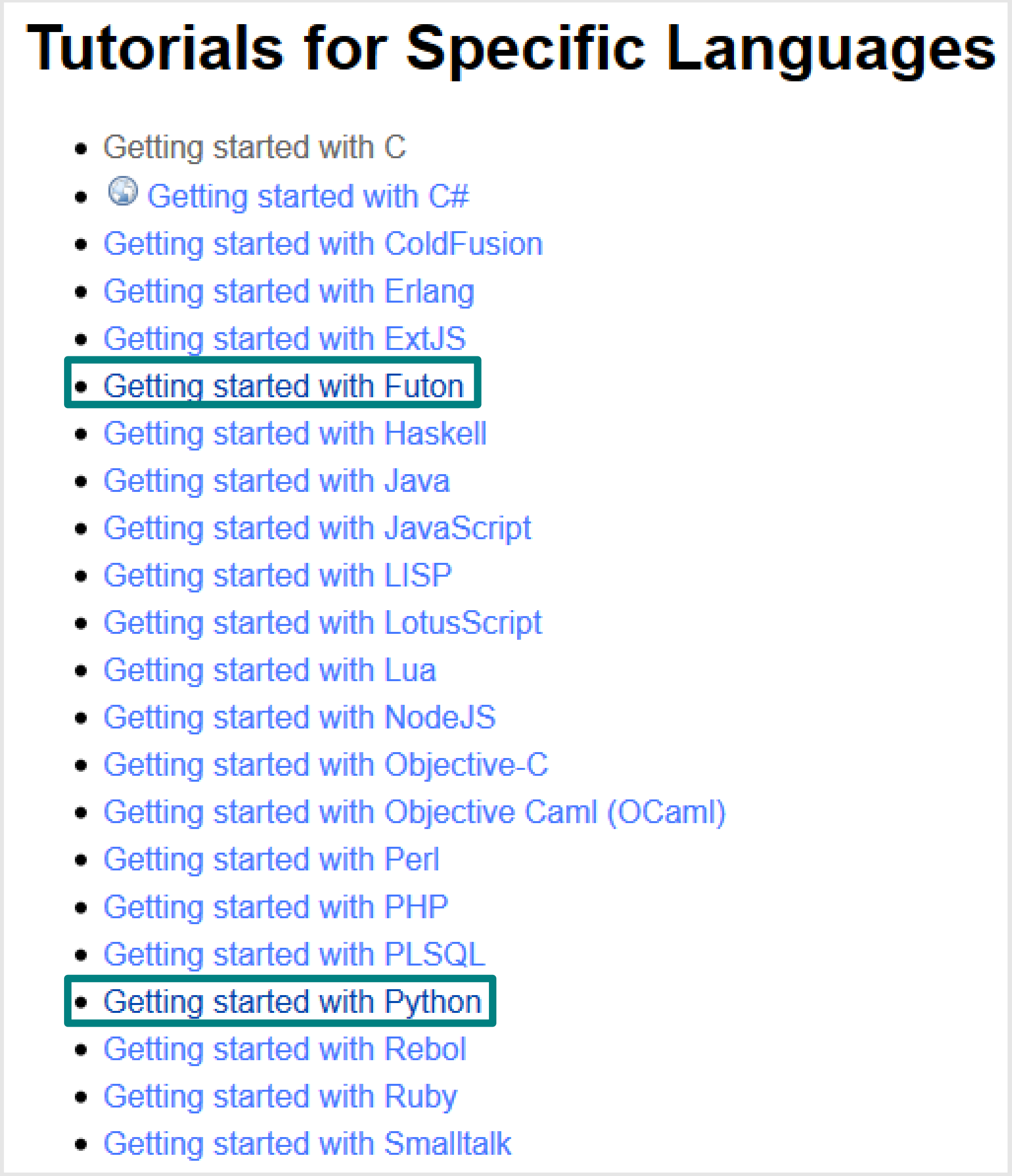
CouchDB available frameworks

• All languages which can deal with HTTP can be used to administrate the db

• Libraries existing for many languages

• C++ tools less developed and less convenient than python tools

Following examples using the python tool couchdbkit (provide a framework for Python to access and manage Couchdb)
→ http://couchdbkit.org/



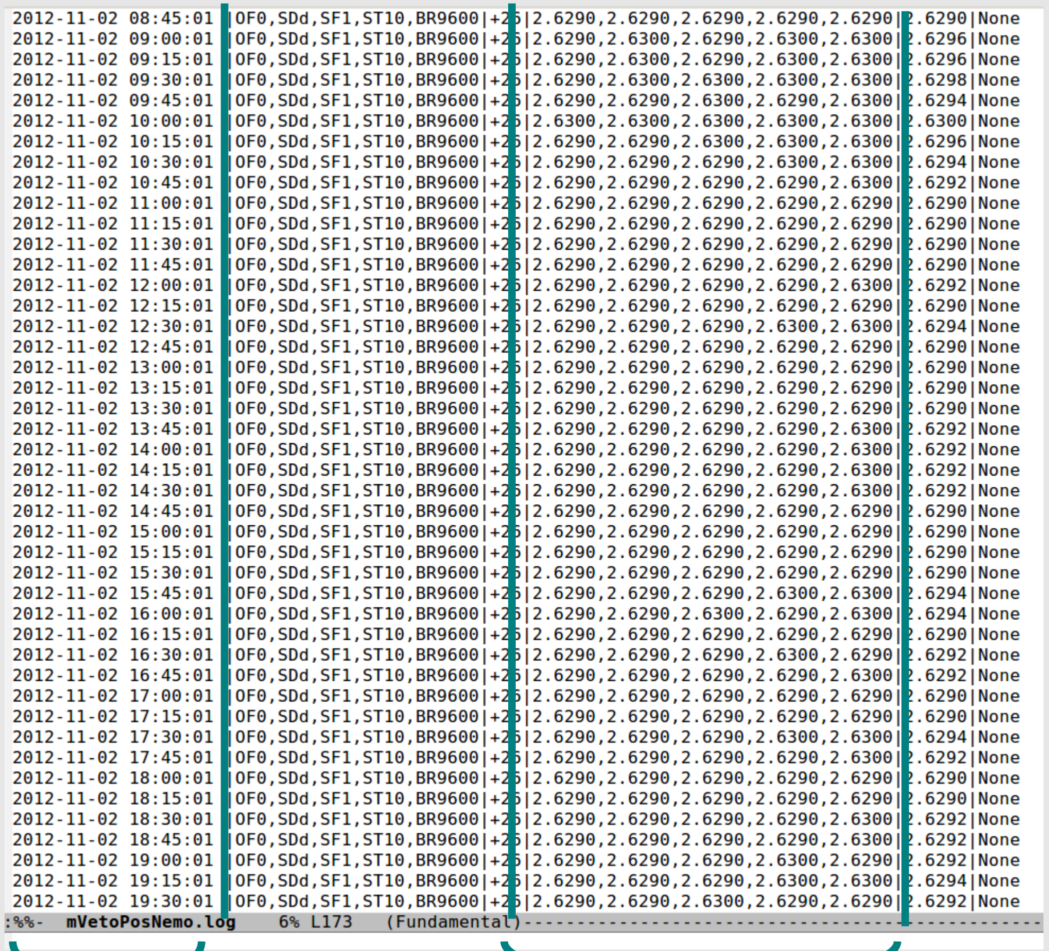
Tutorials for Specific Languages

- Getting started with C
- Getting started with C#
- Getting started with ColdFusion
- Getting started with Erlang
- Getting started with ExtJS
- Getting started with Futon
- Getting started with Haskell
- Getting started with Java
- Getting started with JavaScript
- Getting started with LISP
- Getting started with LotusScript
- Getting started with Lua
- Getting started with NodeJS
- Getting started with Objective-C
- Getting started with Objective Caml (OCaml)
- Getting started with Perl
- Getting started with PHP
- Getting started with PLSQL
- Getting started with Python
- Getting started with Rebol
- Getting started with Ruby
- Getting started with Smalltalk

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

Example: position monitoring using CouchDB

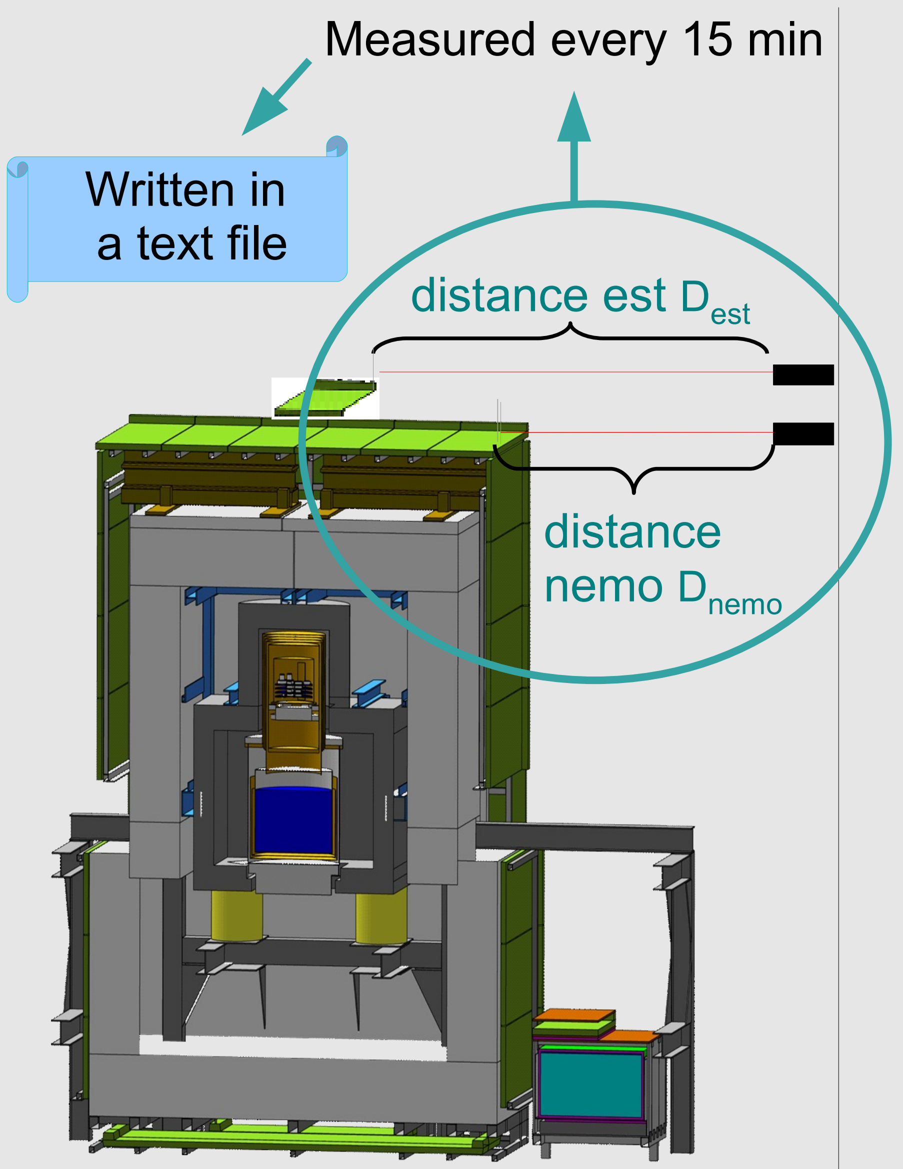
- Position of the muon veto chariots measured every 15 min in text files



```
2012-11-02 08:45:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 09:00:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6300,2.6290,2.6300,2.6300|2.6296|None
2012-11-02 09:15:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6300,2.6290,2.6300,2.6300|2.6296|None
2012-11-02 09:30:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6300,2.6300,2.6300,2.6300|2.6298|None
2012-11-02 09:45:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6300,2.6290,2.6300|2.6294|None
2012-11-02 10:00:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6300,2.6300,2.6300,2.6300,2.6300|2.6300|None
2012-11-02 10:15:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6300,2.6300,2.6300|2.6296|None
2012-11-02 10:30:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6300,2.6300|2.6294|None
2012-11-02 10:45:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6300|2.6292|None
2012-11-02 11:00:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 11:15:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 11:30:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 11:45:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 12:00:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6300|2.6292|None
2012-11-02 12:15:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 12:30:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6300,2.6300|2.6294|None
2012-11-02 12:45:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 13:00:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 13:15:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 13:30:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 13:45:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6300|2.6292|None
2012-11-02 14:00:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6300|2.6292|None
2012-11-02 14:15:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6300|2.6292|None
2012-11-02 14:30:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6300|2.6292|None
2012-11-02 14:45:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 15:00:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 15:15:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 15:30:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 15:45:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6300,2.6300|2.6294|None
2012-11-02 16:00:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6300,2.6290,2.6300|2.6294|None
2012-11-02 16:15:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 16:30:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6300,2.6290|2.6292|None
2012-11-02 16:45:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6300|2.6292|None
2012-11-02 17:00:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 17:15:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 17:30:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6300,2.6300|2.6294|None
2012-11-02 17:45:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6300|2.6292|None
2012-11-02 18:00:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 18:15:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6290|2.6290|None
2012-11-02 18:30:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6300|2.6292|None
2012-11-02 18:45:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6290,2.6300|2.6292|None
2012-11-02 19:00:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6300,2.6290|2.6292|None
2012-11-02 19:15:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6290,2.6300,2.6300|2.6294|None
2012-11-02 19:30:01 |OF0,SDd,SF1,ST10,BR9600|+25|2.6290,2.6290,2.6300,2.6290,2.6290|2.6292|None
:%%-  mVetoPosNemo.log    6% L173   (Fundamental)------------------------------------------------
```

$\underbrace{\text{date + time measurement}}$

$\underbrace{\text{5 measurements of the distance}}$



EDELWEISS setup

WALL

Store a text file content in the db

ADVICE:
Don't store documents individually but create a list of documents and store them in one call

```
# import the couchdbkit librairy which allows the communication with the db
import couchdbkit

# create an empty list which will be used to store documents
docs=[ ]

#open the text file containing the useful informations
f=open('path/workfile.txt','r')
for line in f :                         #go over the file line
   #get the date
     line_list=line.split('|')
     date_str=line_list[0].strip(' ')

   #get the 5 measurement values in a list
      val_list = [float(x) for x in line_list[3].split(',')]

   #put these values in an array using the numpy package
      val_np = np.array(val_list)

   #convert the red date into a time object; be careful of time conversion from your time zone to UTC !!!
       date=datetime.datetime.strptime(date_str,"%Y-%m-%d %H:%M:%S")

   #convert the date in unixtime (ADVICE: always store the time in unixtime)
       unix_time=time.mktime(date.timetuple())
```

Store a text file content in the db

```
#create an empty dictionary to store the document
adoc={ }
adoc['aveValue'] = val_np.mean()
adoc['uncervalue'] = math.sqrt(val_np.var())
adoc['unixtime'] = unix_time
adoc['position'] = 'est'
#append the document in the docs list
 docs.append(adoc)
```

Python document == CouchDb document !

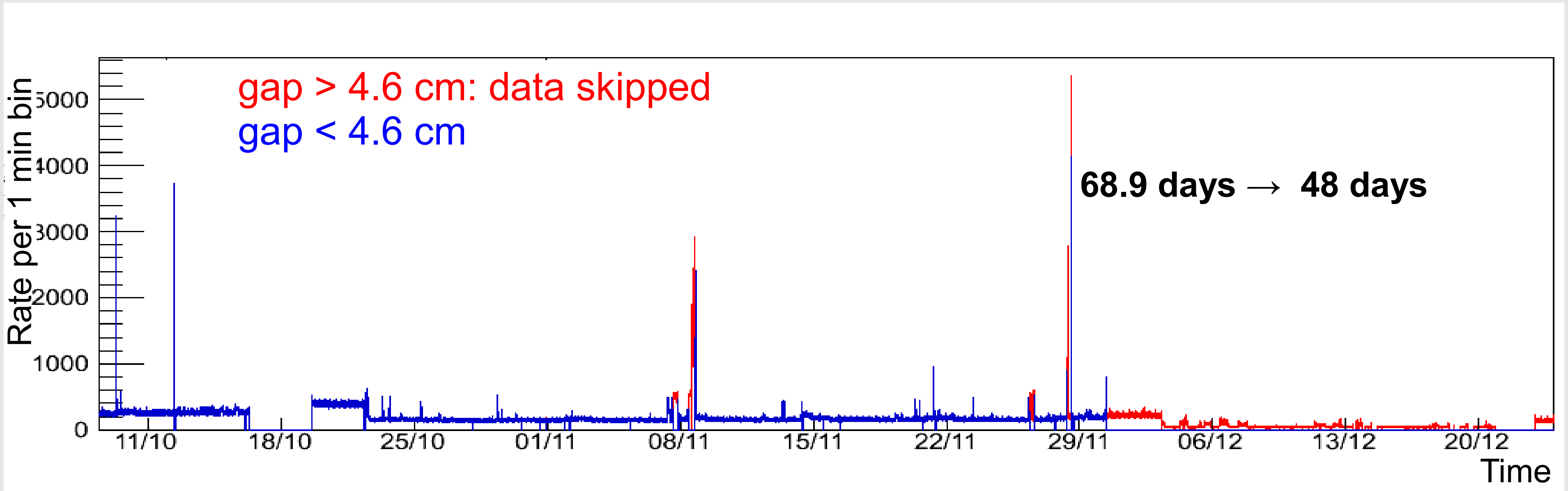
Dictionaries consist of pairs of keys and their corresponding values (like in a db document!)
dict = {'Name': 'Antoine', 'Age': 23, 'Institut':'LASIM'};

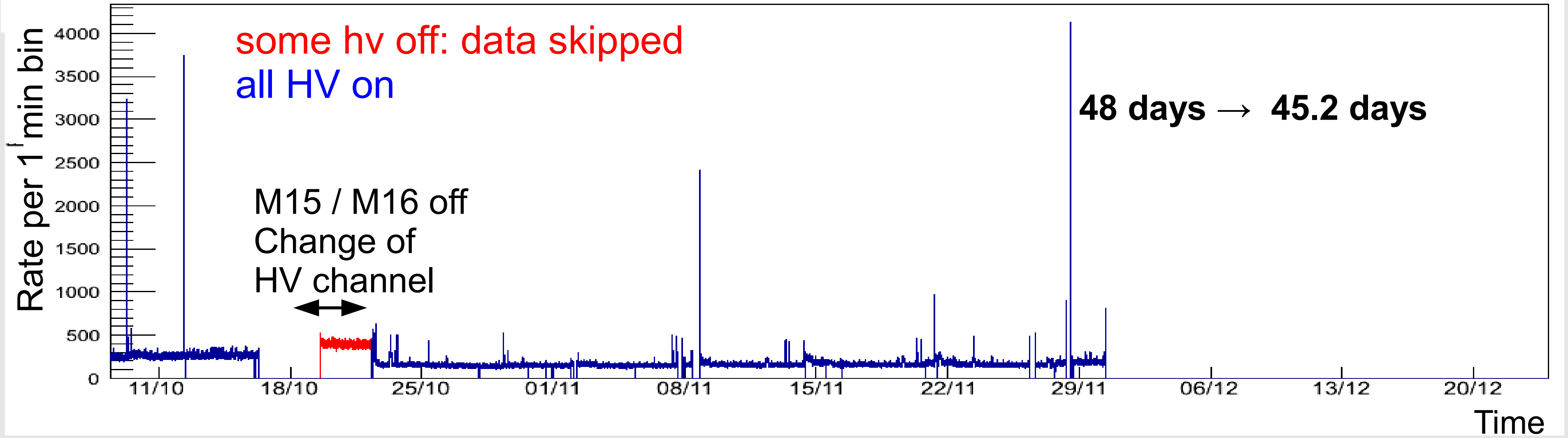
```
#once all the file lines haves been red and informations put in a dictionary,
#call the database and them the list of document:

# connect to the cloudant server
s = couchdbkit.Server('https://username:password@username.cloudant.com')

# create a database with the name “db_name” from a python script
db = s.create_db('dbName')
# call an existing db
db = s('vetopos')
# or create a db if non existing, get the existing one otherwise
db = s.get_or_create_db(dbName)

#save the list of documents append to docs
db.bulk_save(docs)

#to save a single document
#db.save_doc(adoc)
```

Data selection by using the database

- Requirements for a correct analysis for each event of the root tree
  - closed muon veto
  - HV ON for the whole system

- Communication with our Cloudant database





Using views in a python script

Problem: accessing the database for each event is time consuming

Solution: copy the database documents useful the for analysis in a local dictionary

Before to perform the analysis:

```
#create empty lists
DocListEst=[ ]     #to store the position of the est part
DocListNemo=[ ]    #to store the position of the west part

# connect to the cloudant server
s = couchdbkit.Server('https://username:password@username.cloudant.com')

#get to the db called “vetopos”
db=s['vetopos']
```

Beginning of the analysis

End of the analysis

Disabled eventual reduce function

```
#select the view to get position of the est chariot
vr=db.view('app/est_bydate',startkey=StartTime,endkey=EndTime,reduce=False)
#loop over the document of the view
for row in vr:
  #store the useful fields in the dictionary
    DocListEst.append({'PcTime':doc['unixtime'],'Position':doc['aveValue']})

vr2=db.view('app/nemo_bydate',startkey=StartTime,endkey=EndTime,reduce=False)
#loop over the document of the view
for row in vr2:
  #store the useful fields in the dictionary
    DocListNemo.append({'PcTime':doc['unixtime'],'Position':doc['aveValue']})
```

Reducing time consumption while reading documents

If there is only few change of the useful value during the time studied time period (for example of hv): create a reduced dictionary saying when the value changed and the new value

```
#ensure the "Docs" dictionary is sorted by time
Docs.sort(key=lambda x: (x['PcTime']))

#create a reduced list containing the time and new value
ReducedList={ }
ReduceList.append({"time"=Docs[0]["time"],"value":Docs[0]["value"]})

valueRef=Docs[0].get('value')

#loop over the documents in Docs
for item in Docs:
  if item['value']==valueRef:
    print 'no change, don't append the document !'
  else:
     print 'the value has changed. Append the document !'
     ReducedList.append({"time"=Docs[0]["time"],"value":Docs[0]
["value"]})
```

Reducing time consumption while reading documents

If many changes of the useful value: save the list item index of the document in which

```
 for Entry in range (0,t.GetEntries()):
    f.GetEntry(Entry)

     for index in range(save_index,len(Docs)):

       if event.GetPcTimeSec()>=(Docs[index]['time']) and
event.GetPcTimeSec()<(Docs[index+1]['PcTime']):

         save_index=index
        Do stuff
```

More example : delete document from the db

```
connect to the server
s = couchdbkit.Server('https://username:password@username.cloudant.com')
#select the corresponding db
db = s['vetopos']
#select a view
vr=db.view('app/nemo_bydate',reduce=False)

#function(doc) {
#  if(doc.unixtime && doc.position=="est" && typeof(doc.aveValue) === 'number')
#     emit(doc.unixtime, doc.aveValue);
#}
for(line in vr):
  if row['key']>1354589540 and row['key']<1354599999:
    db.delete_doc(row['id'])
```

Database utilities

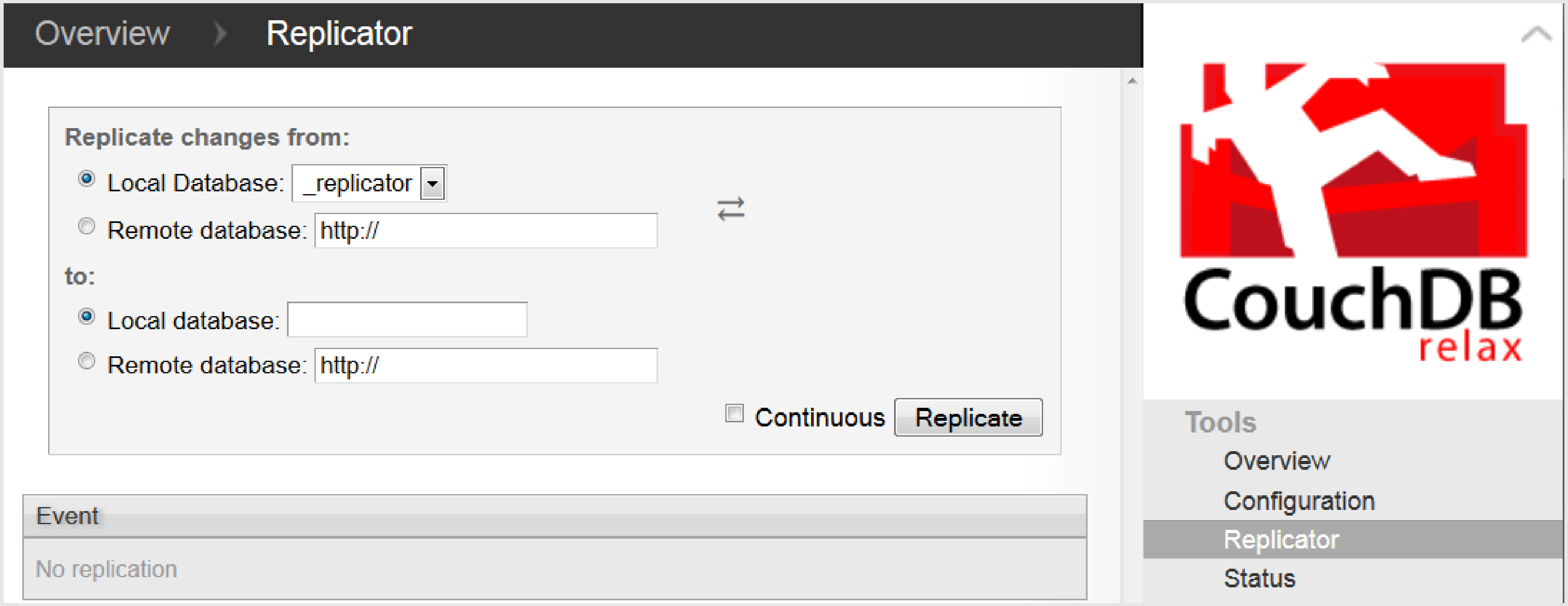
Database replication

• Replication: synchronization of 2 copies of the same database, allowing easy access to data

• The databases can live on the same or different servers. If one copy of the database is changed, replication will send these changes to the other copy.

• To do a replication, the user sends an HTTP request to CouchDB that includes a source and a target database, and CouchDB will send the changes from the source to the target.

• Simple replication from the Futon interface



Replicate changes from:

(●) Local Database: _replicator

( ) Remote database: http://

to:

(●) Local database:

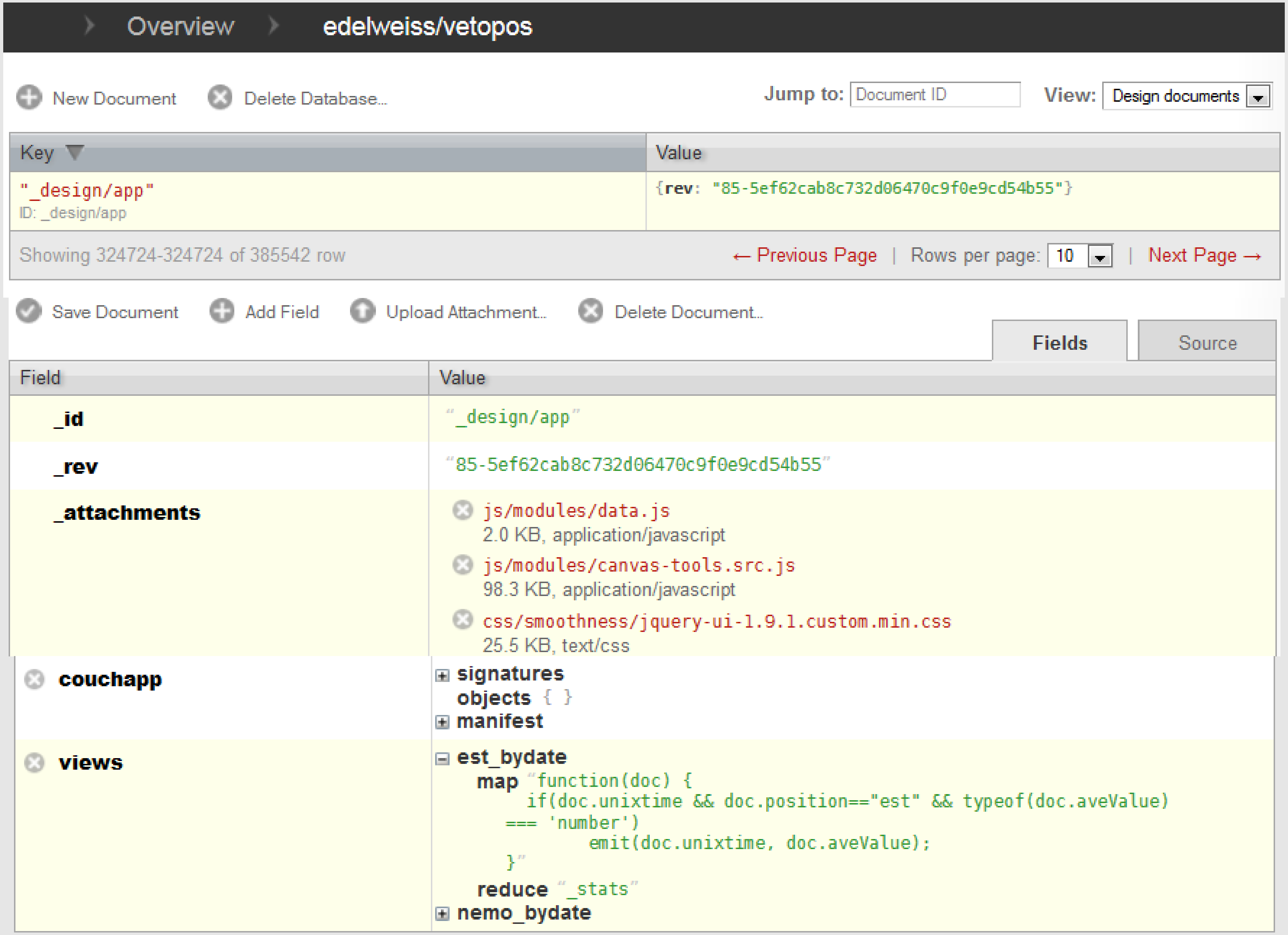
( ) Remote database: http://

☐ Continuous [Replicate]

Event
No replication

CouchApps

CouchApp: standalone web application (based on HTML and JavaScript) that can be entirely self-contained in a design document within the database that provides the data



Jump to: Document ID View: Design documents

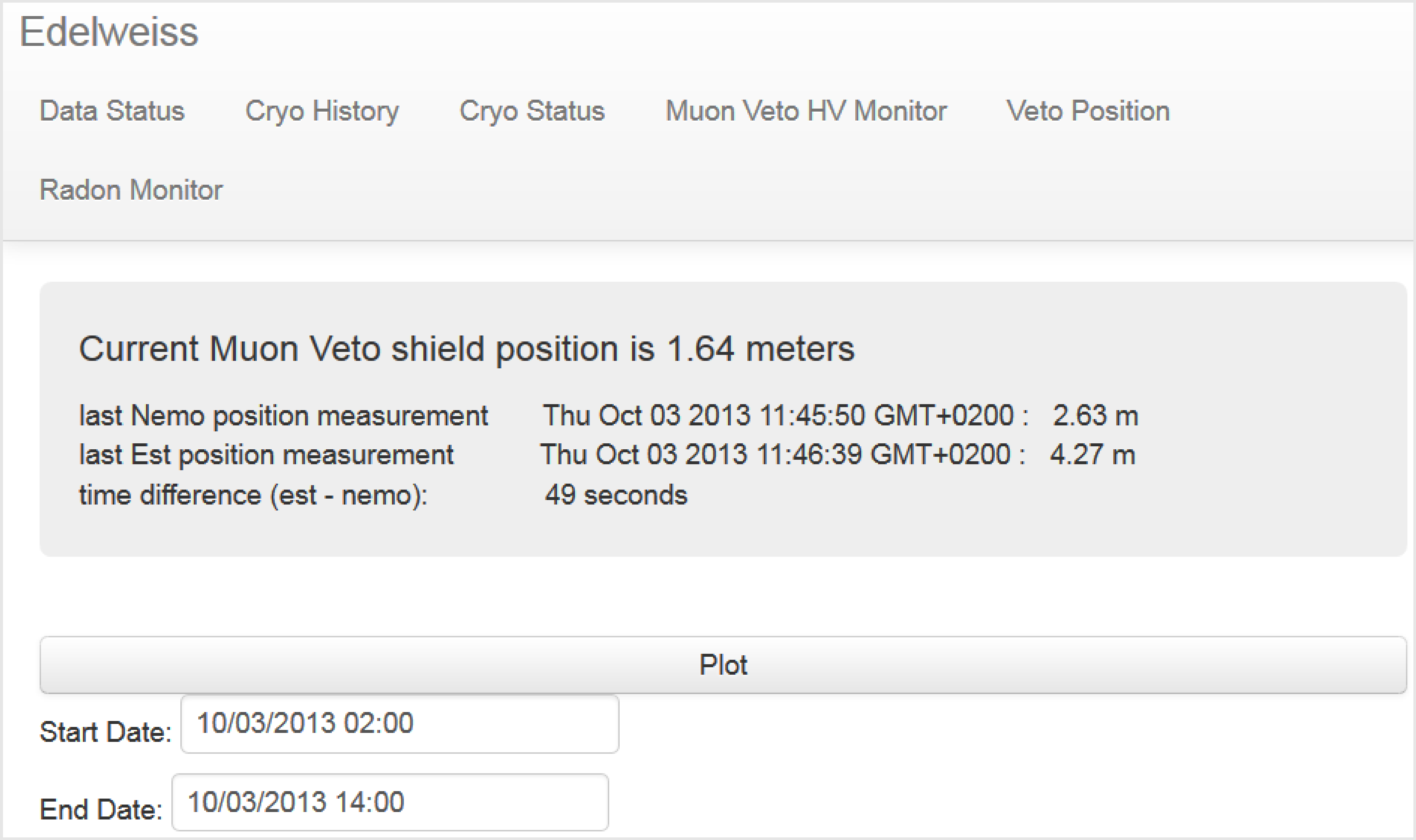
Key	Value
"_design/app" ID: _design/app	{rev: "85-5ef62cab8c732d06470c9f0e9cd54b55"}

Showing 324724-324724 of 385542 row ← Previous Page | Rows per page: 10 | Next Page →

Field	Value
_id	"_design/app"
_rev	"85-5ef62cab8c732d06470c9f0e9cd54b55"
_attachments	js/modules/data.js 2.0 KB, application/javascript; js/modules/canvas-tools.src.js 98.3 KB, application/javascript; css/smoothness/jquery-ui-1.9.1.custom.min.css 25.5 KB, text/css
couchapp	⊞ signatures; objects { }; ⊞ manifest
views	⊟ est_bydate (see below); ⊞ nemo_bydate

```
⊟ est_bydate
    map "function(doc) {
        if(doc.unixtime && doc.position=="est" && typeof(doc.aveValue)
      === 'number')
              emit(doc.unixtime, doc.aveValue);
      }"
    reduce "_stats"
⊞ nemo_bydate
```

CouchApp example



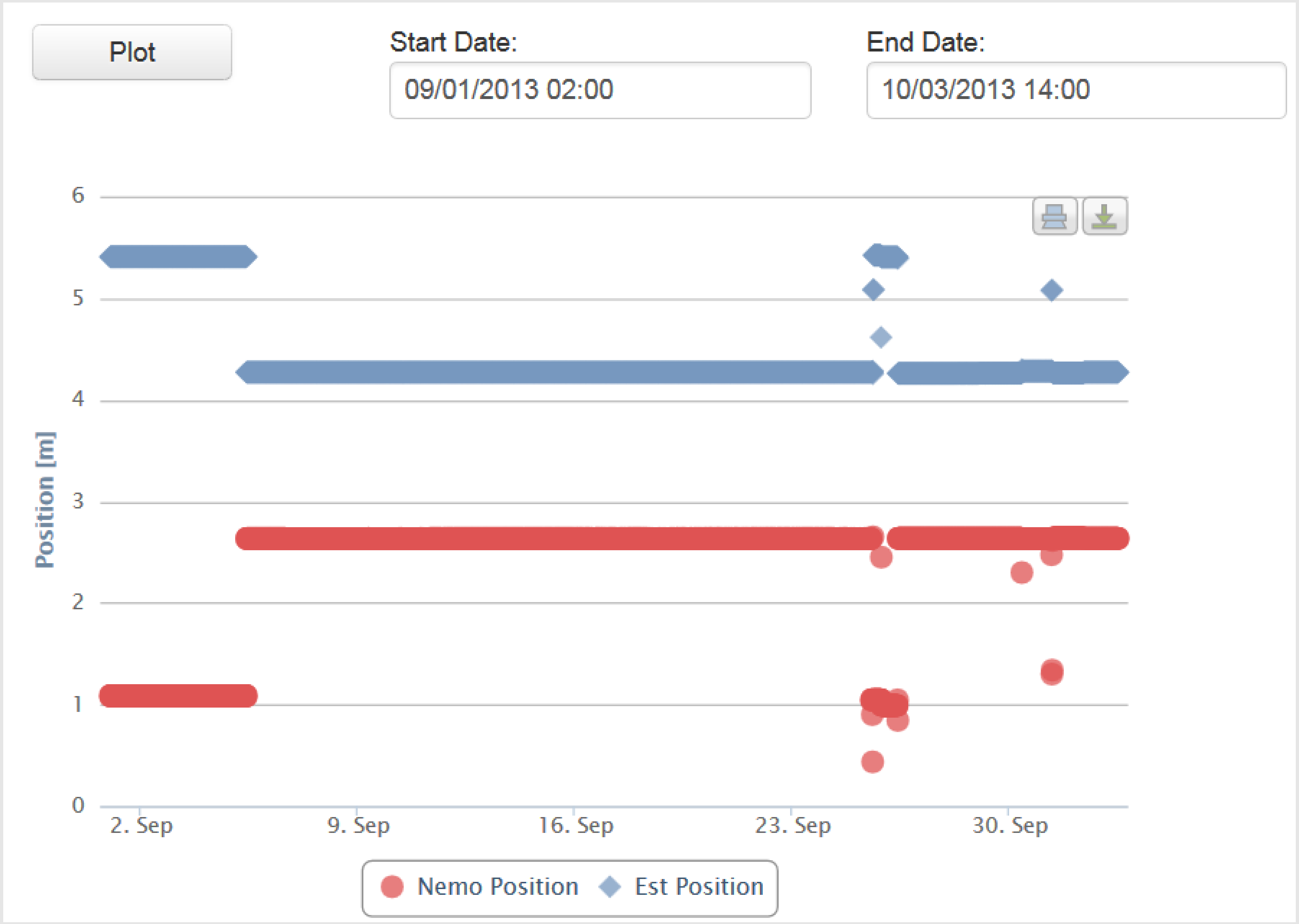
Current Muon Veto shield position is 1.64 meters

last Nemo position measurement	Thu Oct 03 2013 11:45:50 GMT+0200 : 2.63 m
last Est position measurement	Thu Oct 03 2013 11:46:39 GMT+0200 : 4.27 m
time difference (est - nemo):	49 seconds

Start Date: 10/03/2013 02:00

End Date: 10/03/2013 14:00

CouchApp example



Plot

Start Date: 09/01/2013 02:00

End Date: 10/03/2013 14:00

Documentation

Official wiki page
http://wiki.apache.org/couchdb/Documentation

Official apache couchdb website
http://couchdb.apache.org/

Online book dedicated to CouchDB
http://guide.couchdb.org/

Short video tutorial
http://www.youtube.com/watch?v=7ZJCD16sWw4

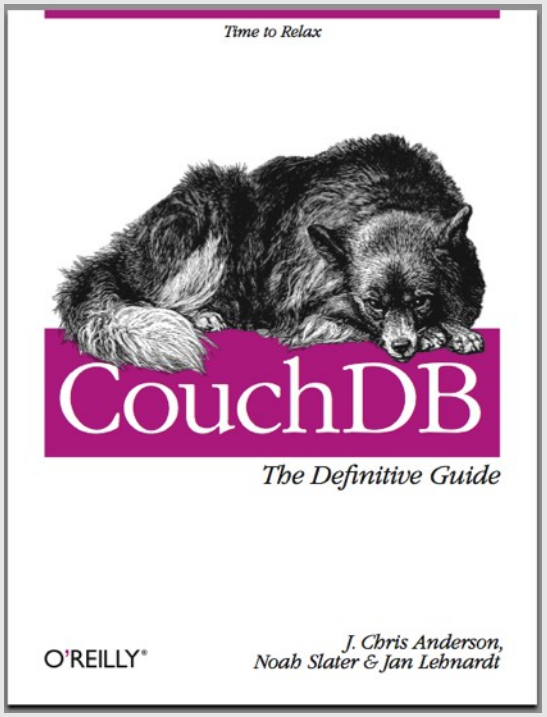
Couchdbkit toolkit
http://couchdbkit.org/

About views:
http://wiki.apache.org/couchdb/HTTP_view_API
http://wiki.apache.org/couchdb/Introduction_to_CouchDB_views

About predefined reduce functions:
http://wiki.apache.org/couchdb/Built-In_Reduce_Functions

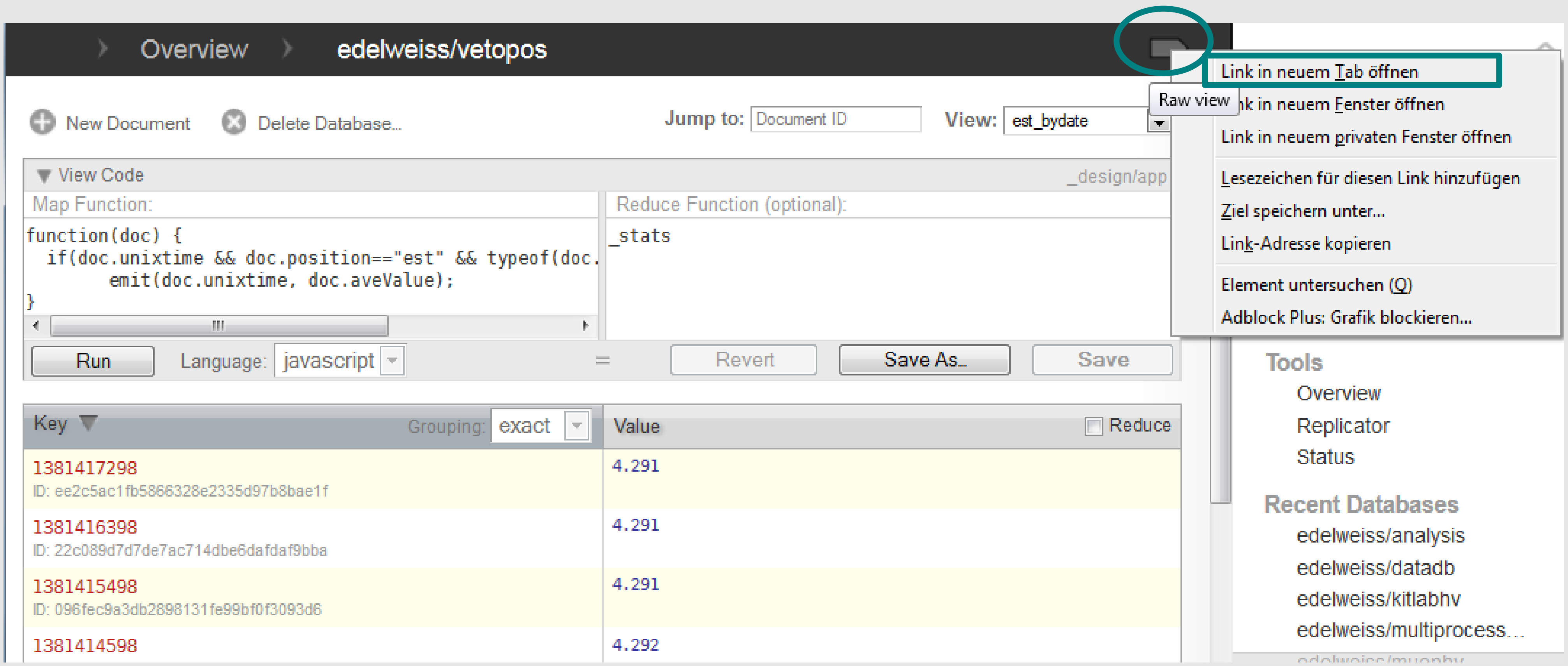
Couchapps toolkit
https://github.com/couchapp/couchapp

Nice tutorial on couchapps
http://www.ibm.com/developerworks/opensource/tutorials/os-couchapp/



Backup slides

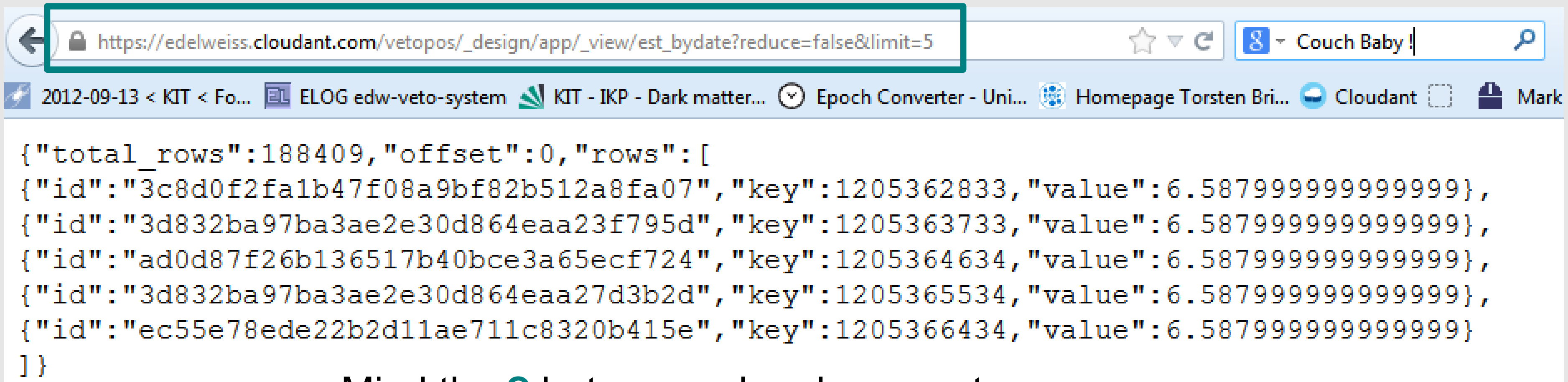
Getting informations using HTTP



```
function(doc) {
  if(doc.unixtime && doc.position=="est" && typeof(doc.
       emit(doc.unixtime, doc.aveValue);
}
```

```
_stats
```

Key (Grouping: exact)	Value
1381417298 ID: ee2c5ac1fb5866328e2335d97b8bae1f	4.291
1381416398 ID: 22c089d7d7de7ac714dbe6dafdaf9bba	4.291
1381415498 ID: 096fec9a3db2898131fe99bf0f3093d6	4.291
1381414598	4.292



https://edelweiss.cloudant.com/vetopos/_design/app/_view/est_bydate?reduce=false&limit=5

```
{"total_rows":188409,"offset":0,"rows":[
{"id":"3c8d0f2fa1b47f08a9bf82b512a8fa07","key":1205362833,"value":6.587999999999999},
{"id":"3d832ba97ba3ae2e30d864eaa23f795d","key":1205363733,"value":6.587999999999999},
{"id":"ad0d87f26b136517b40bce3a65ecf724","key":1205364634,"value":6.587999999999999},
{"id":"3d832ba97ba3ae2e30d864eaa27d3b2d","key":1205365534,"value":6.587999999999999},
{"id":"ec55e78ede22b2d11ae711c8320b415e","key":1205366434,"value":6.587999999999999}
]}
```

Mind the **?** between url and parameters

HTTP parameters

Parameter	Value	Default value	Description
key	*key-value*	-	**Must** be a proper URL encoded JSON value
keys	*array of key-values*	-	**Must** be a proper URL encoded JSON array value
startkey	*key-value*	-	**Must** be a proper URL encoded JSON value
startkey_docid	*document id*	-	document id to start with (to allow pagination for duplicate startkeys)
endkey	*key-value*	-	**Must** be a proper URL encoded JSON value
endkey_docid	*document id*	-	last document id to include in the output (to allow pagination for duplicate endkeys)
limit	*number of docs*	-	Limit the number of documents in the output
stale	*ok / update_after*	-	If **stale=ok** is set, CouchDB will not refresh the view even if it is stale, the benefit is a an improved query latency. If **stale=update_after** is set, CouchDB will update the view **after** the stale result is returned. update_after was added in version 1.1.0.
descending	*true / false*	*false*	change the direction of search
skip	*number of docs*	*0*	skip *n* number of documents
group	*true*	*false*	The group option controls whether the reduce function reduces to a set of distinct keys or to a single result row.
group_level	*number*	-	*see below*
reduce	*true / false*	*true*	use the reduce function of the view. It defaults to true, if a reduce function is defined and to false otherwise.
include_docs	*true / false*	*false*	automatically fetch and include the document which emitted each view entry
inclusive_end	*true / false*	*true*	Controls whether the endkey is included in the result. It defaults to true.
update_seq	*true / false*	*false*	Response includes an **update_seq** value indicating which sequence id of the database the view reflects

http://docs.couchdb.org/en/latest/api/database.html

Predefined reduce functions

• _sum just adds up the emitted values, which must be numbers.

• _count counts the number of emitted values. (It's like _sum for emit(foo, 1).) It ignores the contents of the values, so they can by any type.

• _stats calculates some numerical statistics on your emitted values, which must be numbers.

The reduce output is an object that looks like this:

```
{"sum":2,"count":2,"min":1,"max":1,"sumsqr":2}
```

"sum" and "count" are equivalent to the _sum and _count reductions. "min" and "max" are the minimum and maximum emitted values. "sumsqr" is the sum of the squares of the emitted values (useful for statistical calculations like standard deviation).