



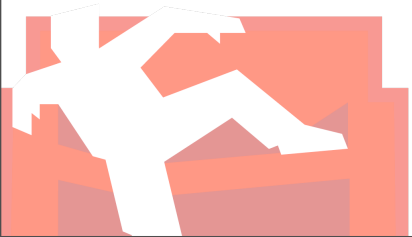
# CouchDB from 10.000ft

with Jan Lehnardt <[jan@apache.org](mailto:jan@apache.org)>

Not the best time for a talk,  
Easy content, just relax

# Why am I here?

- No clue
- Most speakers here
  - are some sort of Erlang expert
  - talk about something Erlangy



# Why am I here?

- Most speakers here
  - are some sort of Erlang expert
  - talk about something Erlangy
- I'm not and I don't



# Maybe it's CouchDB

- Written in Erlang
- You are probably not using it from Erlang
- The HTTP API is as easy in any language



# Maybe it's CouchDB

- Shows what cool things can be done with Erlang
- So here is CouchDB an Erlang Case Study



# What is CouchDB?

- “DB” gives it away, really
- Not a relational database
- But a *Document Database*



# *A Document Database?*

- Concepts borrowed from the Notes Database (NOT from the client!)
- Data is not stored in tables and rows
- No need to design a schema



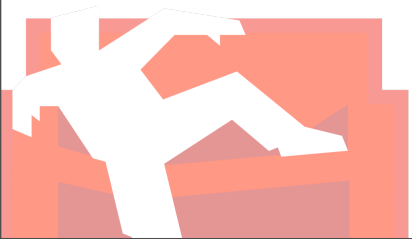
# *A Document Database?*

- Data records are stored as individual *documents*
- Documents can have any structure and the structure can be made up on the fly




# The Documents in CouchDB

- Data stored in JSON format
- JSON stores native language objects in a portable way
- No ORM needed



# A JSON Example

```
{  
  "_id": "123ABC",  
  "_rev": "BCCE34",  
  "name": "Darth Vader",  
  "location": {  
    "name": "Death Star",  
    "street": "Galaxy Street 4",  
    "zip": 56665  
  },  
  "favourite_colour": "black",  
  "tools": ["helmet", "light sabre"]  
}
```



# Working With a Document – The REST API

Create: HTTP PUT /db/docid

Read: HTTP GET /db/docid

Update: HTTP POST /db/docid

Delete: HTTP DELETE /db/docid



HTTP around for ages, simple, proven,  
scales, tools available

# Updating a Document

- [insert gfx showing optimistic locking]

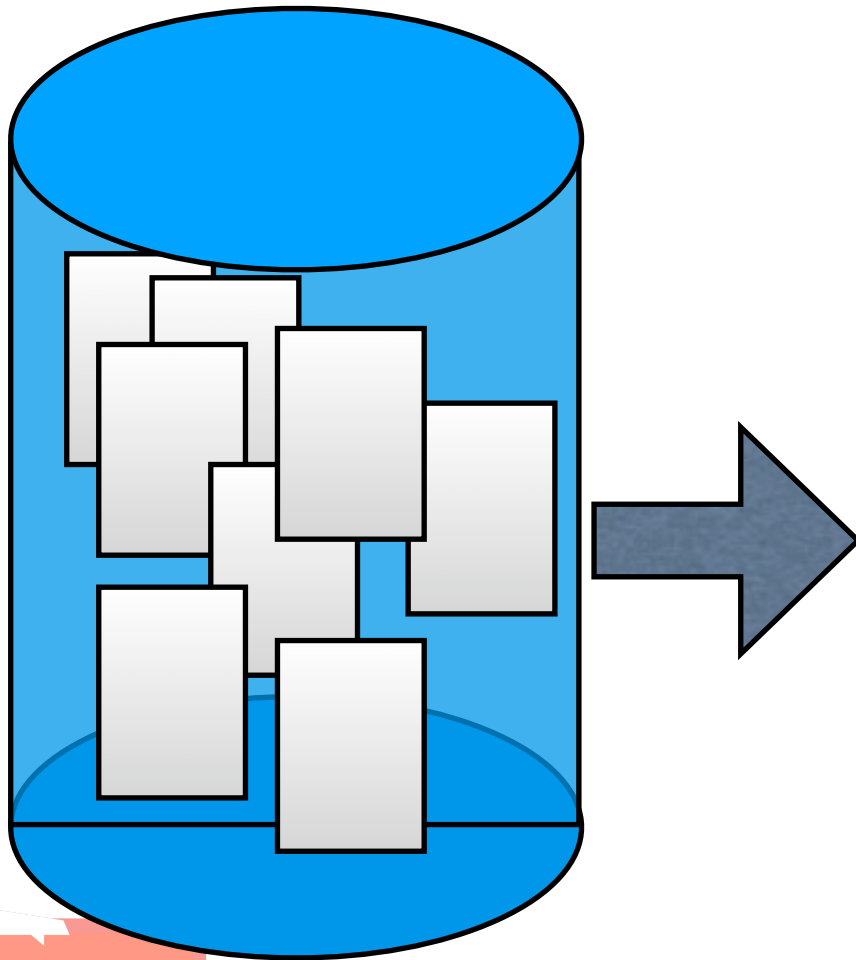


# Views

- “Filter, Collate & Aggregate”
- Bring structure to your data mess
- Map / Reduce powered<sup>1</sup>
- Defaults to JavaScript
- Incremental

<sup>1</sup> Tell audience how awesome it is!

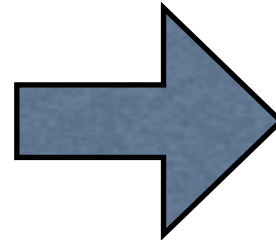
# Views - Map Tags



family	
friends	
friends	
work	
work	
youtube	
...	...

# Views - Reduce Tag Count

family	1
friends	1
friends	1
work	1
work	1
youtube	1
...	...

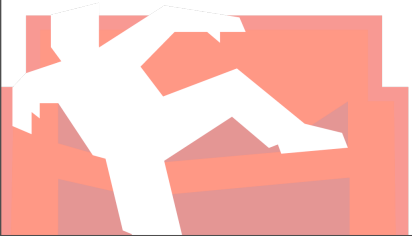


family	1
friends	2
work	2
youtube	1
...	...



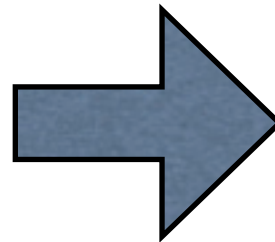
# Views - Map Tags

```
function (doc) {  
  for(var i in doc.tags)  
    emit(doc.tags[i], 1);  
}
```

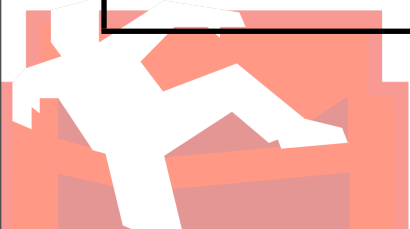


# Views - Reduce Tag Count

family	1
friends	1
friends	1
work	1
work	1
youtube	1
...	...



family	1
friends	2
work	2
youtube	1
...	...



# Views - Reduce Tag Count

```
function (Key, Values) {  
  var sum = 0;  
  for(var i in Values)  
    sum += Values[i];  
  return sum;  
}
```



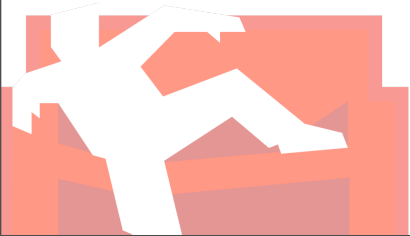
# Replication – The Killer Feature

- Data synchronisation is the core of the solutions to many problems
- Works...
  - Distributed / peer to peer
  - Offline



# Replication – The Killer Feature

- (continued)
- Works...
  - With any number of nodes
  - Automatic conflict detection and resolution(!)



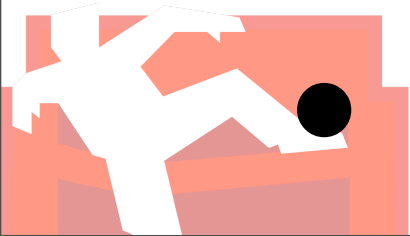
# Inherits all of Erlang's Coolness

- Massive concurrency
- Fault tolerance
- Less code
- No downtime
- Excellent hardware utilisation



# The Storage Engine

- [my pet detail, but bear with me, it is cool!]
- ACID
- MVCC
- Append only
- Compaction



# Search API

- Lucene as a reference
- Works with any other search technology
- View engine works similar.  
Not a JavaScript fan? Use  
XYZ



# The Project

- Damien Katz is the man!
- Open Source
- Apache 2.0
- The Apache Software Foundation gives shelter in case IBM plays bad

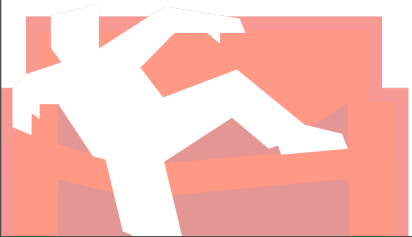


# Questions?

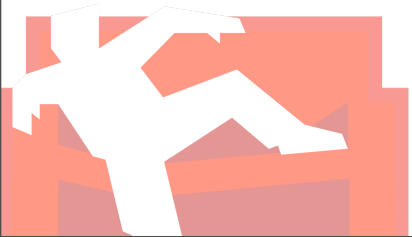


# Questions?

- I prepared two

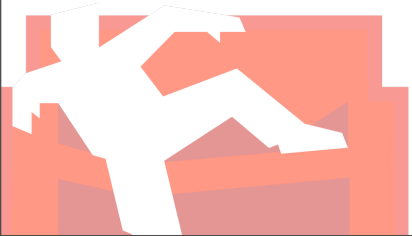


# One – Does it use Mnesia?



# One – Does it use Mnesia?

- No



# Two – Why?



# Two – Why?

- 2GB limit (at the time of research)
- Consistency check on restart not acceptable



# Two – Why?

- Mnesia Replication is good for clusters, not so much for disconnected and distributed edits
- Other “cool” feature not useful to CouchDB

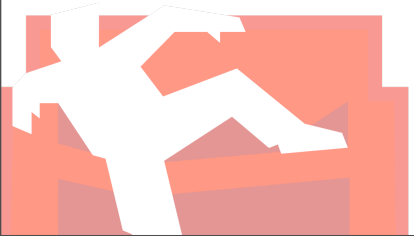


# Two – Why?

- CouchDB's storage engine is optimised – down to hard drive head seeks – for fast writes and super-fast index creation



# Your Questions?



# Bonus Slide

- 6k lines of Erlang
- A few lines of C code

