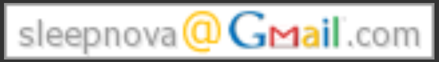
The NoSQL movement

CouchDB as an example

About me

sleepnova - I'm a freelancer 

sleepnova@Gmail.com

Interests:
emerging technology, digital art
web, embedded system, javascript, programming language

Some of my works:
Chrome 小字典
Android app 呼叫小黃

Text file (good old days)

We are all happy with text files

- You already know the API
- Use existing text tools
- Talk directly to the text editor

Update might needs to shift all data
Need to scan to find the record you want
Just can't scale to handle large datasets!

Adding constraints on records/fields

Fixed field/record length
Sorted
Easy to lookup by id (offset = id * length of record)
Update In-place (each row can be modify independently without affecting each other)

Data expanded
Search is still painful

Indexing

Index of search term

ex. index of record No.
0, 0
1, 10
2, 20
3, 35
...

Shorter path to the data
Update/delete needs to rebuild indexes. (expensive!)

Keep evolving...

Store typed binary data to reduce data size and IO
Smarter indexing mechanism (B+/-Tree)
Eliminate redundance to save storage

- Much like refactoring your code
- Toward data normalization

How about data integrity, consistency, rejoin normalized data and transaction?

There comes the Relational Database

Relational model
SQL standards for query, rejoin...
Data schema
  - Integrity check...

Transaction control
  - Isolation level
  - Atomic operation

Which solves many problems above!

Wall again...

Scalability

- Transaction lock (isolation level)
- Synchronization latency

Resistance

- Model mismatch
  - Object-Relational mapping (OR mapping, ORM)
- Schema migration

If you lock too much, users end up waiting all the time!
Static schema doesn't work well in reality, it evolves over time!

CAP theorem

Consistency

- All database clients see the same data, even with concurrent updates.

Availability

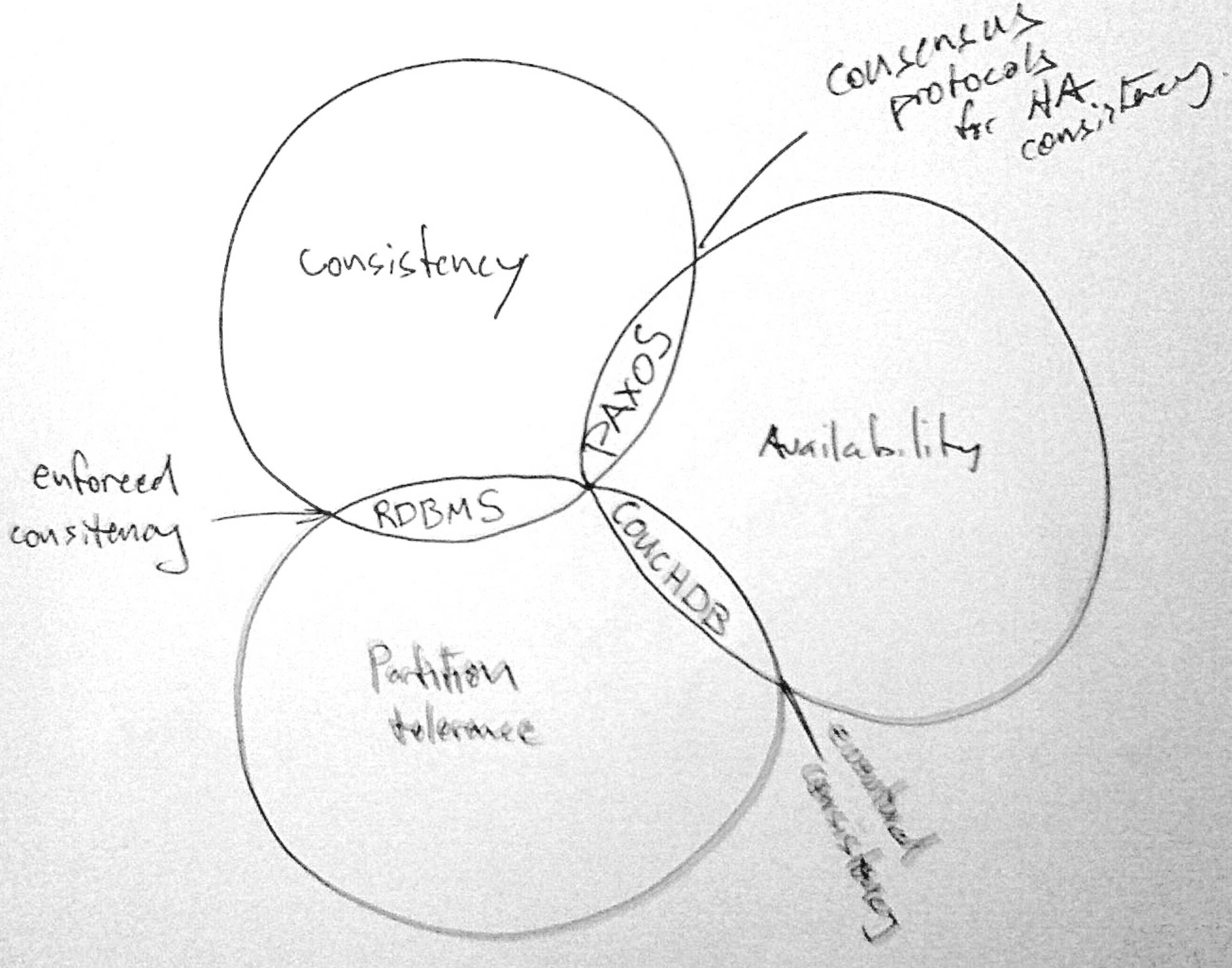
- All database clients are able to access some version of the data.

Partition tolerance

- The database can be split over multiple servers.

Pick two.

THE CAP THEOREM



The NoSQL movement

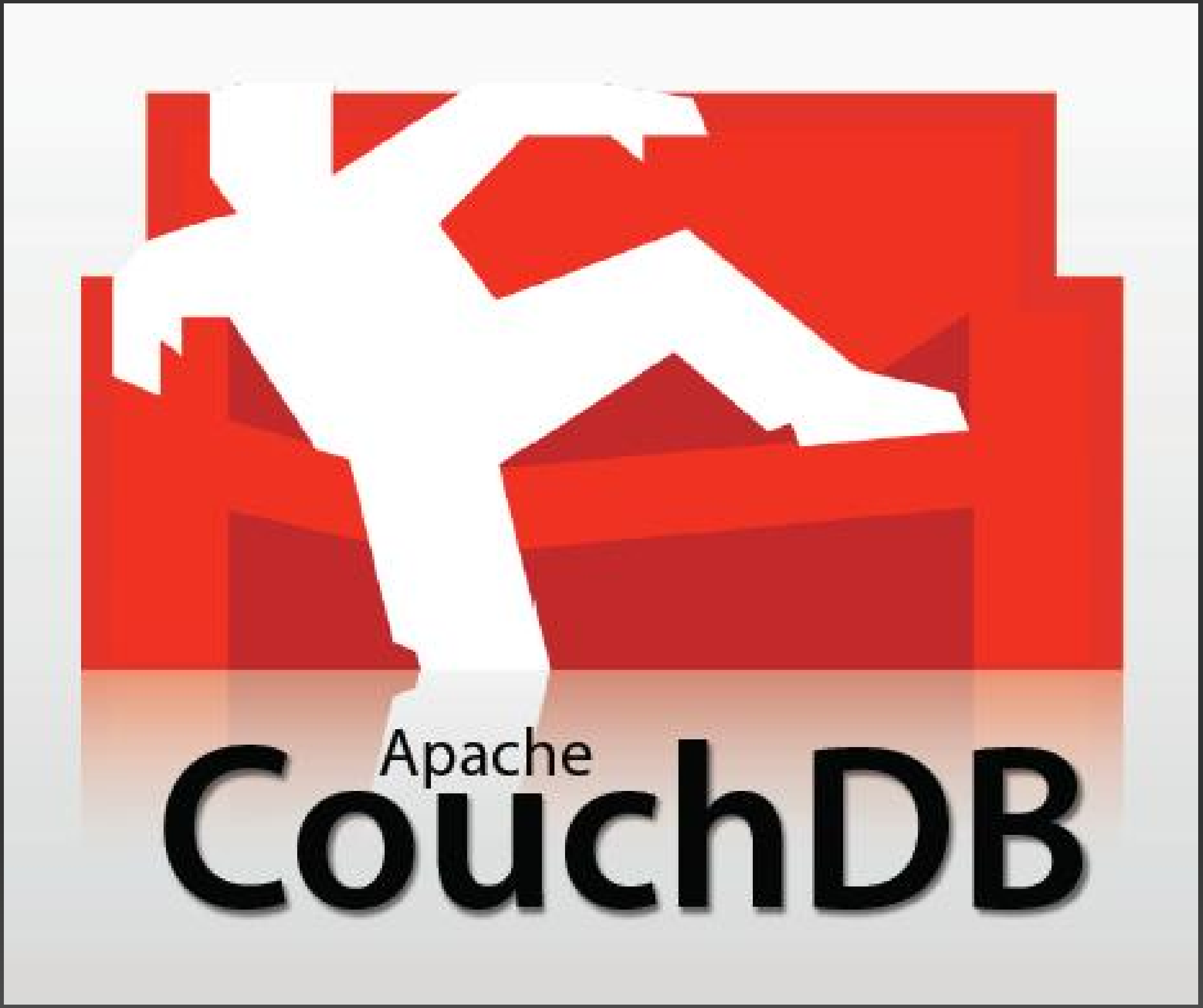
"Not only SQL" - some said.

So now we have

- key-value database
- document database
- graph / network database

NoSQL is about relaxing constraints to give you more options for your context. Giving the controls back so you can do whatever you want with your data with less resistance.

I think it's nothing serious about SQL, we just use this term to refer to the old decisions.


Apache
CouchDB

Introduction CouchDB

If there’s one phrase to describe CouchDB it is **relax**.

Let me tell you something: Django may be built for the Web, but CouchDB is built of the Web. I’ve never seen software that so completely embraces the philosophies behind HTTP. CouchDB makes Django look old-school in the same way that Django makes ASP look outdated. - Jacob Kaplan-Moss

RESTful HTTP

- You already know the API
- Use existing HTTP tools
- Talk directly to the browser

A new era again! :)

RESTful HTTP (CRUD)

Create

- HTTP PUT /db/mydocid

Read

- HTTP GET /db/mydocid

Update

- HTTP PUT /db/mydocid

Delete

- HTTP DELETE /db/mydocid

Document Oriented (JSON)

```
{
  "_id": "COSCUP / GNOME.Asia 2010",
  "_rev": "9-0830646cdcea8835eef54e531fd35e19",
  "date": [2010, 8, 15],
  "at": "Academia Sinica, Taipei, Taiwan",
  "url": {
     "zh-tw": "http://coscup.org/2010/zh-tw",
     "en": "http://coscup.org/2010/en"
  }
}
```

Document Oriented

With _ID(uuid) and _REV(revision)

Real world document behavior

- Bills, letters, tax forms...
- Natural data behavior

Self contained

Schema-less

Atomic operation at document level

Cache-ability

Eventual Consistency

MapReduce View Definition (Indexed)

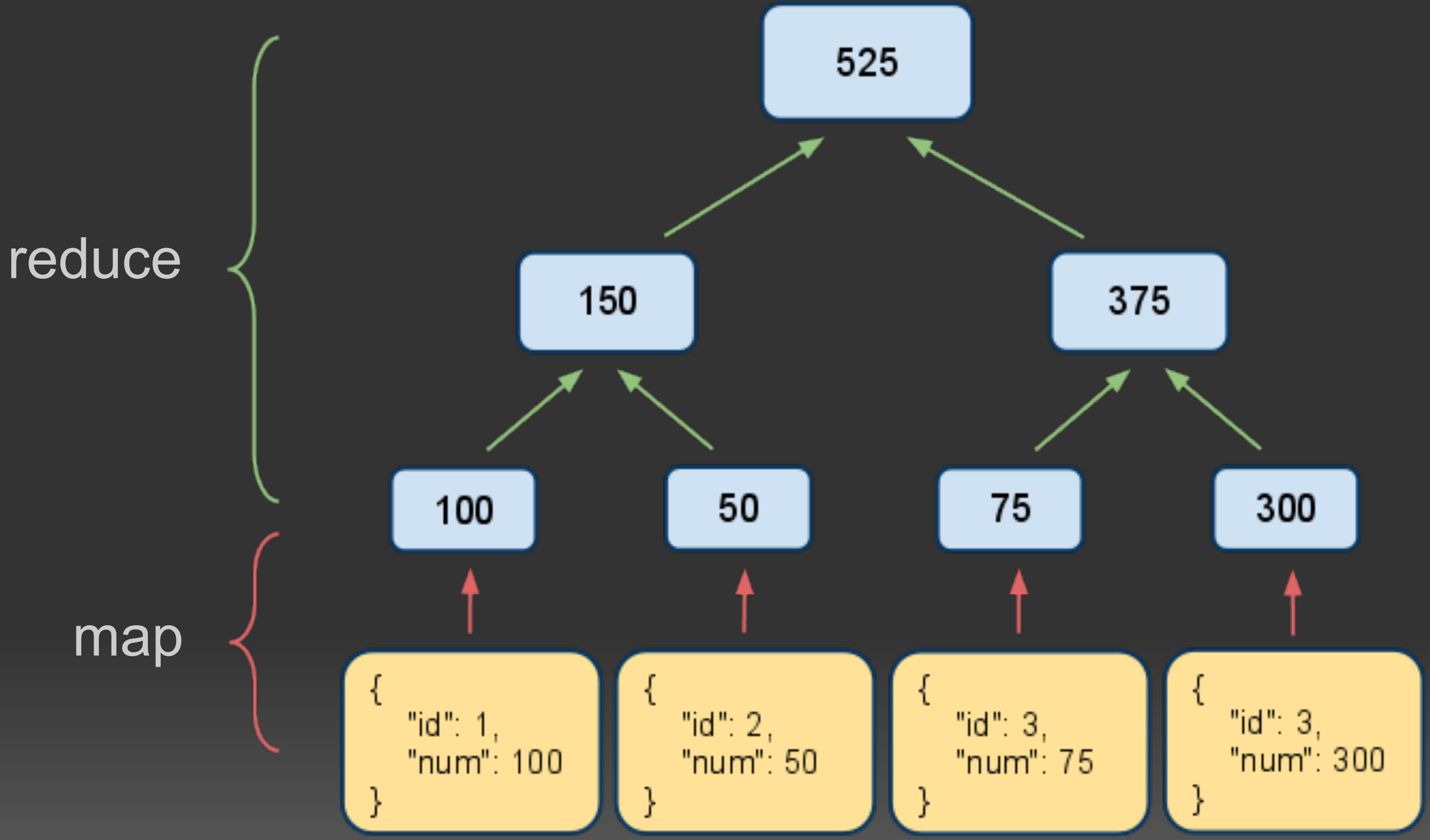
How to query without a query language?
Create view with MapReduce functions in Javascript

ex. summing doc.num up

```
{
  "map":"function(doc){ emit(null, doc.num); }",
  "reduce":"function(key, values){ return sum(values); }"
}
```

Bring function close to data, bring results close you!

MapReduce



```
{
    "id": 1,
    "num": 100
}
```

```
{
    "id": 2,
    "num": 50
}
```

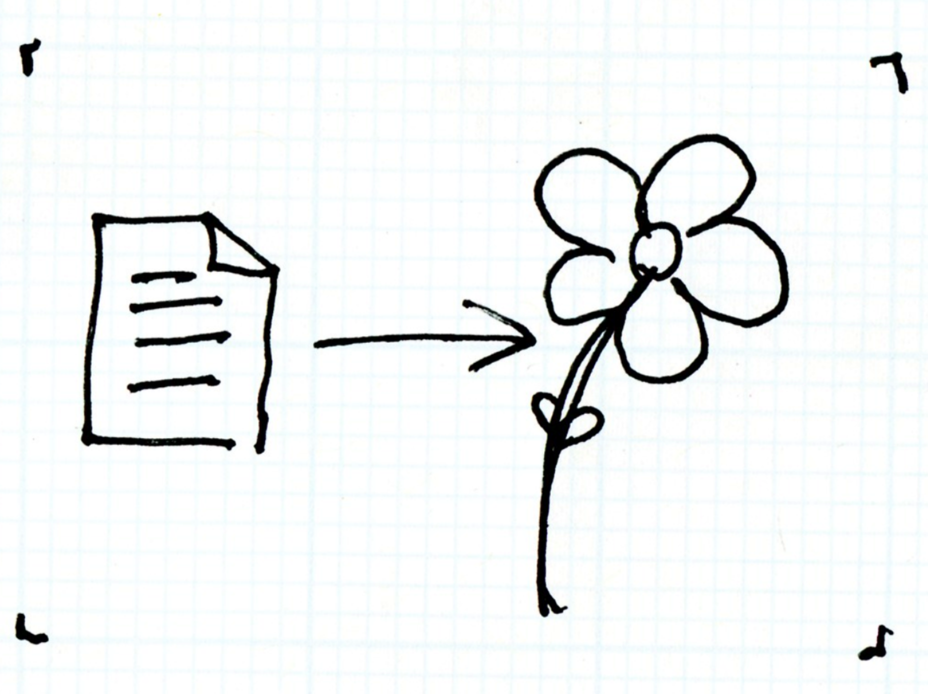
```
{
    "id": 3,
    "num": 75
}
```

```
{
    "id": 3,
    "num": 300
}
```

Applications are documents

Design documents
Two tier web application (CouchApp)
Show function

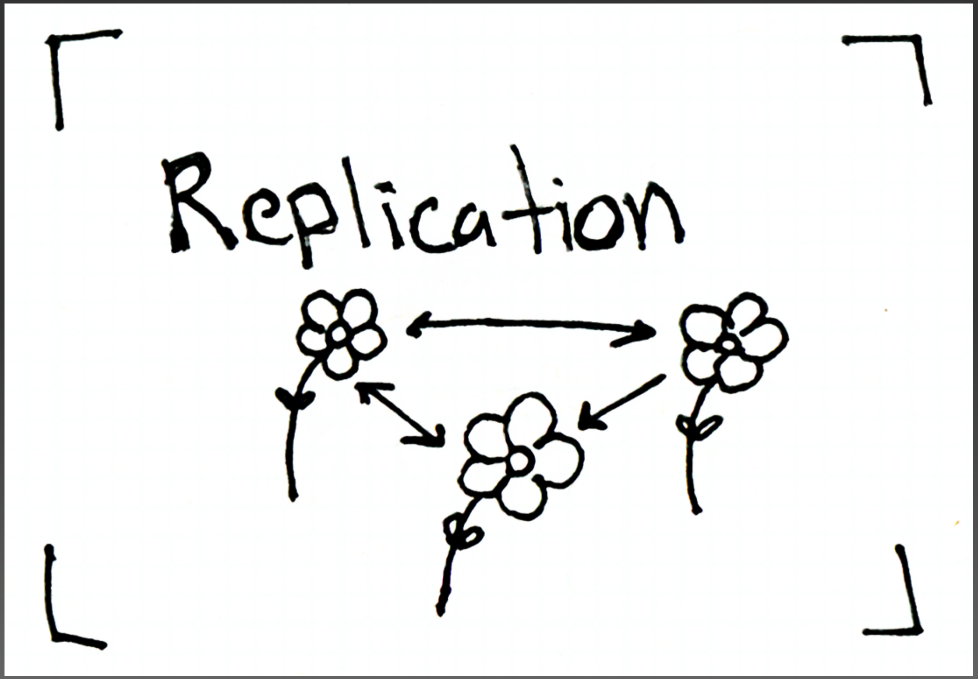
- ○ Different presentation for different HTTP content-type
- ○ Javascript render function :D



Master-Master Replication

Means for synchronize between CouchDB nodes
Each node working independently offline while become one when online
Other CouchDB enabled devices

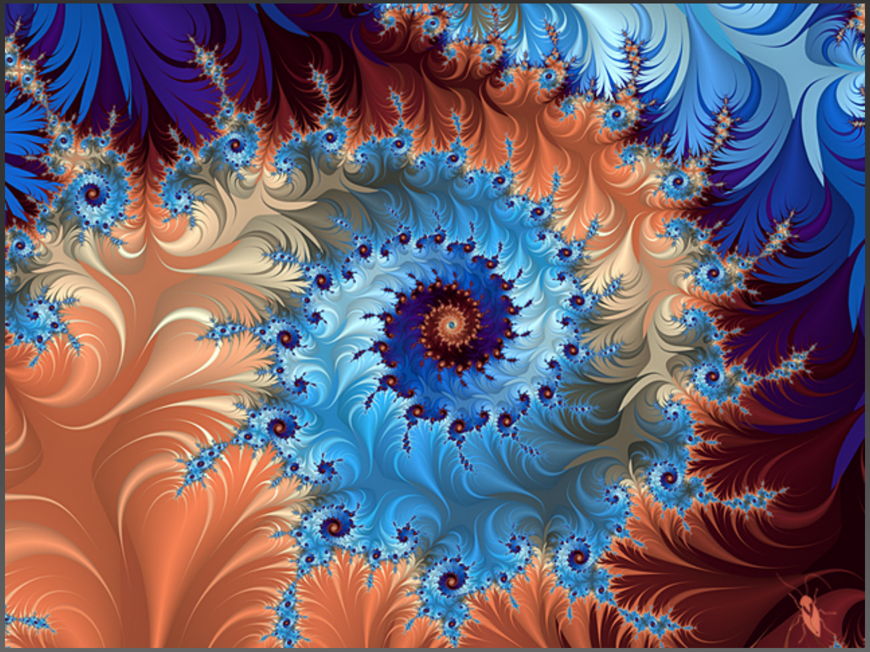
- iPhone
- Andorid
- Browser (Web Storage)



Other Stunning Features

Append only
  - Once written, never touch the data again (robustness)
  - No fix-up phase after a crash
  - Reduce disk seek on write

Change notifications (Comet push)
Fractal scaling (CouchDB Lounge)



I Use Couch DB

A rap by CouchDB team
http://vimeo.com/11852209

Is NoSQL Really Non-relational?

Q: Does that mean my data are going to be non-relational? How can I do things without relations!

A: Well, No!

It only means the database does not force you to describe the relations between your data in a particular way.

In fact, you can have more flexible relations while the database doesn’t add any constraint to it!

Comparing key-value, document and graph database

K-v database is a flat key space storage

- Allows you to put any possible format in it

Document database = k-v storage+ document aware operations

- validation, show, view...etc

Graph/network database

- You can think the keys of k-v db is path/routes to the data in graph db.
- Handles the link/reference and traversing for you.
- Different path/routes can lead to the same object.

Database Trends

JSON format, RESTful architecture
Schema-less, lock free, append only
Much more low-level but easier to start with
Avoid single point of failure
Not a perfect system all the time but always tries it’s best effort to serve you

Thanks!