CouchDB

Relax!

Actually 50 slides for 60 minutes. Good luck.

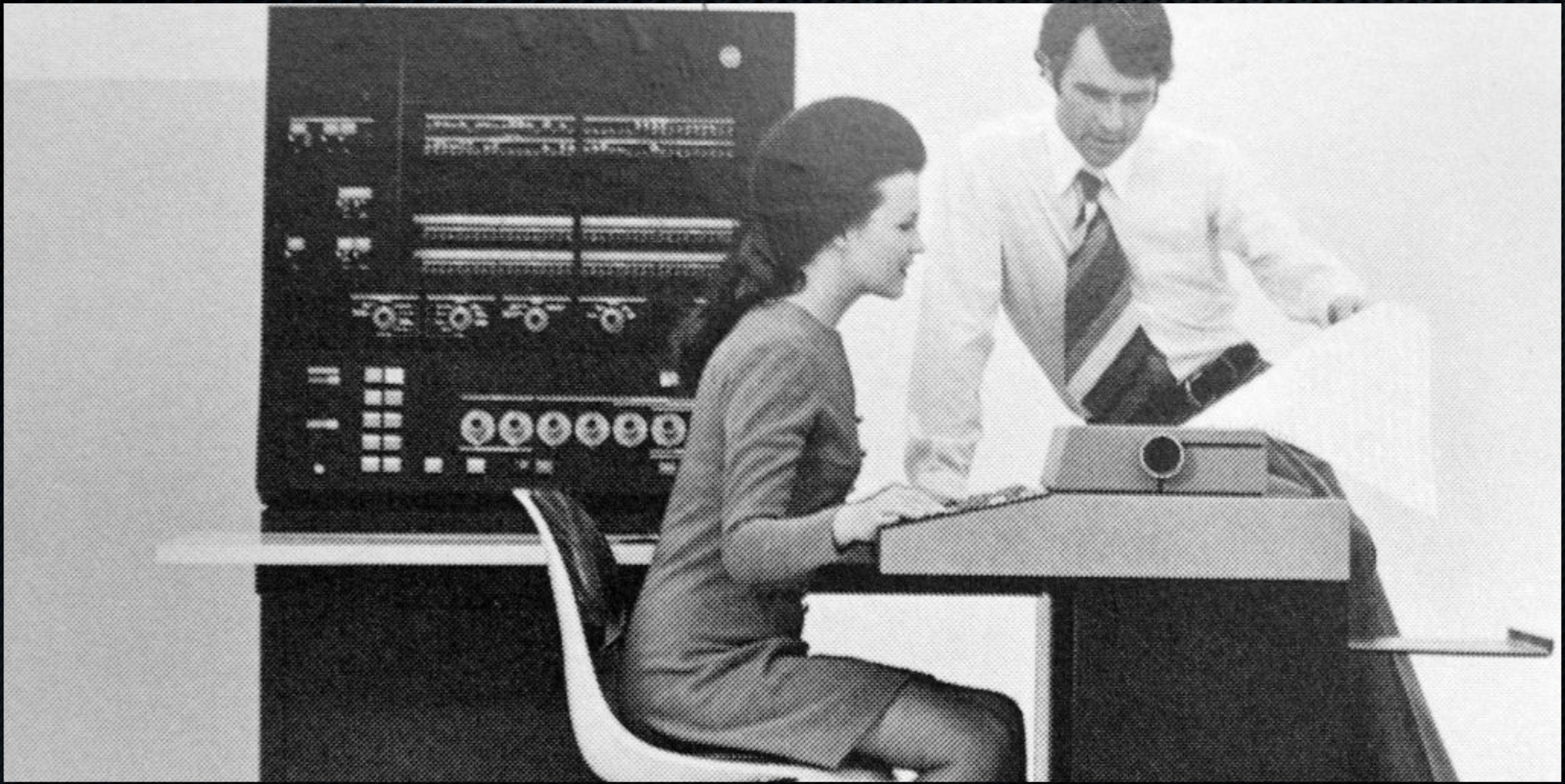
Who’s talking?

- Jan Lehnardt
- CouchDB Developer
- jan@apache.org

And you? Developers, DBAs, architects?
Know CouchDB? Like CouchDB? Use CouchDB?

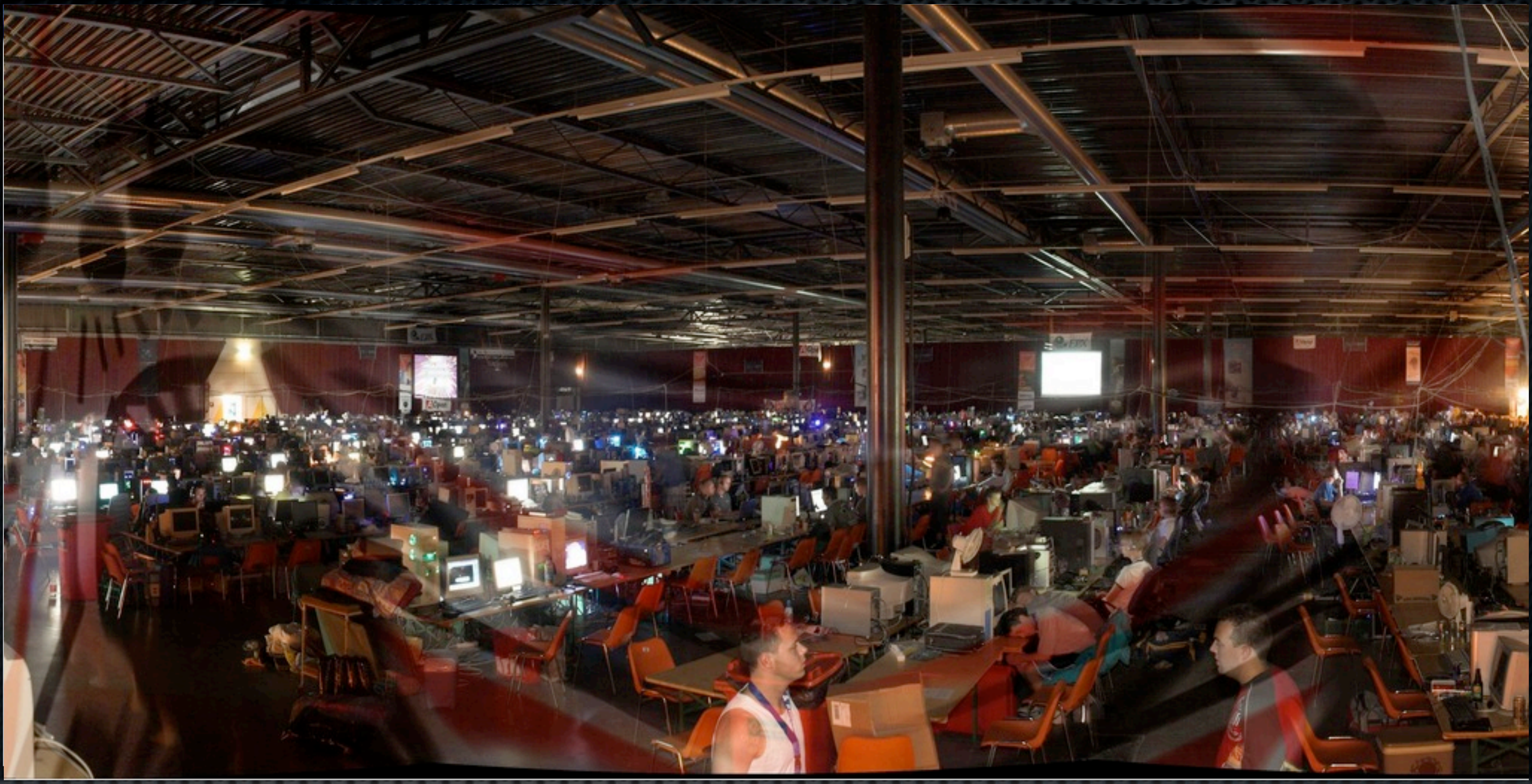
CouchDB –
Built for the Future

“640k processors should be enough for anybody.”



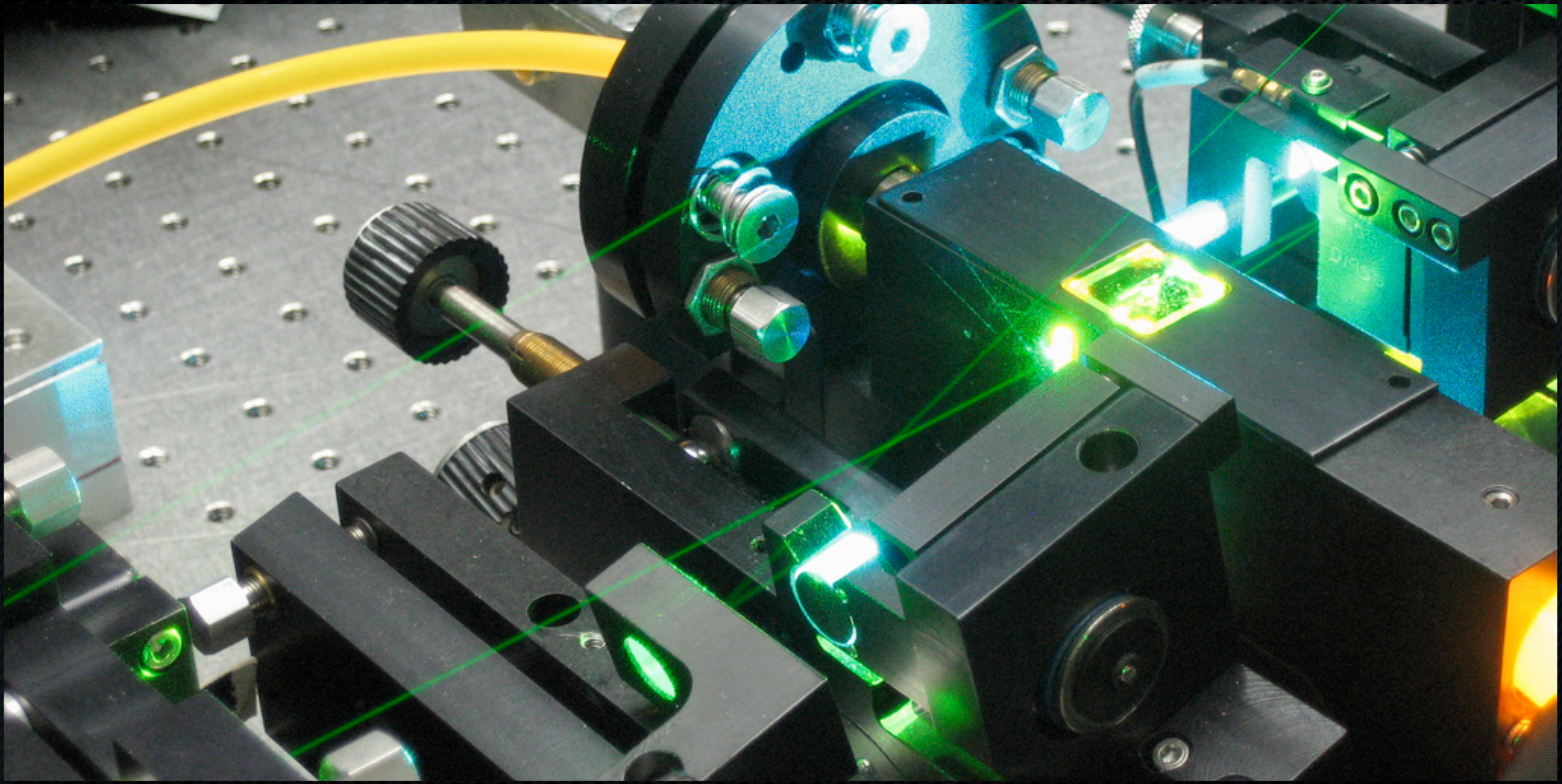
Single-User Machines

Back then



Multi-User Machines

Now



Application: Science

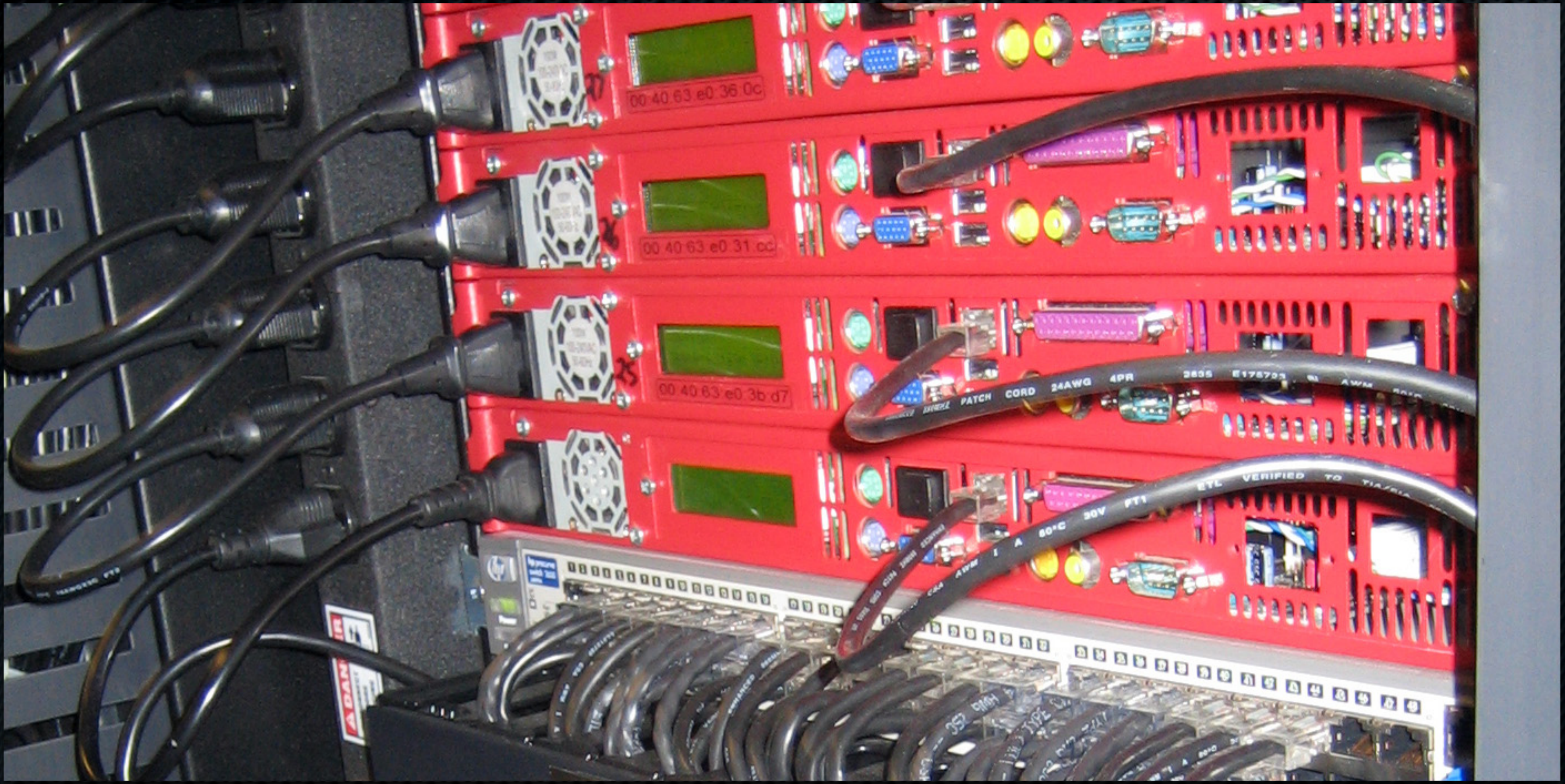
Back then

Application: The Web

Today

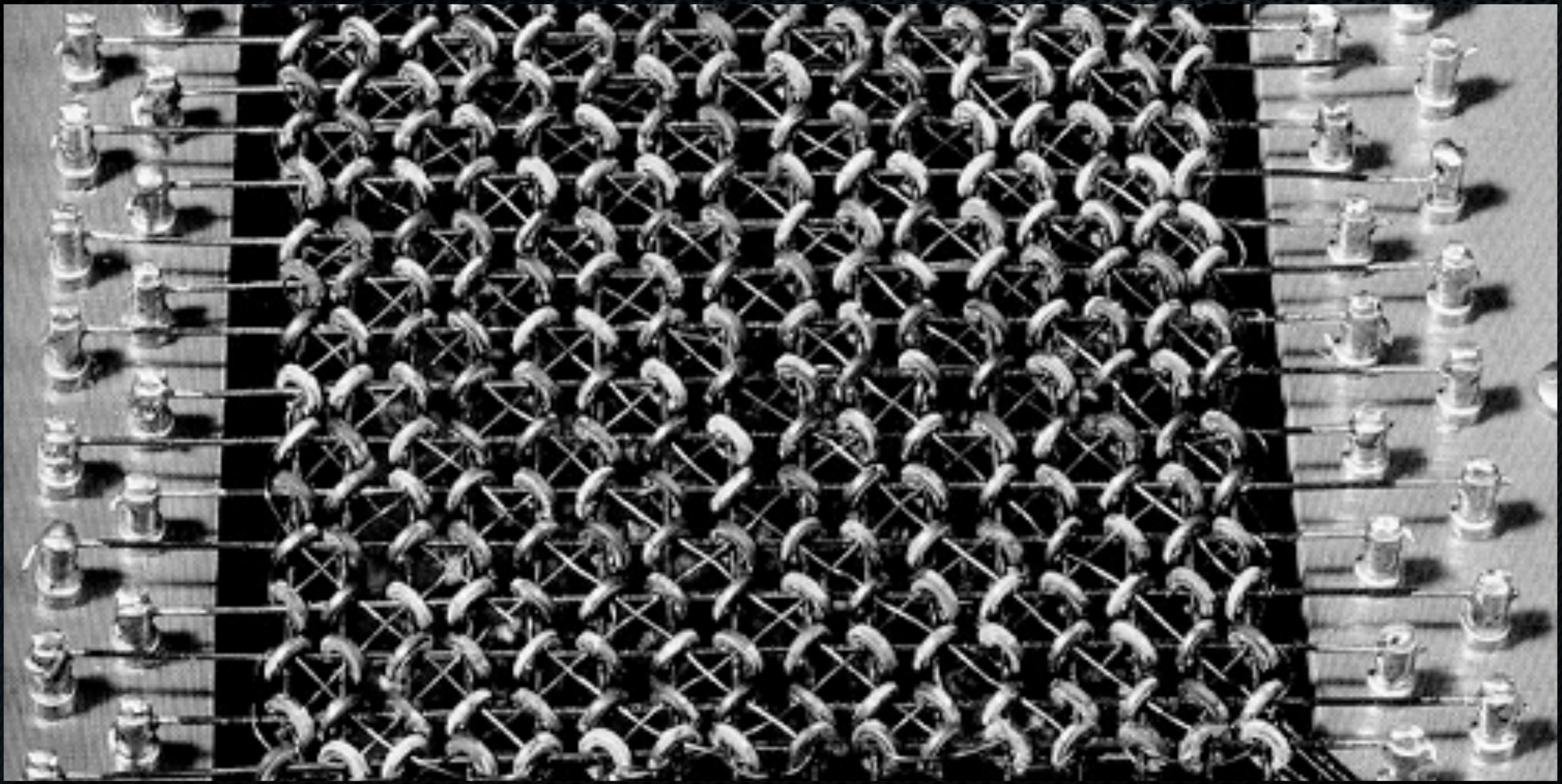
Monolithic Machines

Back then



Lots of Small Servers

Today



CPU, RAM and Disks == $$$

Back then

Components cheaper

Now

RDBMS vs
Just Storing Data

Sorry for bashing!

Real World Data

- Bills, tax forms, letters…
- Same type != same structure
- Can be out of date
- Natural data behaviour

Actual data record, no pointer

RDBMSs

1) beware of speed considerations without having an app to measure
2) or use an ORM which turns out to be a pain in the back for all sorts of reasons
3) Most Data is not inherently relational

RDBMSs

- Design schema upfront

1) beware of speed considerations without having an app to measure
2) or use an ORM which turns out to be a pain in the back for all sorts of reasons
3) Most Data is not inherently relational

RDBMSs

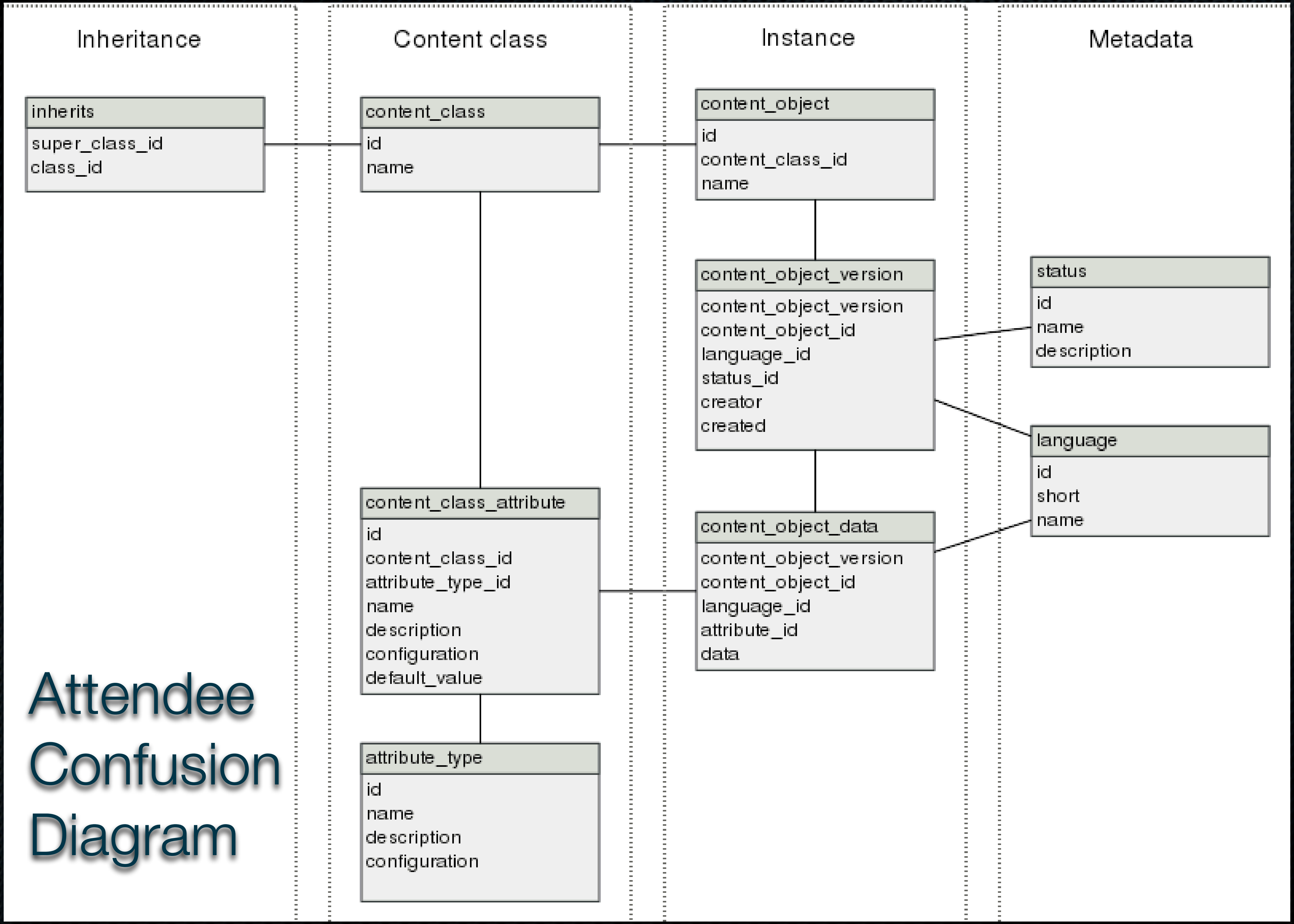
- Design schema upfront
- Write or use software to translate your data into that schema … and back

1) beware of speed considerations without having an app to measure
2) or use an ORM which turns out to be a pain in the back for all sorts of reasons
3) Most Data is not inherently relational

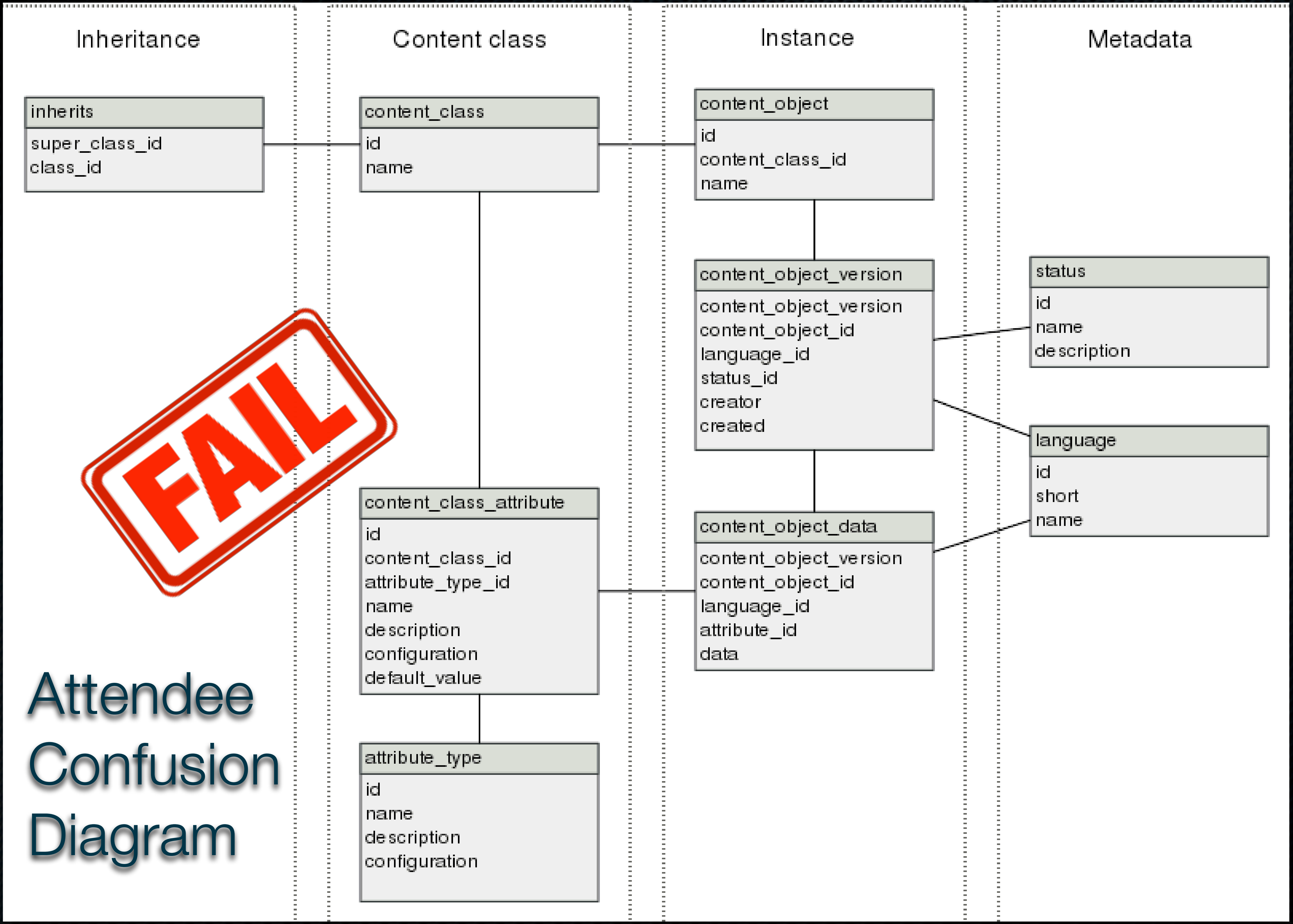
RDBMSs

- Design schema upfront
- Write or use software to translate your data into that schema … and back
- Friction?

1) beware of speed considerations without having an app to measure
2) or use an ORM which turns out to be a pain in the back for all sorts of reasons
3) Most Data is not inherently relational



not interested in low-concurrency sites, denormalization



not interested in low-concurrency sites, denormalization

CouchDB Documents

CouchDB Documents

- Isolated data records called **Documents**
- No schema (!)
- and semi-structured

data records that make up the app’s data objects

```
{
  "_id": "BCCD12CBB",
  "_rev": "AB764C",

  "type": "person",
  "name": "Darth Vader",
  "age": 63,
  "headware":
    ["Helmet", "Sombrero"],
  "dark_side": true
}
```

```
{
  "_id": "BCCD12CBB",
  "_rev": "AB764C",

  "type": "person",
  "name": "Darth Vader",
  "age": 63,
  "headware":
    ["Helmet", "Sombrero"],
  "dark_side": true
}
```

UUID

```
{
  "_id": "BCCD12CBB",
  "_rev": "AB764C",

  "type": "person",
  "name": "Darth Vader",
  "age": 63,
  "headware":
    ["Helmet", "Sombrero"],
  "dark_side": true
}
```

optimistic locking

```
{
  "_id": "BCCD12CBB",
  "_rev": "AB764C",

  "type": "person",
  "name": "Darth Vader",
  "age": 63,
  "headware":
    ["Helmet", "Sombrero"],
  "dark_side": true
}
```

CouchDB Documents

- Supported by all major languages
- No database abstraction needed

Revisions, Attachments

Working with Documents

```
Create: HTTP POST    /db/BCCD12CBB
  Read: HTTP GET     /db/BCCD12CBB
Update: HTTP PUT     /db/BCCD12CBB
Delete: HTTP DELETE /db/BCCD12CBB
```

"Django may be built *for* the Web, but CouchDB is built *of* the Web."

— Jacob Kaplan-Moss, jacobian.org

“Reading the CouchDB API. Smiling.”

— Tim Bray, on Twitter

```
$ curl -X GET http://server/ \
  database/document
{"_id":"ABC","_rev":"1D4","data
":...}
$
```

Recap

- Versioned Object Store
- Optimistic Locking
- REST API

Revisions, Attachments

Views

of Keys and Values

Views

- Filter, Collate, Aggregate
- Powered by MapReduce

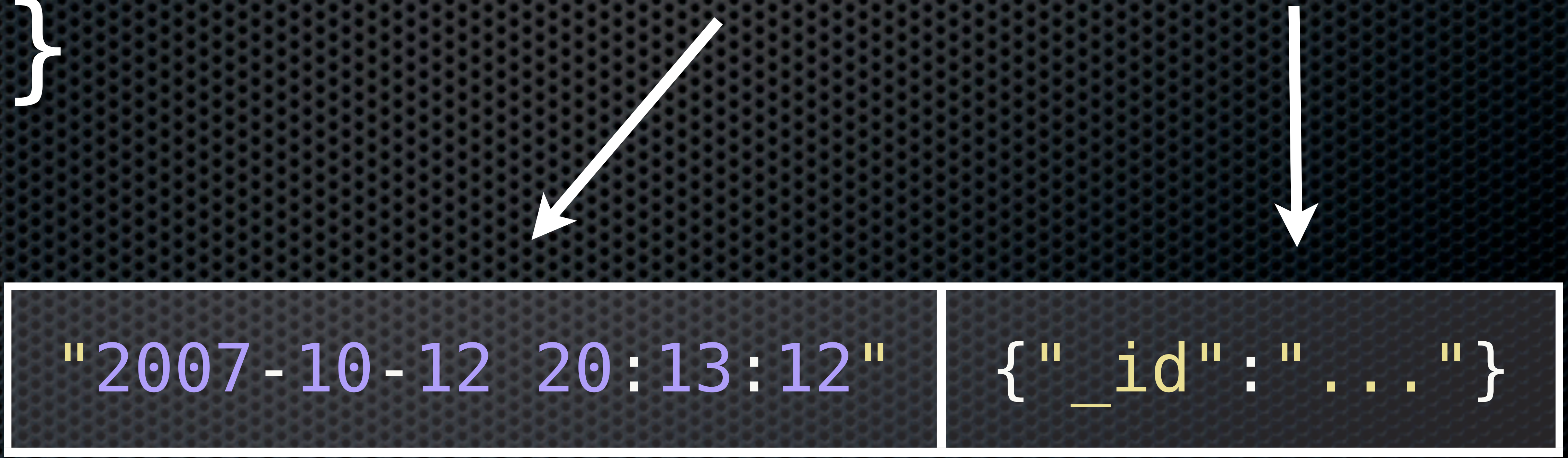
Design documents
functions get executed, you don’t do that

View Examples – Docs by Date

Key	Value
"2007-10-12 20:13:12"	{"_id":"..."}
"2007-12-26 08:37:55"	{"_id":"..."}
"2008-02-03 10:22:34"	{"_id":"..."}
"2008-05-01 14:16:11"	{"_id":"..."}

View Examples – Docs by Date

```
function(doc) {
  emit(doc.date, doc);
}
```



```
"2007-10-12 20:13:12"    {"_id":"..."}
```

Views

- Built incrementally…
- …and on demand
- Reduce optional

map/reduce can be parallelised

Recap

- Versioned Object Store, Optimistic Locking, REST API
- MapReduce Views

Revisions, Attachments

Replication

Replication

Easy Data Synchronization Without Headaches

Replication

- Take your data with you
- CouchDB makes it easy to synchronise machines

rsync-like
Large spectrum of architectures:
 - P2P, Failover, Load Balancing, Backup
Conflicts: auto-detect & resolve, data consistency

Built for the Future

- Written in Erlang – a telco-grade concurrent platform
- Non-locking MVCC and ACID compliant data store

Erlang Processes + messaging
Ericsson AXD 301 - nine nines - 1/30th second per year
Crash resistant

Recap

- Versioned Object Store, Optimistic Locking, REST API, MapReduce Views
- Insane Concurrency Replication & Crash Resistant

Revisions, Attachments

Recap

- Versioned Object Store, Optimistic Locking, REST API, MapReduce Views
- Replication & Crash Resistance

Awesome!

Revisions, Attachments

A Little History

- Damien Katz self funded fulltime development for 2 years
- Now backed by IBM

IBM®

A Little History

- Top Level Apache Project
- Apache 2.0 License

A Little History

- 5th year of development
- Prototype in C++
- 0.8.1: 6666 Lines of Code

Resources

- Twitter: @CouchDB & http://couchdb.org/
- Dress like a Couch: http://shop.couchdb.com
- http://damienkatz.net/ & http://jan.prima.de/
- http://blog.racklabs.com/?p=74
- https://peepcode.com/products/couchdb-with-rails

not covered everything,
other talks + tutorials

Commercial Break

The Book

- O'Reilly
- http://books.couchdb.org/relax
- Apache 2.0 Licensed
- Summer 2009

The Book —Can’t wait?

- Help CouchDB
- Hire me for Consulting, Training & Development
- jan@apache.org

Thank You

Really, thanks.

Got it?

Questions

Bonus Slides

Where is my auto increment

- What is auto_increment?
- Unique identifier
- Sequence denominator

Where is my auto_increment?

- Documents have `_id`s
- Sequences in distributed applications are not
- Timestamps get you a long way, though.

Relation(ship)s

- JOINs please!
- What for?
- Get data that “belongs together”

Relation(ship)s

- One big fat doc?
- Pros: Easy – Cons: Bad with concurrent updates
- Use for: Low volume updates e.g. user-supplied tags

Relation(ship)s

- Master Doc – Slave Doc
- Pros: A little complex – Cons: Fast, good with concurrent updates, tree operations
- Use for: Everything else

Relation(ship)s

```
function(doc) {
  if(doc.ismaster) {
    emit([doc._id, doc.date], doc);
  } else {
    emit([doc.master_id, doc.date], doc);
  }
}
```

Relation(ship)s

...	...
["BAAC67", "2008-09-21"]	{"is_parent",true}
["BAAC67", "2008-09-22"]	{"...","..."}
["BAAC67", "2008-09-23"]	{"...","..."}
["BAAC67", "2008-09-24"]	{"...","..."}
...	...

Transactions!

- Run multiple operations at once
- They all succeed or none gets applied

Transactions

```
POST
{
  "docs": [
    {"_id": "0", "int": 0, "str": "0"},
    {"_id": "1", "int": 1, "str": "1"},
    {"_id": "2", "int": 2, "str": "2"}
  ]
}
```

Transactions!

- Caveats:
- Statement transaction
- No data transaction
- No multi-node transactions

Multi-Node Transactions!

- Why? – Data redundancy
- Use an HTTP proxy
- Nice to build on standard protocols
- Caveat: 2-phase-commit in disguise

MapReduce

View Examples – Docs by Date

Map

Key	Value
`[2007, 10, 12, 20, 13, 12]`	3465
`[2007, 12, 26,  8, 37, 55]`	4200
`[2008,  2,  3, 10, 22, 34]`	3782
`[2008,  5,  1, 14, 16, 11]`	5984

View Examples – Docs by Date

Reduce

Key	**Value**
`null`	17431

View Examples – Docs by Date

Reduce with group_level=1

Key	Value
[2007]	7665
[2008]	9766

View Examples – Docs by Date

Map

Key	Value
`[2007, 10, 12, 20, 13, 12]`	3465
`[2007, 12, 26,  8, 37, 55]`	4200
`[2008,  2,  3, 10, 22, 34]`	3782
`[2008,  5,  1, 14, 16, 11]`	5984

Views - Map Tags

Keys	Values
family	1
friends	1
friends	1
work	1
work	1
youtube	1
...	...

Views - Reduce Tag Count

Keys	Values
family	1
friends	1
friends	1
work	1
work	1
youtube	1
...	...

$\Rightarrow$

Keys	Values
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

Keys	Values
family	1
friends	1
friends	1
work	1
work	1
youtube	1
...	...

$\Rightarrow$

Keys	Values
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;
}
```

Incremental, On-demand
reduce optional

Hot backup?

- POSIX compliant

Hot backup?

-
  ```
  $ cp -r /var/lib/couchdb/* \
    /mnt/backup
  ```

Number Bragging

- Silly read-only benchmark with memory saturation
- 2,500 req/s sustained on a 2Ghz dual core Athlon

Number Bragging

- Silly read-only benchmark with memory saturation
- 2,500 req/s sustained on a 2Ghz dual core Athlon
- **Using 9.8 MB RAM**

Resources

- Twitter: @CouchDB & http://couchdb.org/
- Dress like a Couch: http://shop.couchdb.com
- http://damienkatz.net/ & http://jan.prima.de/
- http://blog.racklabs.com/?p=74
- https://peepcode.com/products/couchdb-with-rails

not covered everything,
other talks + tutorials