FluidDB

Design ➔ Architecture ➔ Tradeoffs

Terry Jones

terry@jon.es
@terrycojones
Or...

How someone who knows nothing about databases started building a database
Early days

- FluidDB is not yet released
- It’s an experiment
- Much remains to be done
- Alpha release *real soon now* (end of June?)
Fluidinfo

- Founded in the UK in March 2006
- Two full-time employees
- Development in Barcelona
- Friends, family, & Esther Dyson funded
Motivations

- How humans work with information: Concepts
- User interface and API difficulties / restrictions
- Personalization
- Make the world more writable
Concepts

Are not owned
No formal structure
No permission is needed
Partial or full disorganization
No predefined set of qualities
No special central piece of content
Easy to reorganize or multiply organize
Concepts

Are not owned
No formal structure
No permission is needed
Partial or full disorganization
No predefined set of qualities
No special central piece of content
Easy to reorganize or multiply organize

To engineer this kind of flexibility,
we must rethink control
Uls and APIs

- Where did my information go?
- Can I extract, re-use, add, delete?
- Is there an API?
- What am I allowed to do?
- Can I search? On what?
- Special pleading
- Wouldn't it be cool if...
Personalization

In our hands vs on our behalf

- Add anything to anything, and search on it
- Protect, share, delete as you wish
- Organize things as you please
- Freedom of choice in applications
Read ➤ Search ➤ Write?

- The web is readable (c. 1990)
- And searchable (c. 1995)
- But not generally writable
The web is readable (c. 1990)
And searchable (c. 1995)
But not generally writable

Plus, search is not very interactive:
- Dull: refine query or ask for more results
- Can’t change results
- Can’t organize
Why don’t our architectures let us work with information more flexibly?
What would it take to build something that did?
How can we address all these problems at once?
At PARC we had a slogan: “Point of view is worth 80 IQ points.” It was based on a few things from the past like how smart you had to be in Roman times to multiply two numbers together; only geniuses did it.

We haven't gotten any smarter, we've just changed our representation system. We think better generally by inventing better representations; that's something that we as computer scientists recognize as one of the main things that we try to do.
At PARC we had a slogan: “Point of view is worth 80 IQ points.” It was based on a few things from the past like how smart you had to be in Roman times to multiply two numbers together; only geniuses did it.

We haven't gotten any smarter, we've just changed our representation system. We think better generally by inventing better representations; that's something that we as computer scientists recognize as one of the main things that we try to do.
VIII
XVII
XLIV
LXXX
XCVI
CCLV
8 queens problem

Use a poor representation with $2^{64}$ (281,474,976,710,656) states:

Look for a smart algorithm!

Or...

Use a good representation with 8! (40,320) states and exhaustive search as an algorithm!
Objects

- **digg.com/date**: May 21, 2009
- **meg/web/rating**: 6
- **tim/seen**: true
- **about**: http://digg.com/news.html
- **mike/opinion**: “half-baked nonsense”
What else?

- Objects have no owner, though attributes do
- No metadata/data duality
- Unlimited aggregation & combination of information
- No a priori organization
- Search, search, search!
- Dynamic data structures
FluidDB

- Objects composed of attributes with values
- Attributes in namespaces: joe/rating, amazon.com/price
- Simple query language
- Simple API: HTTP, JSON, REST etc.
- Users, attributes, namespaces each have an object
- Applications are users too
FluidDB

- Single-instance hosted database (like SimpleDB)
- Enables *sharing* between applications, people
- Distributed storage and query processing
Design goals

- Simple data model
- Simple permissions model
- Simple, easily parallelizable, query language
- Horizontally scalable
- Fast for common tasks
- Implementable!
For each action on a namespace or attribute:

- There’s a policy: ‘open’ or ‘closed’
- And a (possibly empty) list of exceptions
# Permissions

<table>
<thead>
<tr>
<th>attribute or namespace</th>
<th>action</th>
<th>policy</th>
<th>exceptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim/seen</td>
<td>read</td>
<td>closed</td>
<td>tim, meg</td>
</tr>
<tr>
<td>mike/opinion</td>
<td>update</td>
<td>open</td>
<td></td>
</tr>
<tr>
<td>mike/</td>
<td>create</td>
<td>closed</td>
<td>mike</td>
</tr>
<tr>
<td>meg/rating</td>
<td>see</td>
<td>open</td>
<td></td>
</tr>
<tr>
<td>meg/rating</td>
<td>read</td>
<td>closed</td>
<td>meg</td>
</tr>
</tbody>
</table>
Query language

Numeric: attribute value ( =, <, etc.)
Textual: attribute text match
Presence: has attribute
Grouping/logic: (...), and, or, not
Query language

Numeric: attribute value (=, <, etc.)
Textual: attribute text match
Presence: has attribute
Grouping/logic: (...), and, or, not

Designed to bring back object ids
Demo
Data buffet

- username seen, lastvisited, rating, goingtoread, comment
- username me, FBfriend, linkedInContact, met, family
- username/myusername name, password
- flickr.com longitude, latitude, owner, date, camera/make
- username/flickr title, description
- VCspotter fredwilson, bradburnham
- nasdaq.com name, symbol, type, outstanding, value, price
- username/stocks shares, date
- google.com pagerank
- tracks album, artist, name, year
- username/music count, favorite, lastPlay, stars, bestOf2007
Data buffet 2

- email messageId, fromId, told
- username/email from, to, subject, date
- alexa.com rank
- digg.com title, description, date, diggs
- mahalo.com appeared, category
- readwriteweb.com appeared
- reddit.com date, score
- techcrunch.com appeared, URI
- attribute description, name, path
- namespace description, name, path
Example queries

- terry/rating > 5 and has reddit.com/score
- has goingtoread and seen > "January 1, 2008"
- has FBfriend and has linkedInContact
- has james/FBfriend and not has anne/FBfriend
- alexa.com/rank < 50 and fred/comment ~ cool
- has reddit.com/score and not has digg.com/diggs
- has readwriteweb.com/appeared and not has techcrunch.com/appeared
More queries

- terry/seen > "July 1, 2007"

- has russell/myusername/name and not has terry/myusername/name

- flickr.com/latitude > 52.15 and flickr.com/latitude < 52.35 and flickr.com/camera/make ~ Sony and has sally/seen

- amazon.com/stars > 3 and amazon.com/price < 20 and amazon.com/title ~ chess and peter/bookrating > 3 sort by amazon.com/publication-date
Architecture

- Permissions
- Software
- Communications
- Functional
- Storage
- Query processing
- Per box
Twisted

- Asynchronous Python libraries
- We’re using Python because of Twisted
- Event-driven, using deferreds with callbacks
- Steep learning curve, documentation is patchy
AMQP

- Standardized, high performance messaging
- Backed by JP Morgan, Novell, MS, Redhat
- Fully asynchronous, up to 600K messages/sec
- Exchanges, queues, bindings
- We released txAMQP
Thrift

- Released by Facebook
- Serialization of structures
- Definition of services
- RPC
- We released txThrift
Communications

AMQP
Functional

- apps
  - http
    - facade
      - coord
        - control
      - sets
  - objects
    - namespace
      - attr
        - text
    - objects
Functional

http

facade

sets

coord

coord

objects

namespace

attr

text
Functional

http

facadesets

objects

objectsnamespace

coord

amqp

attr
text
Functional

http
http
xmpp
xmpp
facade
facade
coord
coord
sets
sets
amqp
amqp
objects
namespace
attr
text
Functional

- http
- facade
- sets
- objects
- namespace
- attr
- text
- xmpp
- coord
- amqp
- other
Functional
Functional
Functional

- load
- load
- load
- load
- http
- http
- facade
- facade
- sets
- sets
- objects
- objects
- namespace
- namespace
- attr
- attr
- text
- text
- memcache
- memcache
- xmpp
- xmpp
- coord
- coord
- amqp
- amqp
- db
- db
- db
- db
- db
- db
- kv
Attribute Storage

meg/rating

<table>
<thead>
<tr>
<th>object id</th>
<th>user id</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234567</td>
<td>667</td>
<td>26</td>
</tr>
<tr>
<td>6527527</td>
<td>667</td>
<td>188</td>
</tr>
<tr>
<td>2876281</td>
<td>17</td>
<td>207</td>
</tr>
<tr>
<td>7628876</td>
<td>667</td>
<td>1225</td>
</tr>
</tbody>
</table>

tim/books/opinion

<table>
<thead>
<tr>
<th>object id</th>
<th>user id</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>526141</td>
<td>362</td>
<td>nice</td>
</tr>
<tr>
<td>726483</td>
<td>362</td>
<td>fun</td>
</tr>
<tr>
<td>635378</td>
<td>362</td>
<td>boring</td>
</tr>
<tr>
<td>477582</td>
<td>362</td>
<td>sexy</td>
</tr>
<tr>
<td>362782</td>
<td>362</td>
<td>long</td>
</tr>
</tbody>
</table>

- PostgreSQL
- Tall tables
- Independent (column store)
- Backed by key/value store (Amazon S3, for now)
Query processing

\[
\text{digg/date} > \text{“Monday”} \quad \text{or} \quad \text{meg/rating} > 5 \quad \text{has tim/seen}
\]
Query processing

digg/date > “Monday”

meg/rating > 5

has tim/seen

attr

set ops
Attribute affinity

digg/date > “Monday”

meg/rating > 5

has tim/seen

attr

set ops
A controller service, launched on boot

The controller launches new services (processes)

All services talk AMQP as well as pure Thrift

A coordinator brings up new boxes & services

We use Amazon EC2, for now