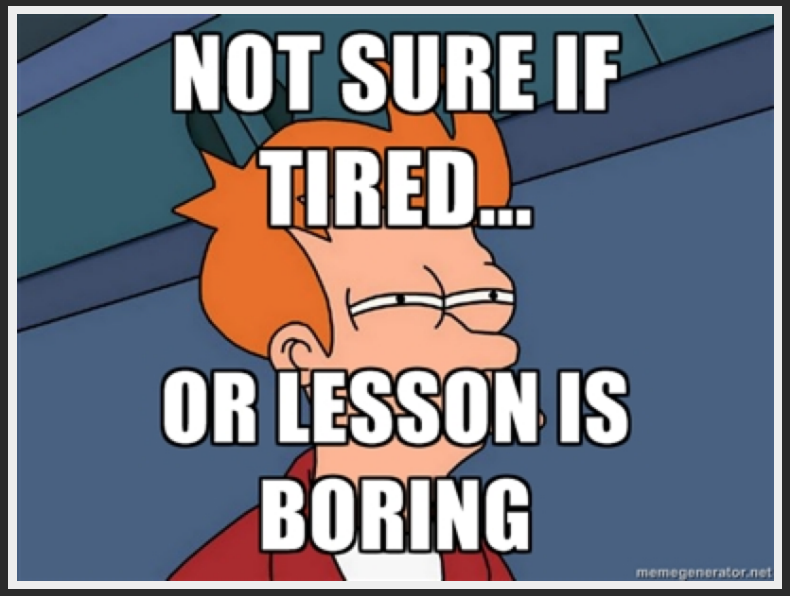
Using MongoDB as a casually agile data store

NoSQL Search Roadshow Copenhagen 2013


NOT SURE IF
TIRED...
OR LESSON IS
BORING
memegenerator.net

?

- What is MongoDB
- Features at-a-glance
- TradingHub
- Observations
- Wishlist

Target audience

Programmers who...

- ...need to save data
- ...like to DELIVER
- ...haven't had that much experience with MongoDB
- ...are experiencing relational agony

Mogens Heller Grabe

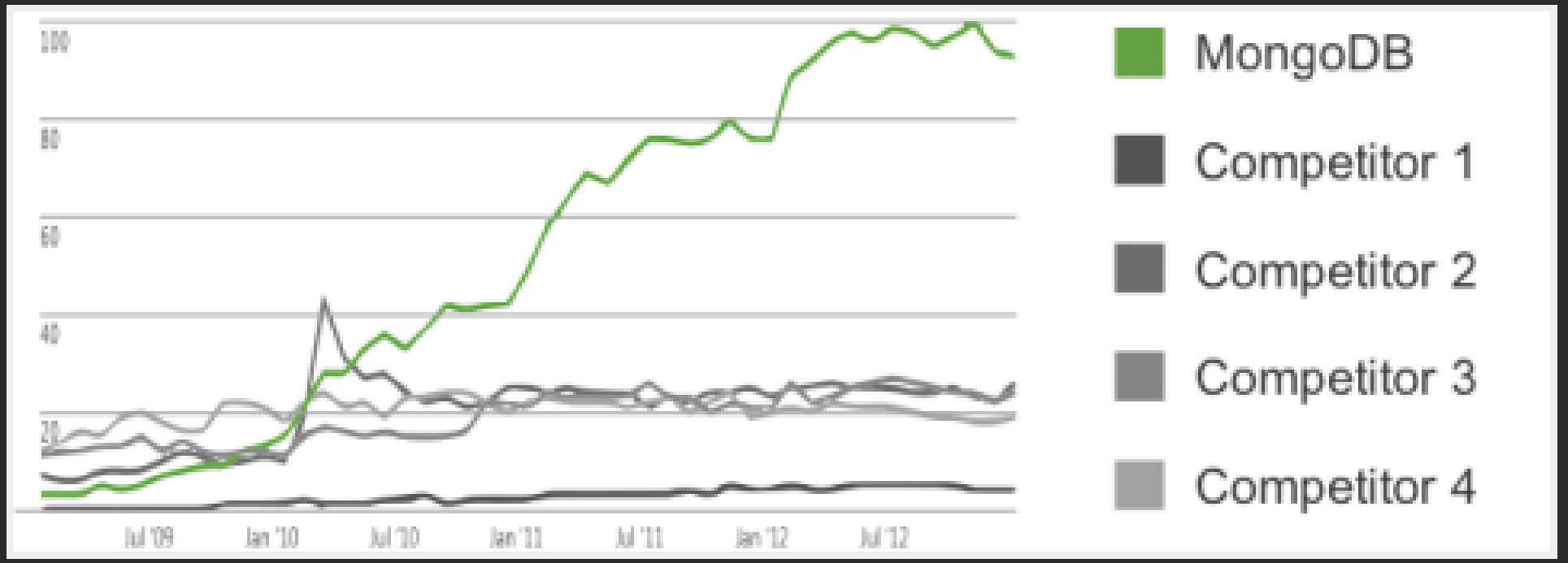
d60

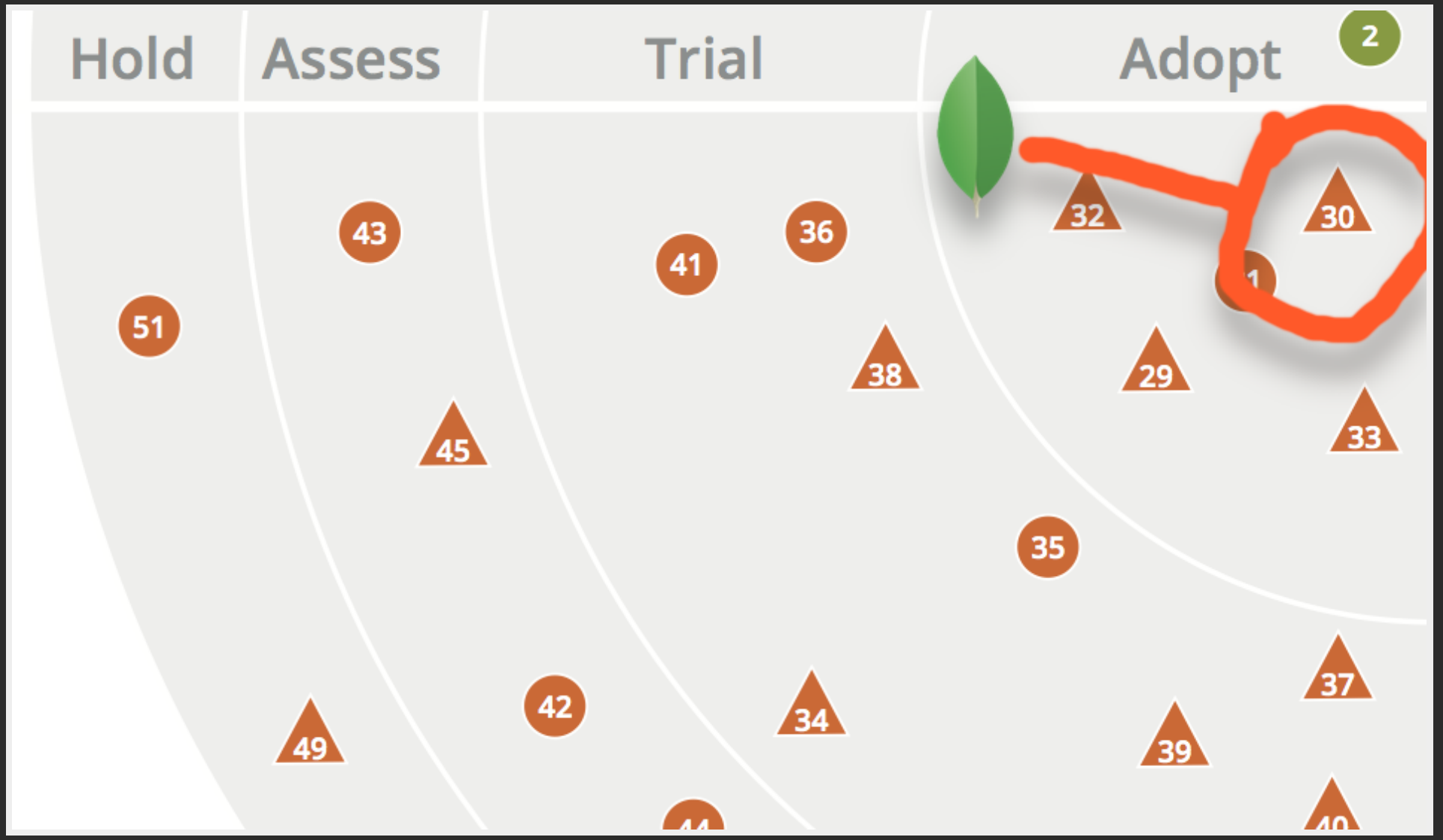
mhg@d60.dk

http://mookid.dk/oncode

@mookid8000

What is MongoDB?


100
80
60
40
20
Jul '09
Jan '10
Jul '10
Jan '11
Jul '11
Jan '12
Jul '12
MongoDB
Competitor 1
Competitor 2
Competitor 3
Competitor 4


Hold
Assess
Trial
Adopt
2
32
30
1
43
36
41
51
38
29
45
33
35
37
42
34
49
39
40

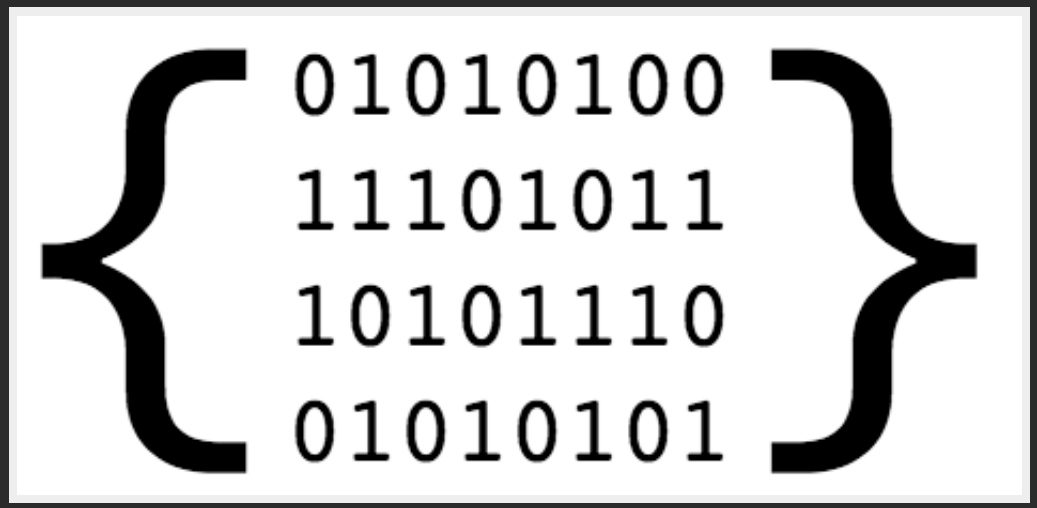
Features at-a-glance

Documents

```
{
    "_id": "whatever",
    "what": {
        contains: "embedded document"
    },
    "more_stuff": ["array", "of", {"what": "ever", "you": "want"}],
    "counter": 23,
    "description": "bla bla bla bla"
}
```

BSON

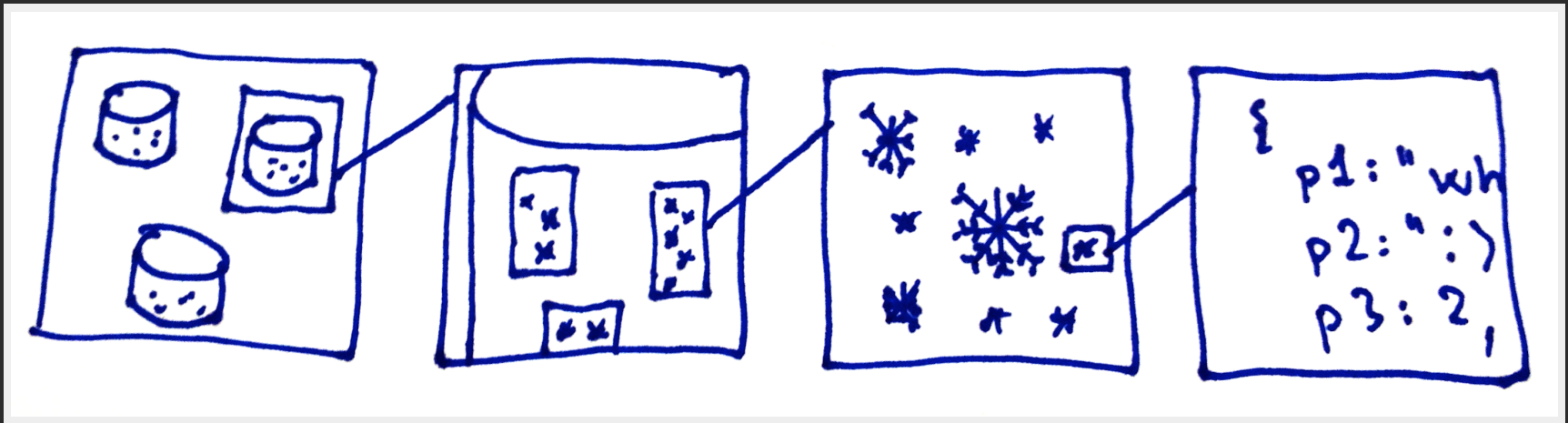
BSON



Adds

- byte • int32 • int64 • double • string • binary
- ObjectId • Date • bool • ...

Data organization (logical)



Database -> Collection -> Doc

Database -> Table -> Row

Data organization (physical)

- Memory-mapped files
- Transaction log

Queries

```
{ "_id": ObjectId("someId") }
```

```
{ "counter": { "$gt": 20 } }
```

```
{
    "what.contains": /^embedded.*/,
    "more_stuff": {
        "$elemMatch": {
            "what": "ever",
            "you": "want"
        }
    }
}
```

```
{
        "_id": "whatever",
        "what": {
                contains: "embedded document"
        },
        "more_stuff": ["array", "of", {"what": "ever", "you": "want"}],
        "counter": 23,
        "description": "bla bla bla bla"
}
```

Updates

First argument: Query that acts as criteria, and then:

```
{ "$inc": { "counter": 1 } }
```

```
{ "$set": { "what": ["replace", "doc", "with", "array"] } }
```

```
{
        "$addToSet": { "more_stuff": "possibly a tag" },
        "$inc": { "counter": 1 },
        "$set": { "location": { "lng": 55.84, "lat": 9.80 }} }
}
```

which gets processed atomically per document

Aggregation

```
{
        name: "Mogens Heller Grabe",
        likes: [
                "Finca Vista Hermosa",
                "Colombia Supremo",
                "El Salvador Honey"
        ]
}
```

Aggregation

```
[
        {$unwind: "$likes"},
        {$group: {_id: "$likes", count: {$sum: 1}}},
        {$sort: {count: -1}},
        {$limit: 5}
]
```

Indexes

```
db.some_collection.ensureIndex({"counter": 1})
```

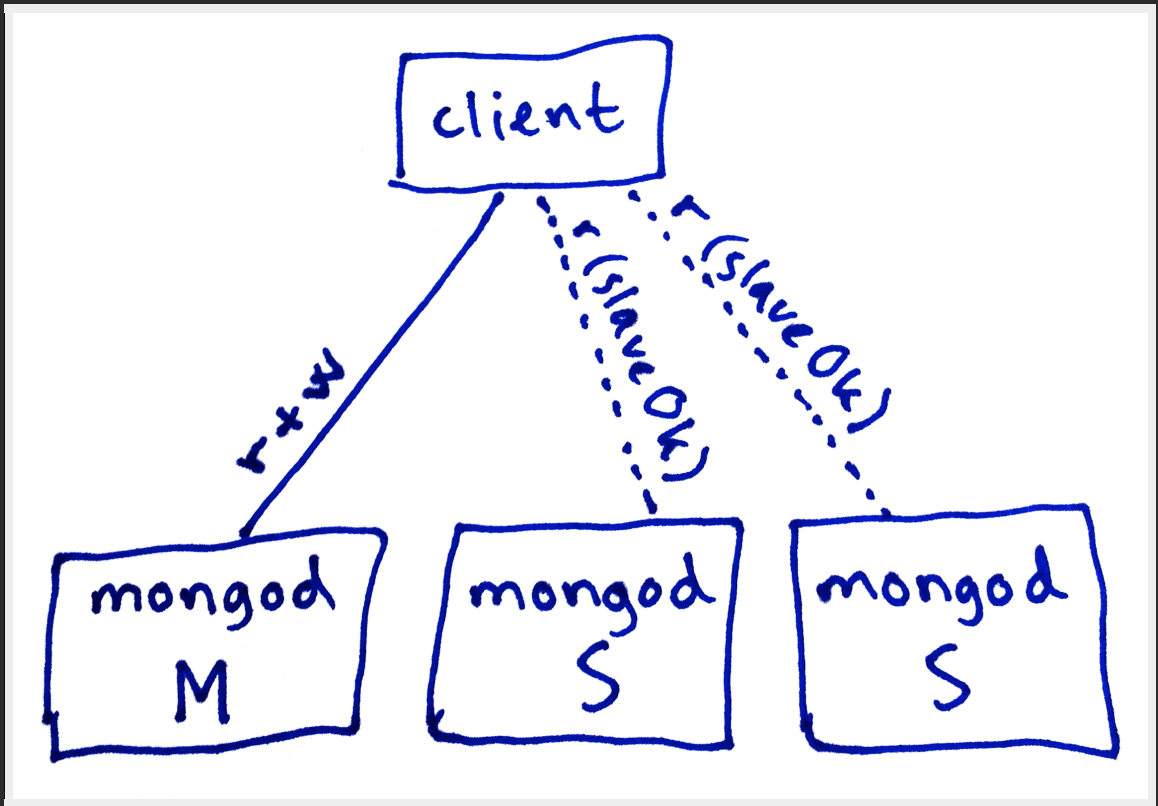
```
db.some_collection.ensureIndex({
        "more_stuff": 1,
        "what.contains": 1
})
```

```
db.some_collection.ensureIndex({"location": "2d"}, {"background": true})
```

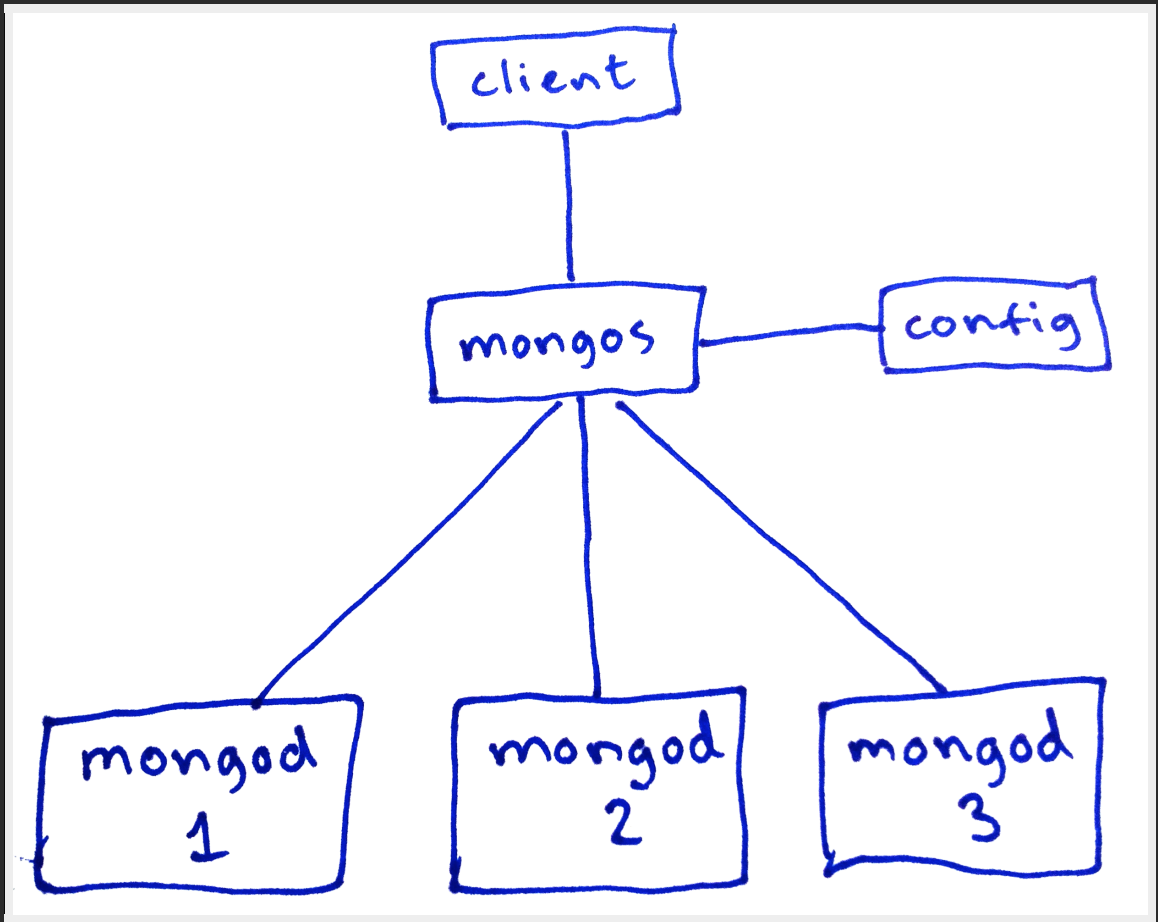
```
db.some_collection.ensureIndex({"description": "text"})
```

Deployment

Replica set


client
r+w
r (slaveOk)
r (slaveOk)
r (slaveOk)
mongod
M
mongod
S
mongod
S

Sharding



Durability

To have it:

< 1.8: Required replication

1.8: Could enable journaling

>= 2.0: Journaling enabled by default

Durability

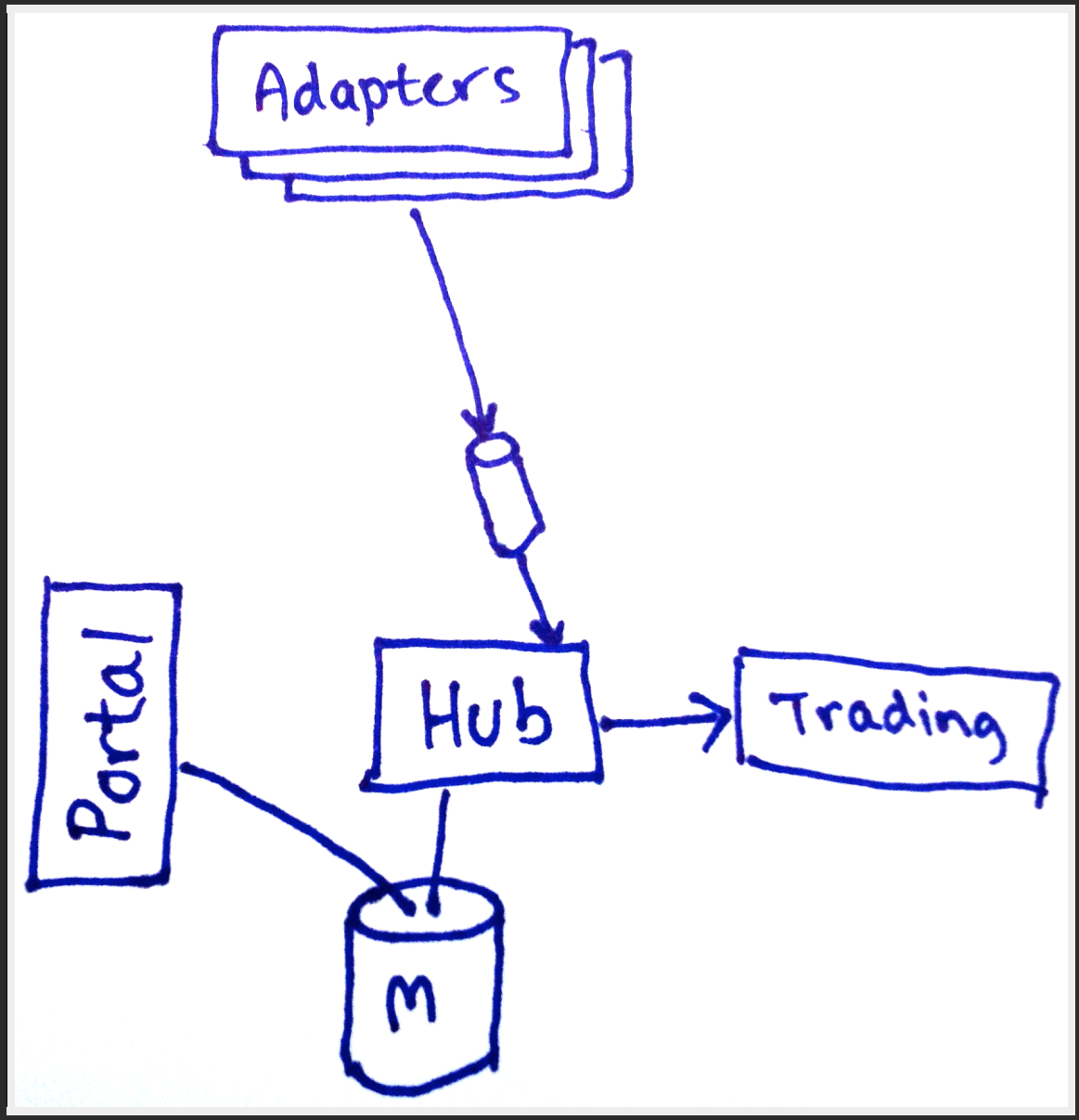
Can be had in *boring mode*

a.k.a. "easy mode"

TradingHub

Pull trades from various sources

Overview


Adapters
Portal
Hub
Trading
M

Model

- Trade integration flow
- Enrichment information
- Archive and logs

Trade integration flow

```
{
        _id: BinData(3, "base64stuffinhere"),
        _rev: 14,
        TradeData: {
                /*
                all the stuff about traded volumes,
                times, prices etc.
                 */
        },
        Meta: {
                SuccessfullyDelivered: true,
                FailedDeliveryAttempts: 2,
                Log: [
                        {
                                Time: ISODate("2013-05-17T07:04:43.139Z"),
                                Message: "bla bla bla"
                        },
                        ...
                ]
        }
```

Trade integration flow update

```
// ...work on updating the trade flow up here

// now, update it in DB
var idToMatch = updatedTradeFlow.Id;
var revToMatch = updatedTradeFlow.Revision;

updatedTradeFlow.Revision++;

var idMatch = Query<TradeFlow>.EQ(f => f.Id, idToMatch);
var revMatch = Query<TradeFlow>.EQ(f => f.Revision, revToMatch);

var result = trades.Update(Query.And(idMatch, revMatch),
                           Update.Replace(updatedTradeFlow));

if (result.DocumentsAffected == 0)
{
    throw new ConcurrencyException("w00t!");
}
```

Enrichment information

Basically just a bunch of locally cached information from the main trading system ...*with the right indexes!*

```
> db.gridMappings.find({SourceId: "SomeTradeSource"}).explain()
{
        "cursor" : "BtreeCursor SourceId_1",
        "isMultiKey" : false,
        "n" : 1,
        "nscannedObjects" : 1,
        "nscanned" : 1,
        "millis": 0, // <- like that     (-▪_▪)
(...)
```

Archiving/logging

Whenever XML is received from one of the adapters:

```
> db.receivedXml.insert({
        /* date, sender, and all the raw XML */
})
```

Whenever an admin aborts an integration flow:

```
> db.deletedTrades.insert({
    DeletedData:  /* the entire trade flow at the time of deletion */,
    DeletionInfo: {
        DeletedBy: "Mogens",
        Date: ISODate("someDate"),
        Comment: "Thought it would be funny"
    }
})
```

...etc!

Easy Peasy  Squeezy!

Stats

Docs: ~ 10 * 10^6

Database size: ~ 3 GB

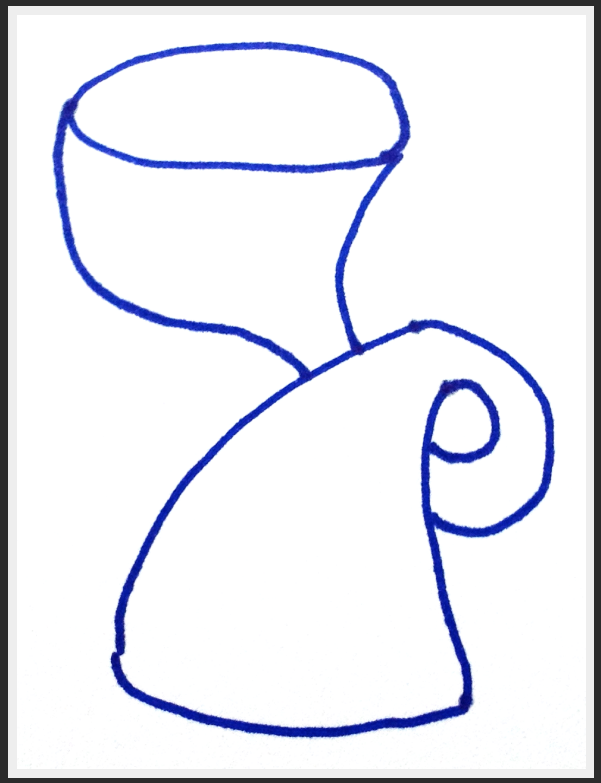
Zipped: ~ 25 MB :)

Prediction: You're not impressed.

That's also not what I wanted to do ;)

Observations

Schemaless FTW!1



Transactions

Transactions not supported across multiple documents - therefore:

1. Load integration flow
2. Mutate in-mem
   Make 0..* *idempotent* database ops
3. Update integration flow

Cannot assume any kind of isolation!

Binary data type

Remember that `BinData(...)` thing?

yeah, avoid it if you can...

e.g. .NET `Guid` gets serialized as `BinData`

Modeling

Relational data modeling: Capture the data, make sense of it later

Document-oriented data modeling: Capture the data in a form that makes sense

Modeling

Click tracking, relational style:

Time	ProductId	UserId
2013-06-13 14:15	1234567	123
2013-06-13 14:15	7634512	123
2013-06-13 14:16	1234567	123
2013-06-13 14:16	1234567	123

Modeling

Click tracking, document style:

```
{
        "_id": {
                "date": "2013-06-13",
                "product_id": 1234567
        },
        "clicks": 9,
        "hours": {
                "12": {
                        "clicks": 5,
                        "30": { "clicks": 3 },
                        "31": { "clicks": 2 },
                        "users": ["111299"]
                },
                "13": {
                        "clicks": 4,
                        "55": { "clicks": 4 },
                        "users": ["111299"]
                }
        }
}
```

Modeling

Click tracking, document style:

```
db.clicks.update(
        _id: {
                "date": "2013-06-13",
                "product_id": 1234567
        },
        {
                $inc: {
                        "clicks": 1,
                        "hours.16.clicks": 1,
                        "hours.16.12.clicks": 1
                },
                $addToSet: {"hours.16.users": "111299"}
        }
, true) //< upsert!
```

Modeling

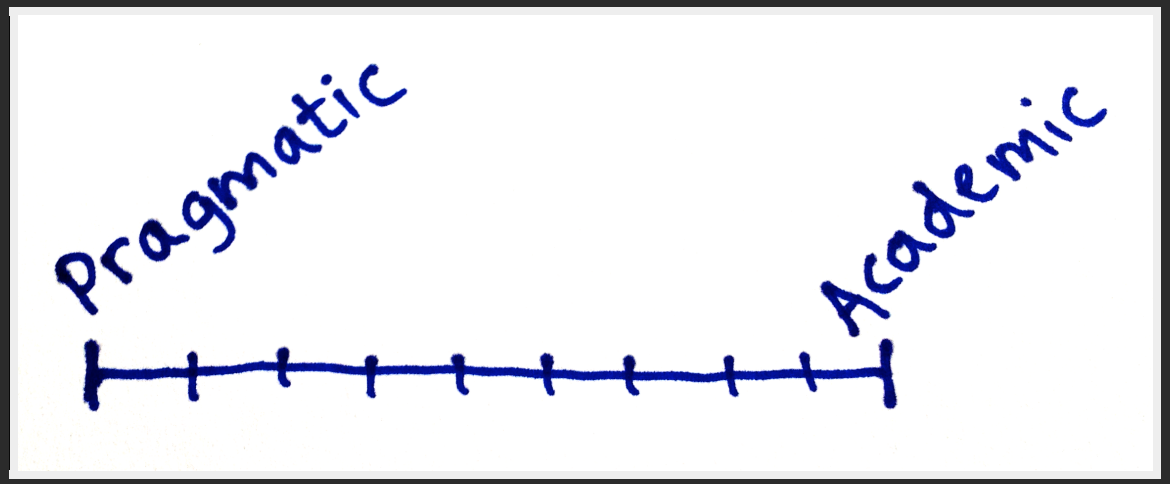
YMMV

+ the aggregation framework made it easier to make sense out of many small pieces of data

Write concern?

Two kinds of durability:

- Replication
- Transaction log ("journaling")


Pragmatic
Academic

Wishlist

~~Full-text search~~

It's so cool that it got that!

Now I can Google my archives of received XML

Automatic optimistic concurrency

Multi-document transactions

FoundationDB and **HyperDex** can do that...

=> it's not impossible!

(proof by contradiction)

"Joins"

RethinkDB and **RavenDB** can do that...

=> it's not impossible!

(proof by contradiction)

Materialized aggregation

Summary

I'd definitely use MongoDB again - mostly because of

- low ceremony
- high flexibility

Thank you for listening!

...and a big thank you to **Hakim** for creating the immensely awesome **reveal.js**

...and thanks to Tony Hisgett for that cool **nice bison picture**

Mogens Heller Grabe

d60

mhg@d60.dk

@mookid8000

http://mookid.dk/oncode