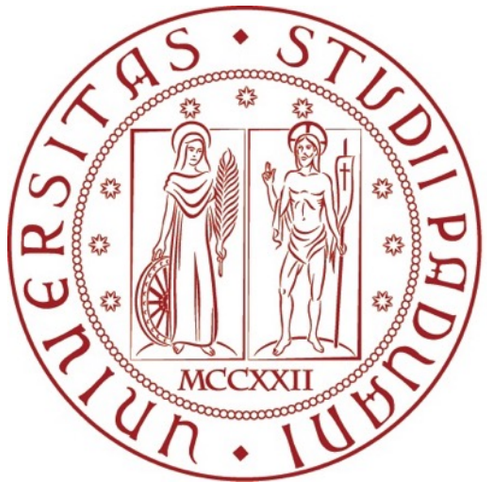


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Massimo Brignoli

massimo@mongodb.com

@massimobrignoli

Stefano Dindo

s.dindo@zero12.it

@stefanodindo

13 Ottobre 2013

Cos'è MongoDB ?

Database NOSQL **document** oriented

Si propone come soluzione al problema della quantità di informazioni: è un sistema che si adatta meglio alla crescita consistente delle dimensioni degli archivi, elaborandoli in poco tempo.

Benefici del modello a documenti

Agilità & Flessibilità
- Evoluzione semplice del modello di dati
- Adattamento e cambi rapidi sulla struttura dati

Rappresentazione dati intuitiva e naturale
- Sviluppatori sono più produttivi
- Le applicazioni risultano più semplici da gestire

Riduzione di operazioni di join e disk seek
- La programmazione è più semplice
- Maggiori performance

Introduzione … dai concetti base alle prime query

Terminologia

SQL - Like	Corrispettivo in MongoDB
Database	Database
Tabella	Collection
Riga	Document
Colonna	Field

Cos'è un documento per MongoDB?

Un documento è una struttura dati in stile JSON formata da un numero variabile di coppie campo - valore

```
{
   “nome”: “Giorgio”,
   “cognome”: “Rossi”,
   “email”: “giorgio.rossi@gmail.com”,
   “cell”: 3281432896,
   “sport”: [“nuoto”, “calcio”]
}
```

MongoDB salva i documenti su disco in formato BSON (http://bsonspec.org).
La dimensione massima di un documento **BSON è 16MB**.

MongoDB: Modello a Documenti

PERSON

Pers_ID	Surname	First_Name	City
0	Miller	Paul	London
1	Ortega	Alvaro	Valencia
2	Huber	Urs	Zurich
3	Blanc	Gaston	Paris
4	Bertolini	Fabrizio	Rome

CAR

Car_ID	Model	Year	Value	Pers_ID
101	Bently	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2


```
{
   _id: ‘Objectid(“4b2b9…”)’,
   first_name: ‘Paul’,
   surname: ‘Miller’,
   city: ‘London’,
   location: [45.123,47.232],
   cars: [
      { model: ‘Bentley’,
        year: 1973,
        value: 100000, … },
      { model: ‘Rolls Royce’,
        year: 1965,
        value: 330000, … }
   ]
}
```

Documento

Esempio

```
{
  _id: ‘Objectid(“4b2b9…”)’,
  first_name: ‘Paul’,
  surname: ‘Miller’,
  city: ‘London’,
  location: [45.123,47.232],
  cars: [
    { model: ‘Bentley’,
      year: 1973,
      value: 100000, … },
    { model: ‘Rolls Royce’,
      year: 1965,
      value: 330000, … }
  ]
}
```

Tipi di dati

Null `{ x: null }`

Boolean `{ x: true }`

Number `{ x: 3.14 } { x: 3 }`

String `{ x: “zero12” }`

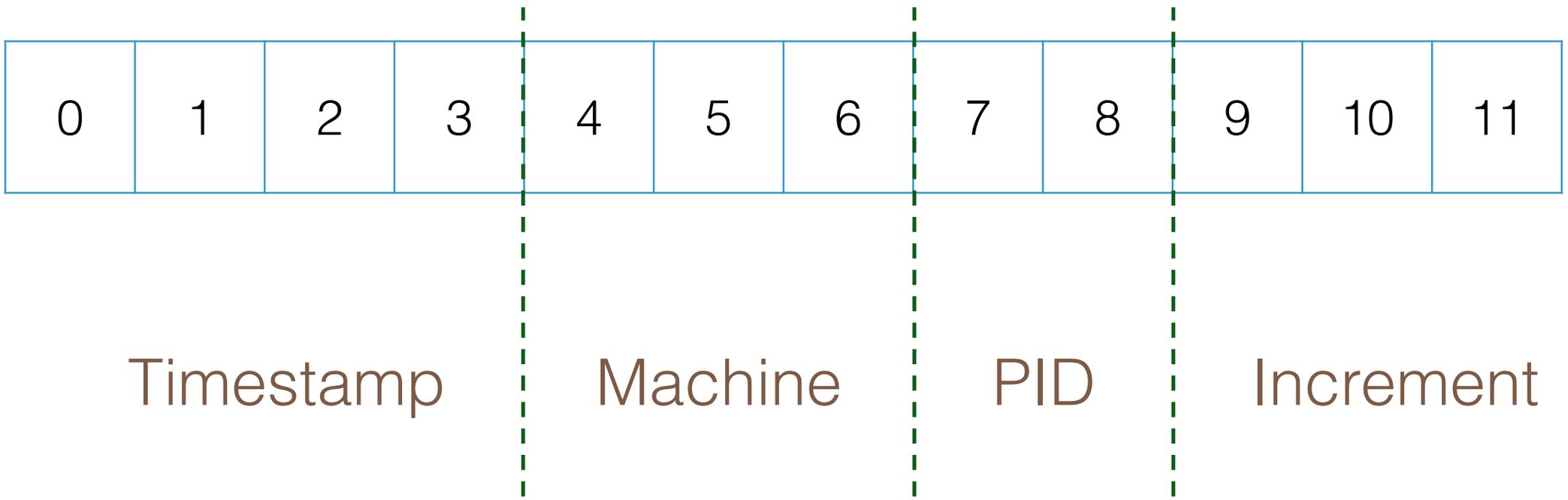
Date `{ x: new Date() }`

Array `{ x: [“a”,”b”, “c”] }`

Embedded documents

`{ x: {y: “a” } }`

Object id



Terminologia

Mongod:

E' il processo server che esegue il database e che rende disponibile tutti i servizi per la creazione di database, collections, l'inserimento di documenti e le relative funzioni di interrogazione

Mongos:

E' il servizio di routing per processare le query provenienti dallo strato applicativo e che determina la locazione dei dati richiesti in una configurazione Sharding.

Mongo:

Rappresenta l'eseguibile per avviare la shell Javascript per interagire con database, collezioni e documenti che possono essere interrogati e manipolati da riga di comando.

MongoDB shell

MongoDB dispone nativamente di una shell javascript per l'amministrazione del database

mongo 172.31.21.36:27017/firstDB

```
ubuntu@ip-172-31-21-36:~$ mongo 172.31.21.36:27017/firstDB
MongoDB shell version: 2.6.3
connecting to: 172.31.21.36:27017/firstDB
>
```

Principali Comandi:

use <nome db>

show collections

db.<nomeCollection>.insert({…})

db.<nomeCollection>.findOne()

db.<nomeCollection>.update(<criterio>, {…})

db.<nomeCollection>.remove(<criterio>, {…})

Query

```
db.<nomeCollection>.insert(
            <Documenti o array di Documenti>,
            {
                writeConcern: <document>,
                ordered:<boolean>
            }
)
```

```
db.testcollection.insert({item: “card”, qty: 15})
```

```
output: WriteResult({“Inserted”: 1})
```

find({…})

db.<nomeCollection>.find({<criterio>},{return key})

productsold

```
{
  custom_id: “A123”,
  amount: 500,
  status: “A”
}
```

```
{
  custom_id: “A123”,
  amount: 250,
  status: “A”
}
```

```
{
  custom_id: “B212”,
  amount: 200,
  status: “A”
}
```

```
{
  custom_id: “A123”,
  amount: 300,
  status: “D”
}
```

db.productsold.find({status: “D”})

```
{
  custom_id: “A123”,
  amount: 300,
  status: “D”
}
```

find({…})

db.<nomeCollection>.find({<criterio>},{return key})

productsold

```
{
  custom_id: “A123”,
  amount: 500,
  status: “A”
}
```

```
{
  custom_id: “A123”,
  amount: 250,
  status: “A”
}
```

```
{
  custom_id: “B212”,
  amount: 200,
  status: “A”
}
```

```
{
  custom_id: “A123”,
  amount: 300,
  status: “D”
}
```

db.productsold.find({status: “D”},
{customer_id: 1, amount: 1 })

```
{
  custom_id: “A123”,
  amount: 300
}
```

Alcuni operatori per query condizionali

$lt: < $lte: <= $gt: > $gte: >= $or

productsold

```
{
  custom_id: “A123”,
  amount: 500,
  status: “A”
}
```

```
{
  custom_id: “A123”,
  amount: 250,
  status: “A”
}
```

```
{
  custom_id: “B212”,
  amount: 200,
  status: “A”
}
```

```
{
  custom_id: “A123”,
  amount: 300,
  status: “D”
}
```

db.productsold.find(
{amount: {“$gte”: 200,“$lte”: 250 }})

```
{
  custom_id: “A123”,
  amount: 250,
  status: “A”
}
```

```
{
  custom_id: “B212”,
  amount: 200,
  status: “A”
}
```

Altri operatori per query

Es. operatori logici

$and: {$and: [{<espressione1},{<espressione1}, …]}

$or: {$or: [{<espressione1},{<espressione1}, …]}

Es. Element

$exists: {field: {$exists: <boolean>}}

Es. Array

$all: {field: {$all: [<value>, <value>]}}

$elemMatch: {field: {$elemMatch: [<q1>, <q2>, …]}}

Query su embedded document

```
{
  _id: ‘Objectid(“c75r4bb6…”)’,
  name: {
      first: ‘Stefano’
      last: ‘Dindo’},
  age: 30
}
```

db.user.find({name.first: “Stefano”, name.last: “Dindo”})

```
db.<nomeCollection>.update(
        <query>,
        <update>,
        {
            upsert: <document>,
            multi:<boolean>,
            writeConcern: <document>
        }
                )
```

```
db.testcollection.update({item: “card”}, {$inc: {qty: 6}})

output: WriteResult({“Matched”: 1,”Upserted”:1,
“Modified”: 1})
```

db.testcollection.update({item: “card”}, {“txt”: “Ciao”})

output: WriteResult({“Matched”: 1,”Upserted”:0, “Modified”: 1})

ATTENZIONE USARE COMANDO $SET

db.testcollection.update({item: “card”}, {$set: {“txt”: “Ciao”}})

```
db.<nomeCollection>.remove(
            <query>,
            {
                justone: <boolean>,
                writeConcern: <document>
            }
)
```

```
db.testcollection.remove({item: “card”})
```

```
output: WriteResult({“Removed”: 1)
```

GridFS

Cos’è ?

E’ un insieme di specifiche per salvare files su MongoDB la cui dimensione eccede i 16MB

Si possono salvare SOLO files, ovvero stream di byte

Come funziona

MongoDB divide il file in chunk e salva i chunks come documenti separati.

Al momento del salvataggio del file, vengono popolate due collections, la collection fs.chunks e la collection fs.files

```
{
  "_id" : <ObjectId>,
  "files_id" : <ObjectId>,
  "n" : <num>,
  "data" : <binary>
}
```

fs.chunks

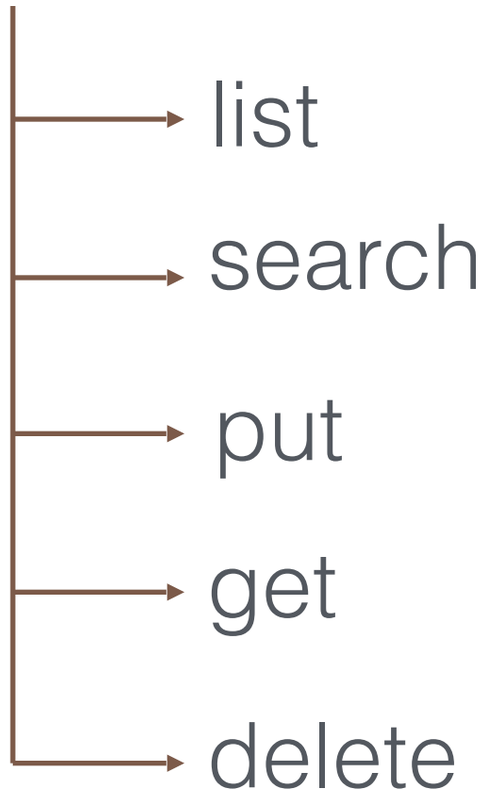
```
{
  "_id" : <ObjectId>,
  "length" : <num>,
  "chunkSize" : <num>,
  "uploadDate" : <timestamp>,
  "md5" : <hash>,
  "filename" : <string>,
  "contentType" : <string>,
  "aliases" : <string array>,
  "metadata" : <dataObject>,
}
```

fs.files

Come usarlo

1. Attraverso un driver
2. Uso dell'utility mongofiles

- mongofiles <options> <commands> <filename>



Architetture

Perchè replica dei dati

- Quanto punti di failure vuoi avere?
- Quanti sonni tranquilli vuoi dormire?
- Mai avuto problemi di connettività?
- Più nodi facilitano l’uso dei dati in modo diverso

Replica Set

Creazione ReplicaSet

Nodo 1

Nodo 2

Nodo 3

Configurazione ReplicaSet - 1 di 2

1. Definire il nome del ReplicaSet (es. zero12rs)

2. Avviare mongodb con l'opzione ReplicaSet nel 1° nodo

   $mongod --replSet zero12rs -f mongod.conf --fork

3. Avviare mongodb con l'opzione ReplicaSet nel 2° nodo

   $mongod --replSet zero12rs -f mongod.conf --fork

4. Avviare mongodb con l'opzione ReplicaSet nel 3° nodo

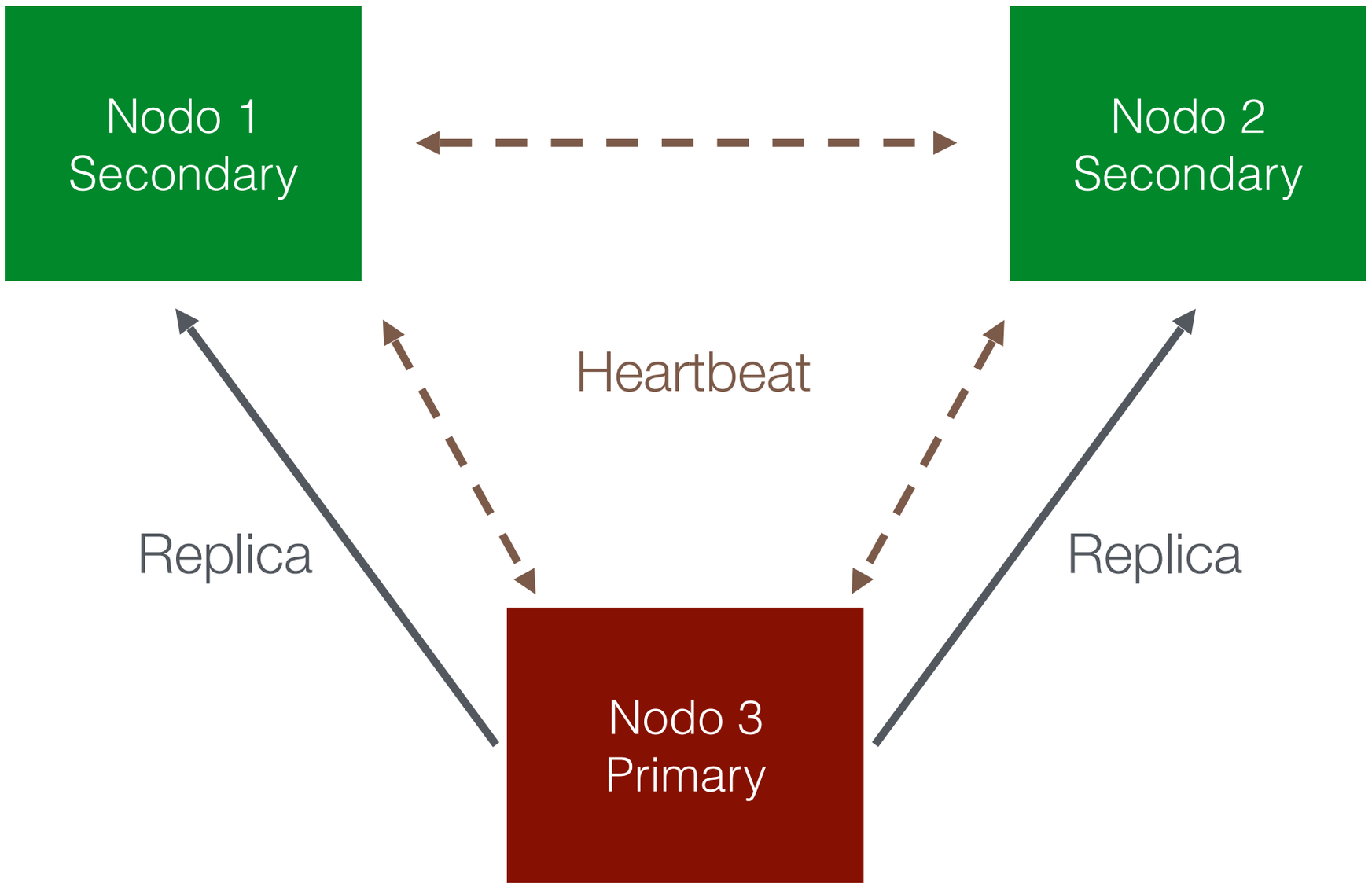
   $mongod --replSet zero12rs -f mongod.conf --fork

4. Eseguire la configurazione effettiva dei nodi replicaSet

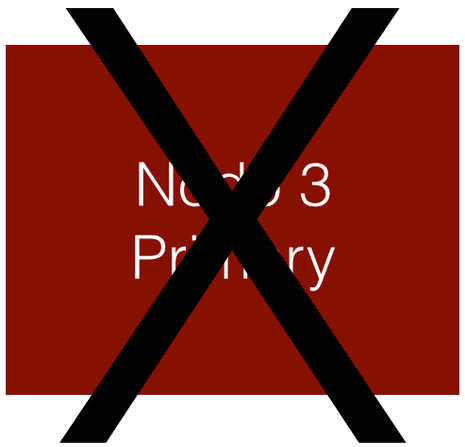
Configurazione ReplicaSet - 2 di 2

```
> conf = {
    _id : "zero12rs”,
    members : [
        {_id : 0, host : "A”},
        {_id : 1, host : "B"},
        {_id : 2, host : "C”},
    ]
}

> rs.initiate(conf)
```

ReplicaSet - 1 di 3

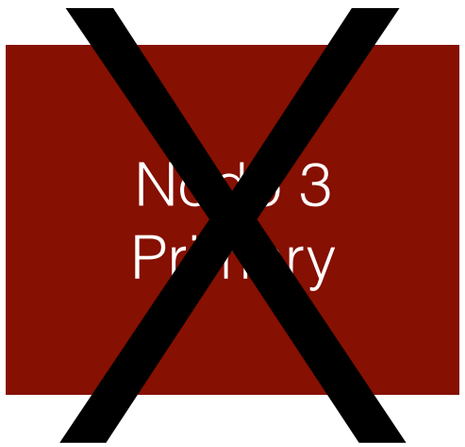


ReplicaSet - 2 di 3



FAIL !

ReplicaSet - 3 di 3

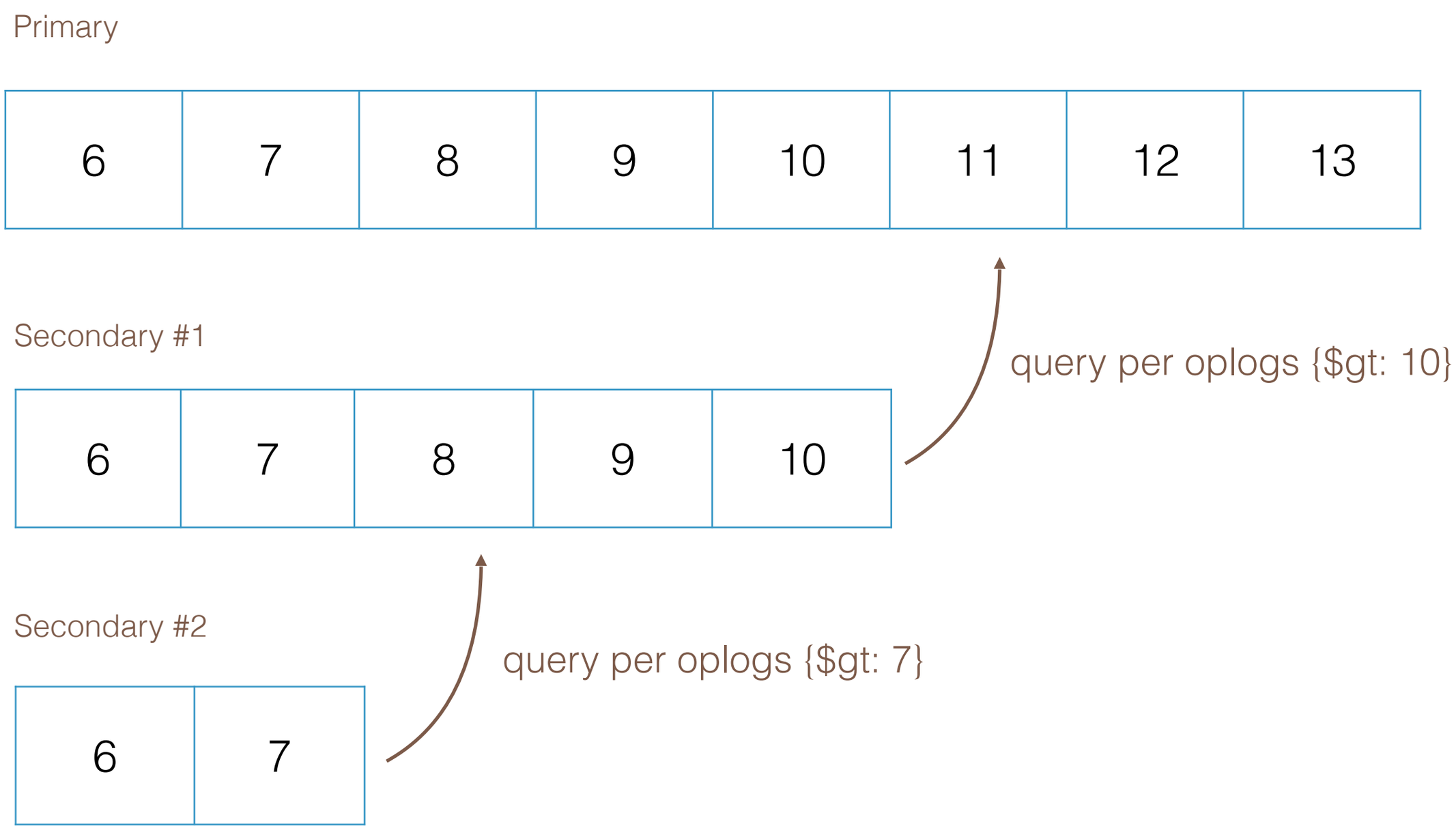


FAIL !

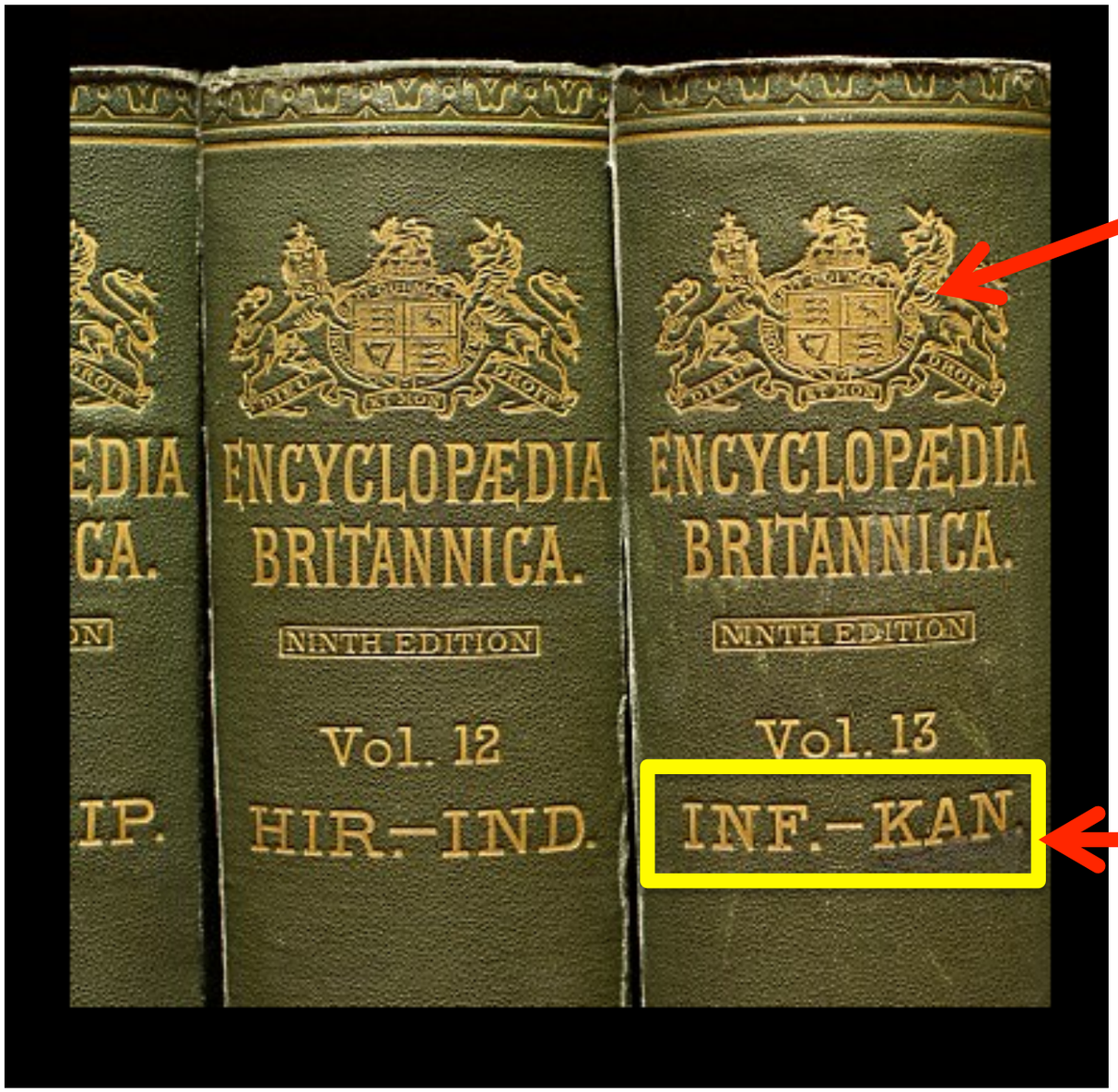
Come funziona la replica

Come funziona la replica ?

oplog: contiene la lista di tutte le operazioni di scrittura e aggiornamenti fatti nel nodo primario



Sharding



Shard

Shard key range

Elementi necessari per creare uno sharding

Mongod:

E' il processo server che esegue il database e che rende disponibile tutte i servizi per la creazione di database, collections, l'inserimento di documenti e le relative funzioni di interrogazione

Mongos:

E' il servizio di routing per processare le query provenienti dallo strato applicativo e che determina la locazione dei dati richiesti in una configurazione Sharding.

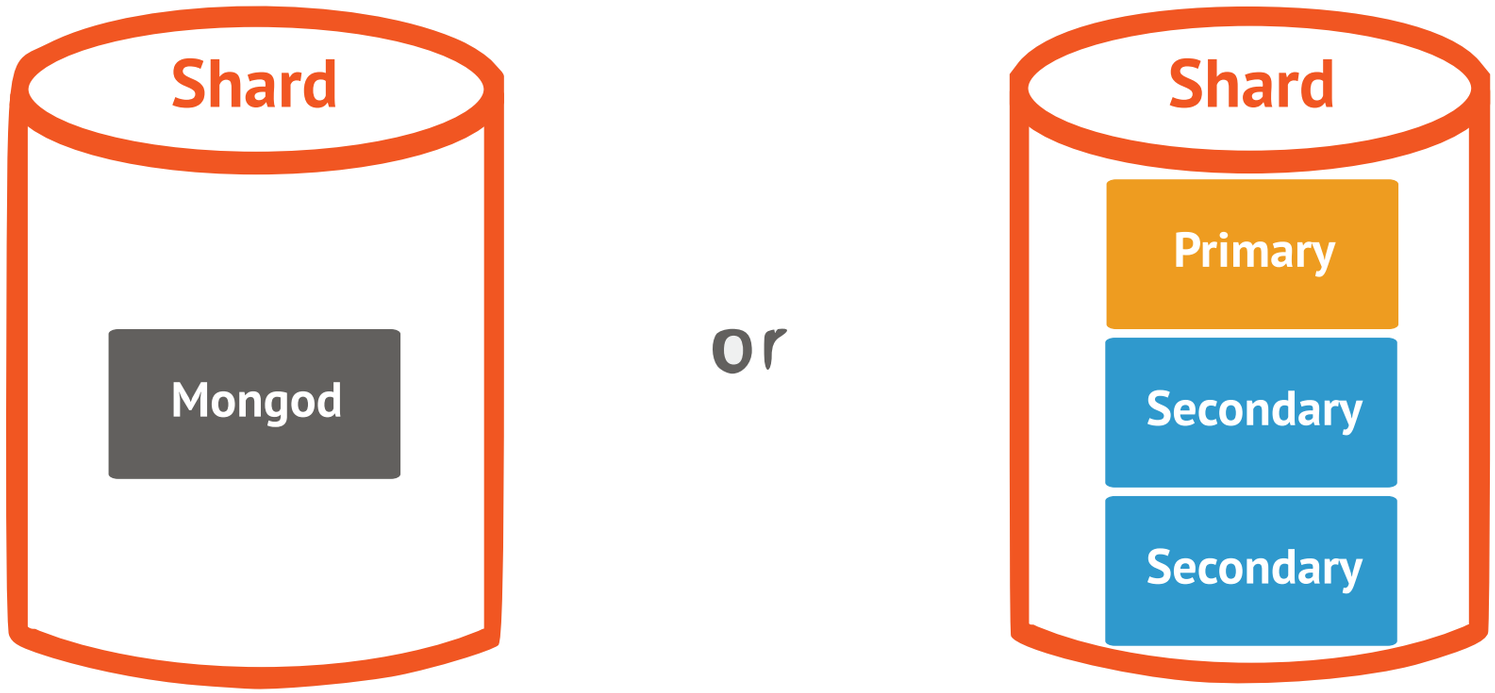
Config Server:

Rappresenta i server contenenti i metadati relativi alle chiavi di sharding e alla distribuzione delle informazioni.

Mongo:

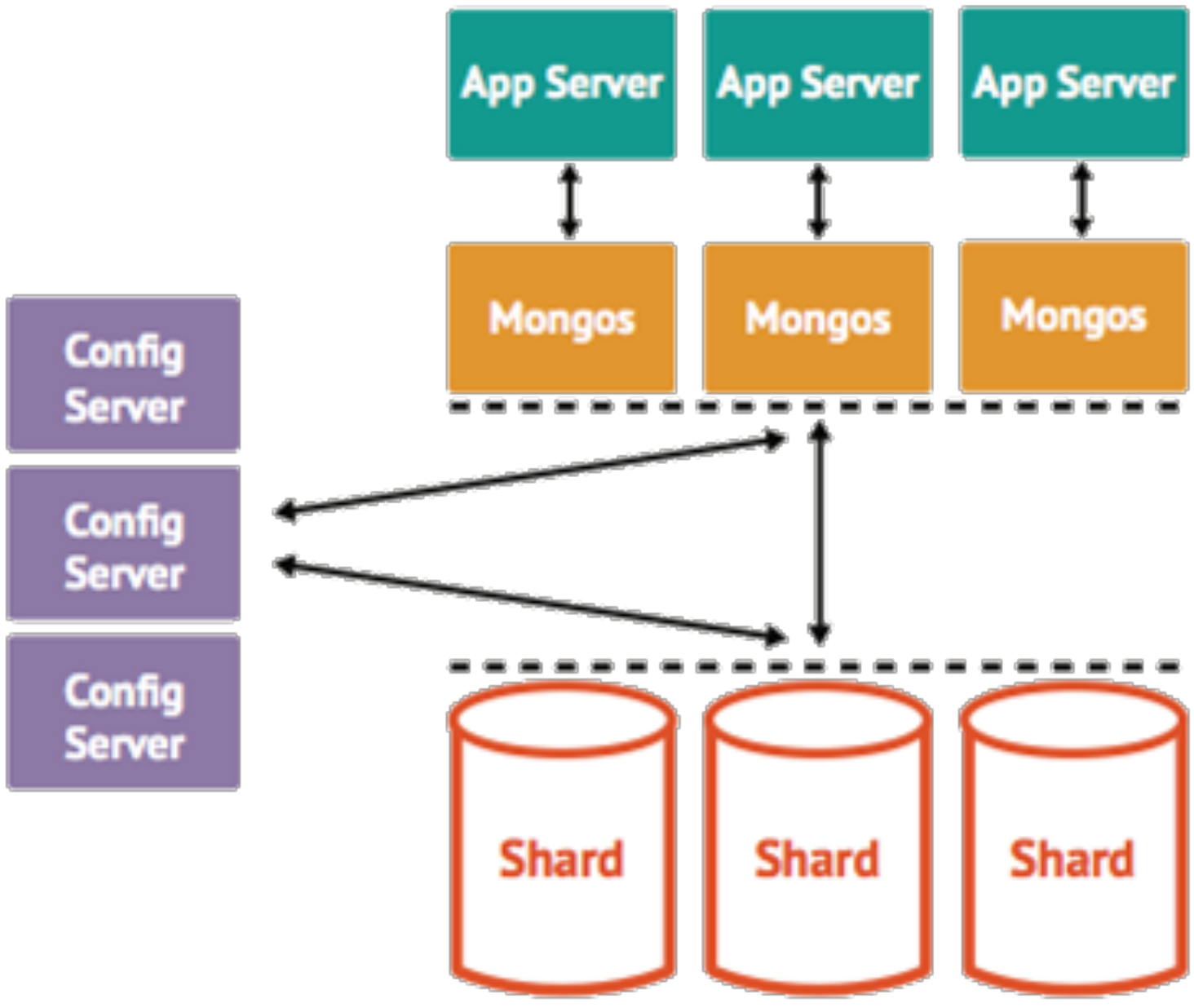
Rappresenta l'eseguibile per avviare la shell Javascript per interagire con database, collezioni e documenti che possono essere interrogati e manipolati da riga di comando.

Distribuzione dei dati
(Sharding)



- E’ un nodo di un cluster
- Può essere un singolo mongod oppure un Replica Set

Esempio architettura Sharding



Sharding Key

- La chiave di Shard è immutabile
- I valori della chiave di Shard sono immutabile
- La chiave di Shard deve essere indicizzata
- La chiave di Shard è limitata ad una dimensione di 512 bytes
- La chiave di Shard è usata per definire il routing delle query

  - Usare un campo usato nelle query

- Solamente la shard key può essere unica tra tutti gli shards

  - `_id` è unico all'interno del singolo shard

Indici

Indici

Un indice è una struttura dati che permette la localizzazione rapida delle informazioni relative al campo su cui è costruito e riveste una parte fondamentale in fase di ottimizzazione delle query.

Creare un indice in MongoDB

db.<nomeCollection>.ensureIndex(<JSON criteria>)

esempio

db.users.ensureIndex({“score”: 1})

1: ASC

-1: DES

Esempio

db.users.ensureIndex({“score”: 1})

[0] -> 0x0c965148
[0] -> 0xf51f818e
[0] -> 0x00fd7934
...
[0] -> 0xd246648f
[1] -> 0xf78d5bdd
[1] -> 0x68ab28bd
[1] -> 0x5c7fb621
...
[30] -> 0x67ded4b7
[30] -> 0x3996dd46
[30] -> 0xfce68412
[30] -> 0x91106e23

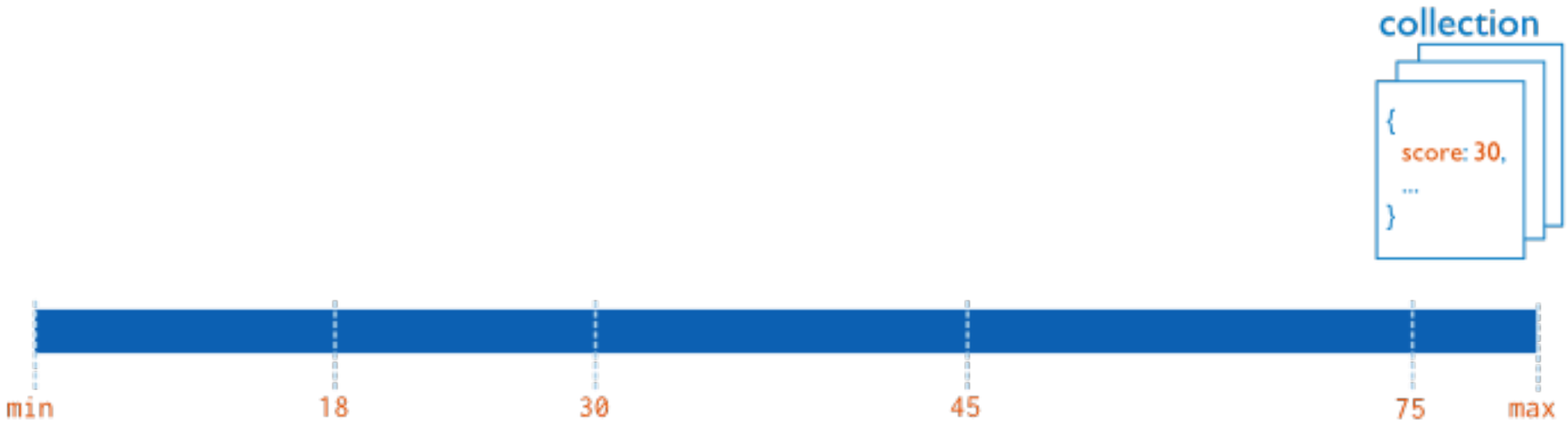
Index Type

Default _id

Tutte le collezioni hanno un indice per _id di un documento.

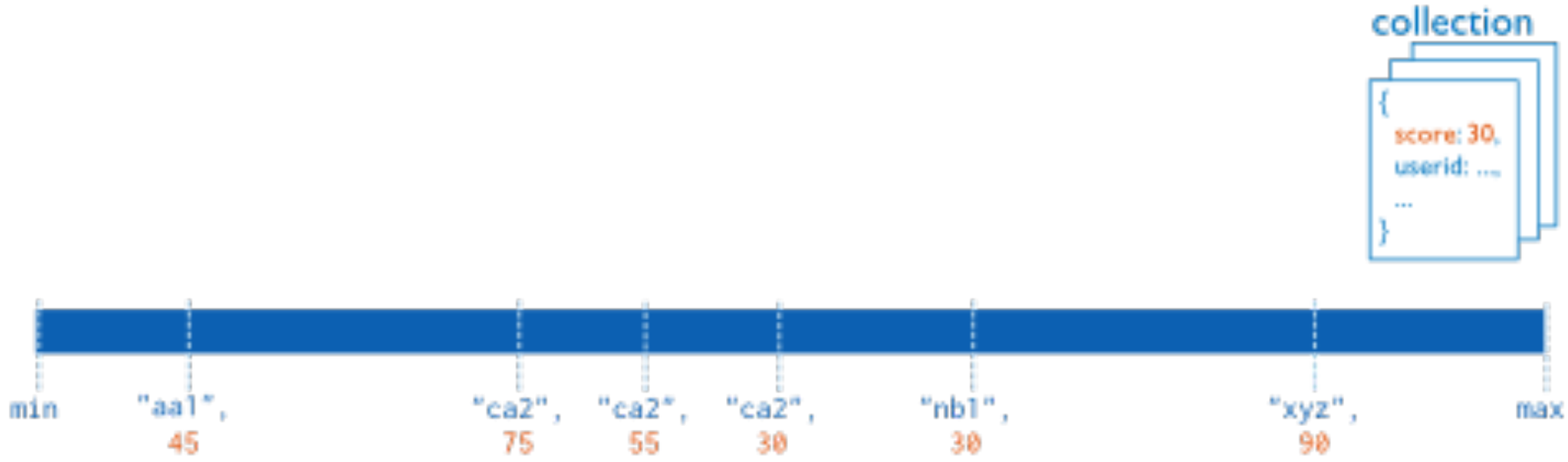
_id è unico e previene la possibilità di un client di scrivere due documenti con lo stesso valore nel campo _id

Single Field



{ score: 1 } Index

Compound Index



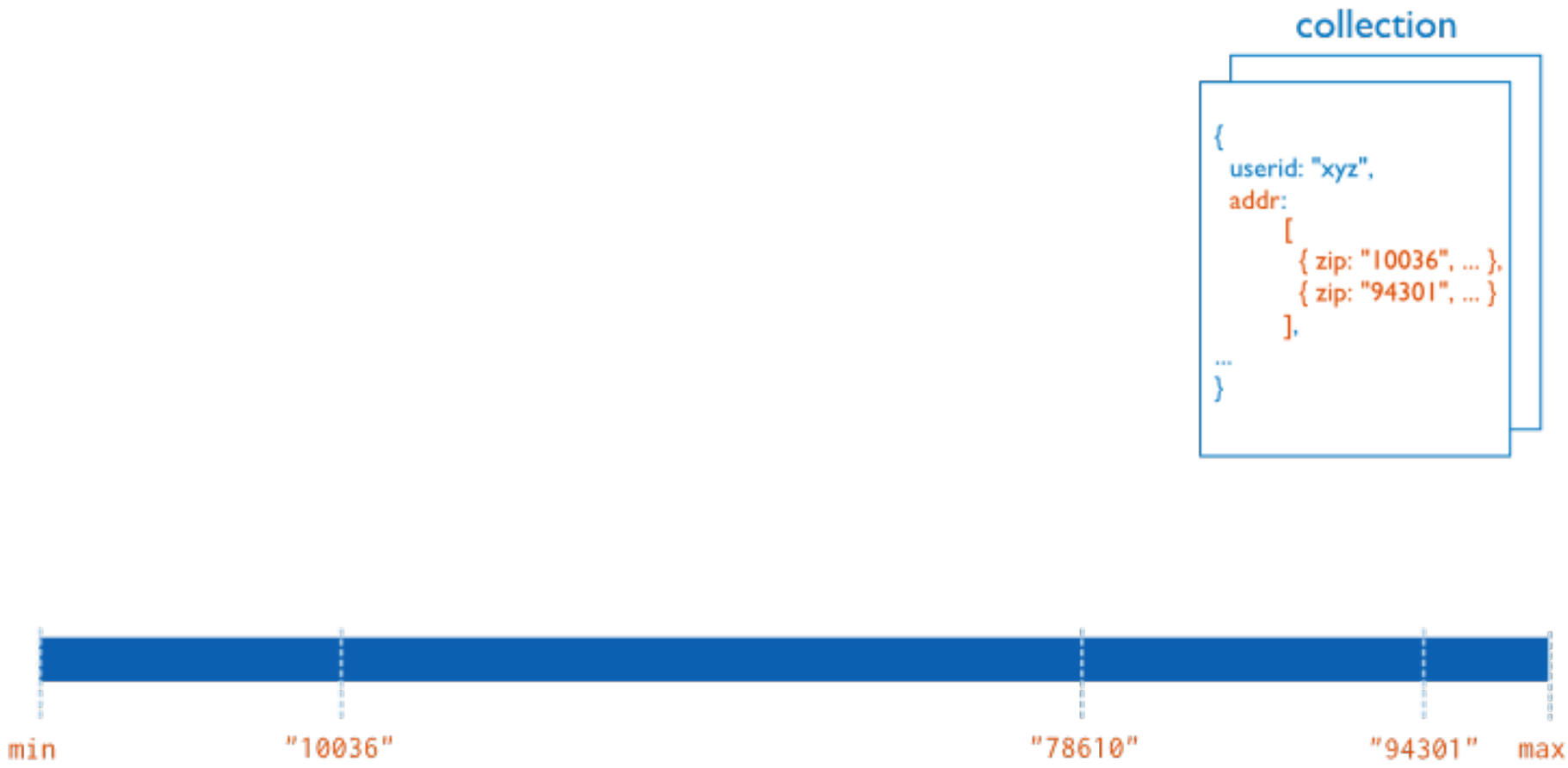
{ userid: 1, score: -1 } Index

Esempio

db.users.ensureIndex({“age”: 1, “username”: 1})

[20, "user100309"] -> 0x0c965148
[20, "user100334"] -> 0xf51f818e
[20, "user100479"] -> 0x00fd7934
...
[21, "user99985"] -> 0xd246648f
[21, "user100156"] -> 0xf78d5bdd
[21, "user100187"] -> 0x68ab28bd
[21, "user100192"] -> 0x5c7fb621
...
[21, "user999920"] -> 0x67ded4b7
[22, "user100141"] -> 0x3996dd46
[22, "user100149"] -> 0xfce68412
[22, "user100223"] -> 0x91106e23

Multikey Index



{ "addr.zip": 1 } Index

Quando usare gli indici

Quando gli indici sono utili	Quando NON sono utili gli indici
Collections Grandi	Piccole Collections
Documenti di grandi dimensioni	Documenti di piccole dimensioni
Queries selettive	Queries non selettive

#MongoDB

Data Processing and Aggregation

Massimo Brignoli

Senior Solutions Architect, MongoDB Inc.

mongoDB

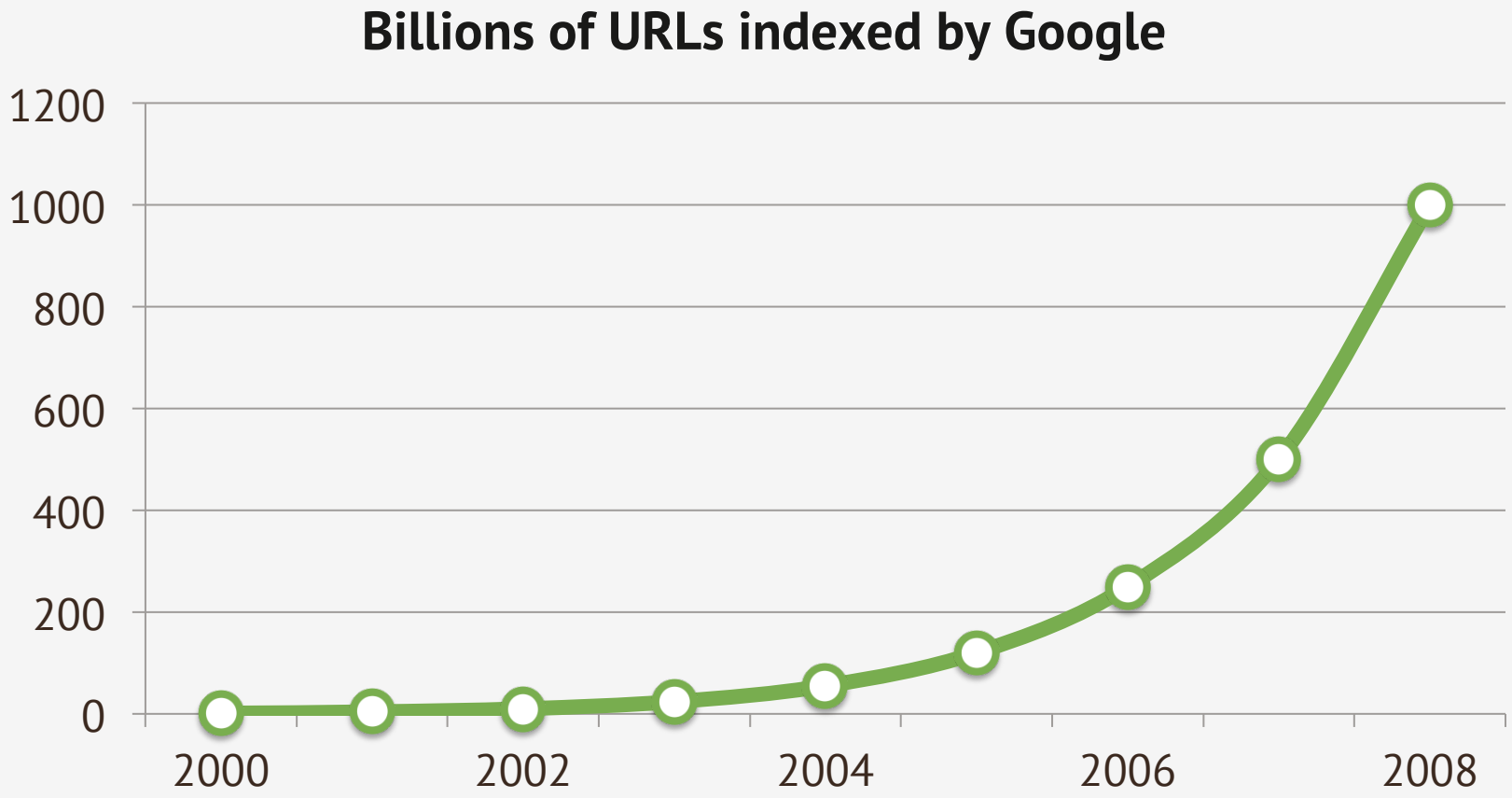
Big Data

mongoDB

Who Am I?

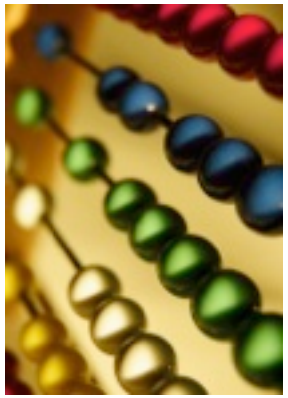
- Job Title
- X years DB Experience

Exponential Data Growth



For over a decade

Big Data == Custom Software



amazon web services

Amazon DynamoDB

Google Bigtable

In the past few years

Open source software has emerged enabling the rest of us to handle **Big Data**

How MongoDB Meets Our Requirements

- MongoDB is an **operational** database
- MongoDB provides **high performance** for **storage and retrieval** at large scale
- MongoDB has a robust query interface permitting intelligent operations
- MongoDB is ***not*** a data processing engine, but provides processing functionality

MongoDB data processing options

http://www.flickr.com/photos/torek/4444673930/

Getting Example Data

mongoDB

The "hello world" of MapReduce is counting words in a paragraph of text.

Let's try something a little more **interesting...**

What is the most popular pub name?

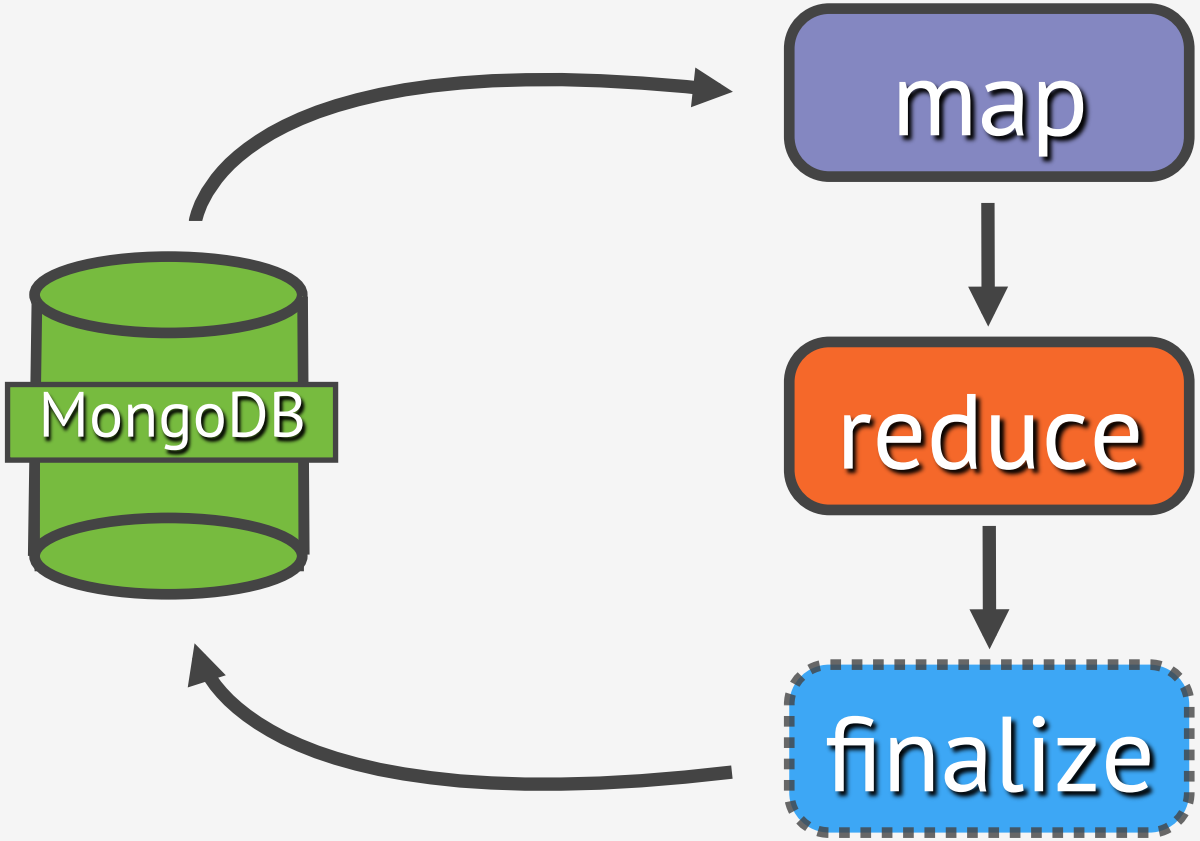
Open Street Map Data

```
#!/usr/bin/env python

# Data Source
# http://www.overpass-api.de/api/xapi?*[amenity=pub][bbox=-10.5,49.78,1.78,59]

import re
import sys

from imposm.parser import OSMParser
import pymongo

class Handler(object):
    def nodes(self, nodes):
        if not nodes:
            return
        docs = []
        for node in nodes:
            osm_id, doc, (lon, lat) = node
            if "name" not in doc:
                node_points[osm_id] = (lon, lat)
                continue
            doc["name"] = doc["name"].title().lstrip("The ").replace("And", "&")
            doc["_id"] = osm_id
            doc["location"] = {"type": "Point", "coordinates": [lon, lat]}
            docs.append(doc)
        collection.insert(docs)
```

Example Pub Data

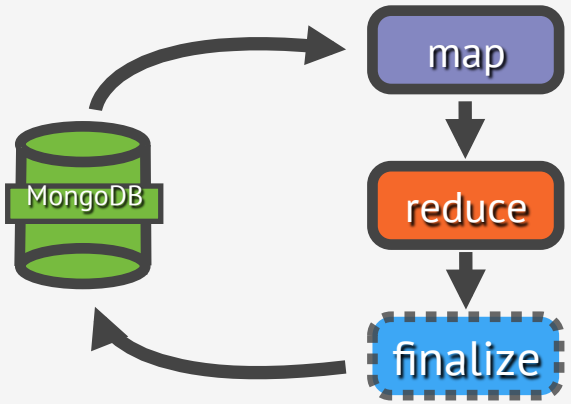

```
{
   "_id" : 451152,
   "amenity" : "pub",
   "name" : "The Dignity",
   "addr:housenumber" : "363",
   "addr:street" : "Regents Park Road",
   "addr:city" : "London",
   "addr:postcode" : "N3 1DH",
   "toilets" : "yes",
   "toilets:access" : "customers",
   "location" : {
      "type" : "Point",
      "coordinates" : [-0.1945732, 51.6008172]
   }
}
```

MongoDB MapReduce

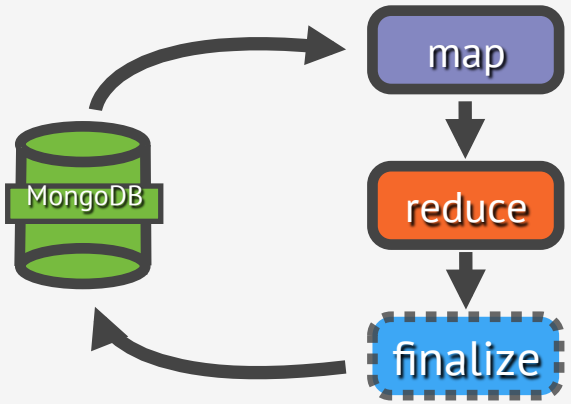


Map Function

```
> var map = function() {

    emit(this.name, 1);
```



Reduce Function




```
> var reduce = function (key, values) {
    var sum = 0;
    values.forEach( function (val) {sum += val;} );
    return sum;
  }
```

Results

```
> db.pub_names.find().sort({value: -1}).limit(10)

{ "_id" : "The Red Lion", "value" : 407 }
{ "_id" : "The Royal Oak", "value" : 328 }
{ "_id" : "The Crown", "value" : 242 }
{ "_id" : "The White Hart", "value" : 214 }
{ "_id" : "The White Horse", "value" : 200 }
{ "_id" : "The New Inn", "value" : 187 }
{ "_id" : "The Plough", "value" : 185 }
{ "_id" : "The Rose & Crown", "value" : 164 }
{ "_id" : "The Wheatsheaf", "value" : 147 }
{ "_id" : "The Swan", "value" : 140 }
```

The Wheatsheaf
The Red Lion
The Royal Oak
The White Horse
The Rose And Crown
The Crown Inn
The Coach And Horses
The Bell
The Kings Head
The Black Horse
The Cross Keys
The Swan Inn
The Railway
The Crown
The New Inn
The White Hart
The Prince of Wales
The Plough
The Rising Sun
The Kings Arms
The George
The Ship
The Cricketers
The Bull
The Castle Inn
The Queen'S Head
The Fox And Hounds
The Ship Inn
The Swan
The Black Bull
The Star
The Green Man
The Nags Head
The Bulls Head
The Rose & Crown
The Anchor Inn
The Bay Horse
The Duke Of York
The Robin Hood
The Shoulder Of Mutton
The Three Horseshoes
The Horse And Jockey
The Black Swan
The George And Dragon
The Coach & Horses
The Three Tuns
The Seven Stars
The King'S Arms
The Sun Inn
The Railway Inn
The Bell Inn
The Windmill
The Duke Of Wellington
The Waggon And Horses
The Station
The Woodman
The Miners Arms
The Beehive
The Lamb Inn
The Lamb
The Butchers Arms
The Railway House
The George Inn
The Bird In Hand
The Angel
The Greyhound
The Fox
The Village Inn
The George & Dragon
The Bridge Inn
The Dolphin
The Black Lion
The Green Dragon
The Plough Inn
The White Swan
The Masons Arms
The Anchor
The Star Inn
The Golden Lion
The Victoria
The Queens Head
The Castle
The Hare And Hounds
The Five Bells
The Fountain
The Albion
The Griffin
The Forresters Arms
The Railway Tavern
The Half Moon
The Barley Mow
The Blue Bell
The Lord Nelson
The King'S Head
The Sun
The White Lion

Pub Names in the Center of London

```
> db.pubs.mapReduce(map, reduce, { out: "pub_names",
    query: {
     location: {
      $within: { $centerSphere: [[-0.12, 51.516], 2 / 3959] }
     }}
  })

{
    "result" : "pub_names",
    "timeMillis" : 116,
    "counts" : {
        "input" : 643,
        "emit" : 643,
        "reduce" : 54,
        "output" : 537
    },
    "ok" : 1,
}
```

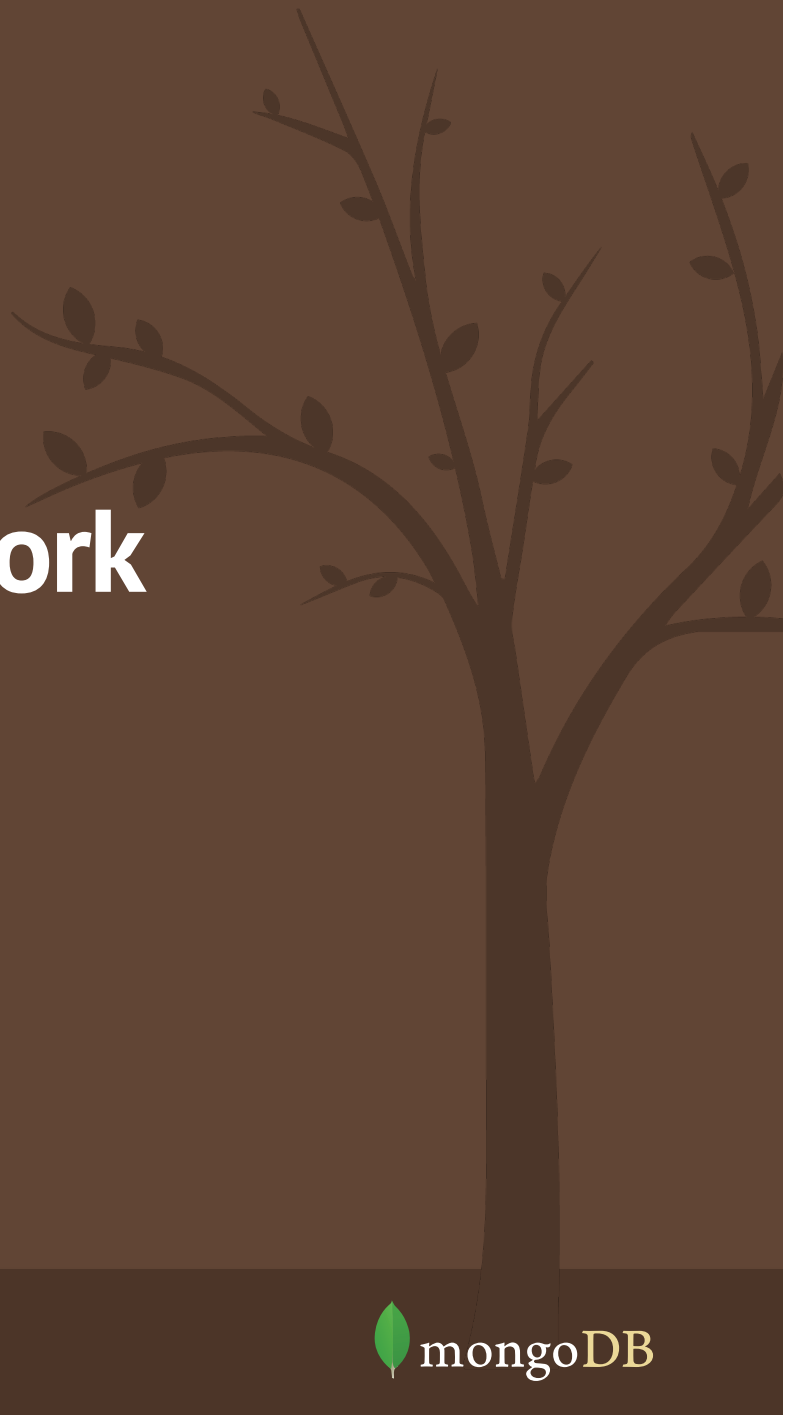
Results

```
> db.pub_names.find().sort({value: -1}).limit(10)

{ "_id" : "All Bar One", "value" : 11 }
{ "_id" : "The Slug & Lettuce", "value" : 7 }
{ "_id" : "The Coach & Horses", "value" : 6 }
{ "_id" : "The Green Man", "value" : 5 }
{ "_id" : "The Kings Arms", "value" : 5 }
{ "_id" : "The Red Lion", "value" : 5 }
{ "_id" : "Corney & Barrow", "value" : 4 }
{ "_id" : "O'Neills", "value" : 4 }
{ "_id" : "Pitcher & Piano", "value" : 4 }
{ "_id" : "The Crown", "value" : 4 }
```

MongoDB MapReduce

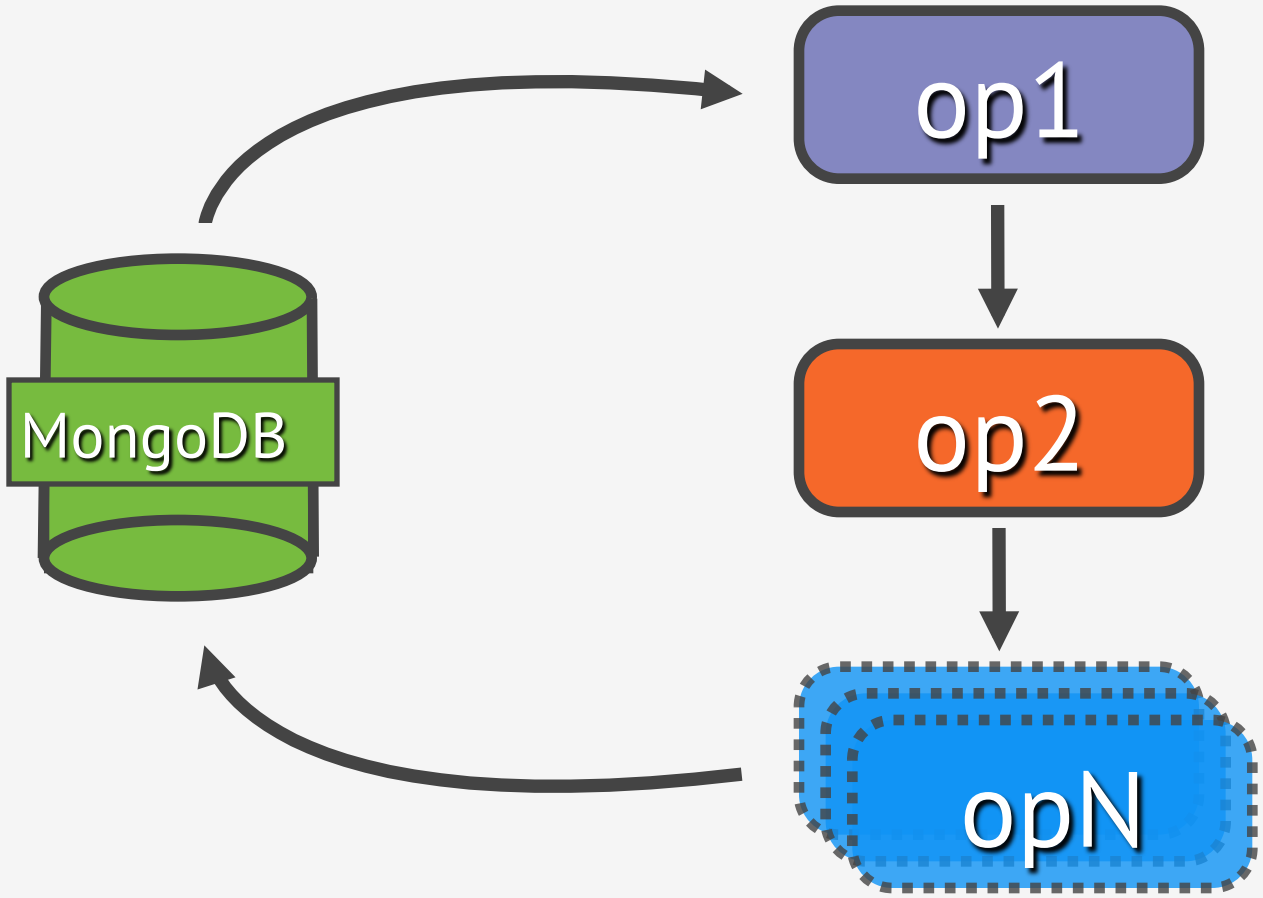
- Real-time
- Output directly to document or collection
- Runs inside MongoDB on local data
  - Adds load to your DB
  - In Javascript – debugging can be a challenge
  - Translating in and out of C++

Aggregation Framework



mongoDB

Aggregation Framework



Aggregation Framework in 60 Seconds

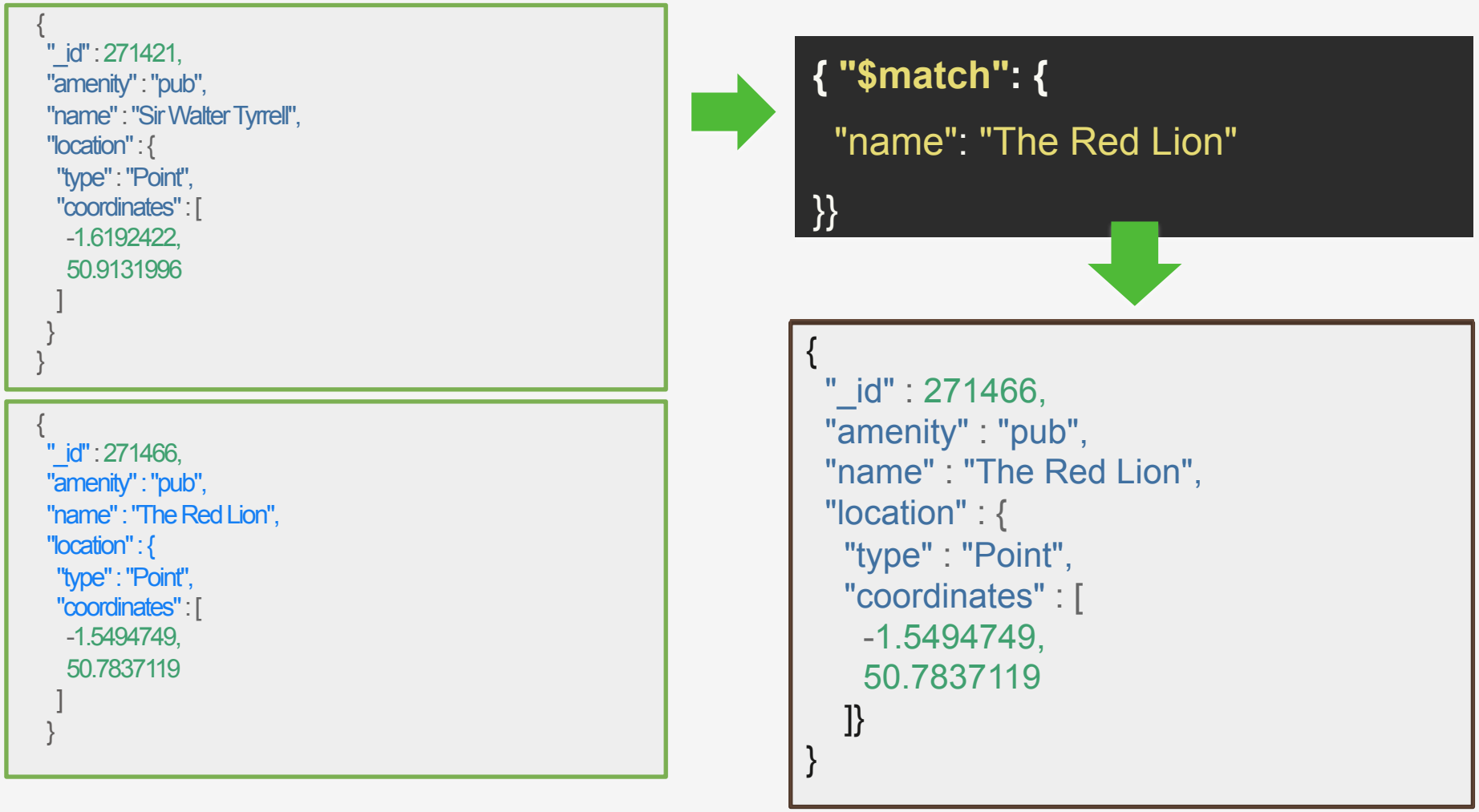
Aggregation Framework Operators

- $project
- $match
- $limit
- $skip
- $sort
- $unwind
- $group

$match

- Filter documents
- Uses existing query syntax
- If using $geoNear it has to be first in pipeline
- $where is not supported

Matching Field Values



$project

- Reshape documents
- Include, exclude or rename fields
- Inject computed fields
- Create sub-document fields

Including and Excluding Fields

```
{
  "_id" : 271466,
  "amenity" : "pub",
  "name" : "The Red Lion",
  "location" : {
    "type" : "Point",
    "coordinates" : [
      -1.5494749,
      50.7837119
    ]
  }
}
```

```
{ “$project”: {

“_id”: 0,
“amenity”: 1,
“name”: 1,

}}
```

```
{
“amenity” : “pub”,
“name” : “The Red Lion”
}
```

Reformatting Documents

```
{
  "_id" : 271466,
  "amenity" : "pub",
  "name" : "The Red Lion",
  "location" : {
    "type" : "Point",
    "coordinates" : [
      -1.5494749,
      50.7837119
    ]
  }
}
```

```
{ “$project”: {

“_id”: 0,
“name”: 1,
“meta”: {
        “type”: “$amenity”}
}}
```

```
{
“name” : “The Red Lion”
“meta” : {
        “type” : “pub”
}}
```

$group

- Group documents by an ID
- Field reference, object, constant
- Other output fields are computed
  $max, $min, $avg, $sum
  $addToSet, $push $first, $last
- Processes all data in memory

Back to the pub!

- http://www.offwestend.com/index.php/theatres/pastshows/71

Popular Pub Names

```
>var popular_pub_names = [
{ $match : location:
      { $within: { $centerSphere:
            [[-0.12, 51.516], 2 / 3959]}}}
},
{ $group :
      { _id: “$name”
      value: {$sum: 1} }
},
{ $sort : {value: -1} },
{ $limit : 10 }
```



Results

```
> db.pubs.aggregate(popular_pub_names)
{
    "result" : [
        { "_id" : "All Bar One", "value" : 11 }
        { "_id" : "The Slug & Lettuce", "value" : 7 }
        { "_id" : "The Coach & Horses", "value" : 6 }
        { "_id" : "The Green Man", "value" : 5 }
        { "_id" : "The Kings Arms", "value" : 5 }
        { "_id" : "The Red Lion", "value" : 5 }
        { "_id" : "Corney & Barrow", "value" : 4 }
        { "_id" : "O'Neills", "value" : 4 }
        { "_id" : "Pitcher & Piano", "value" : 4 }
        { "_id" : "The Crown", "value" : 4 }
    ],
    "ok" : 1
}
```

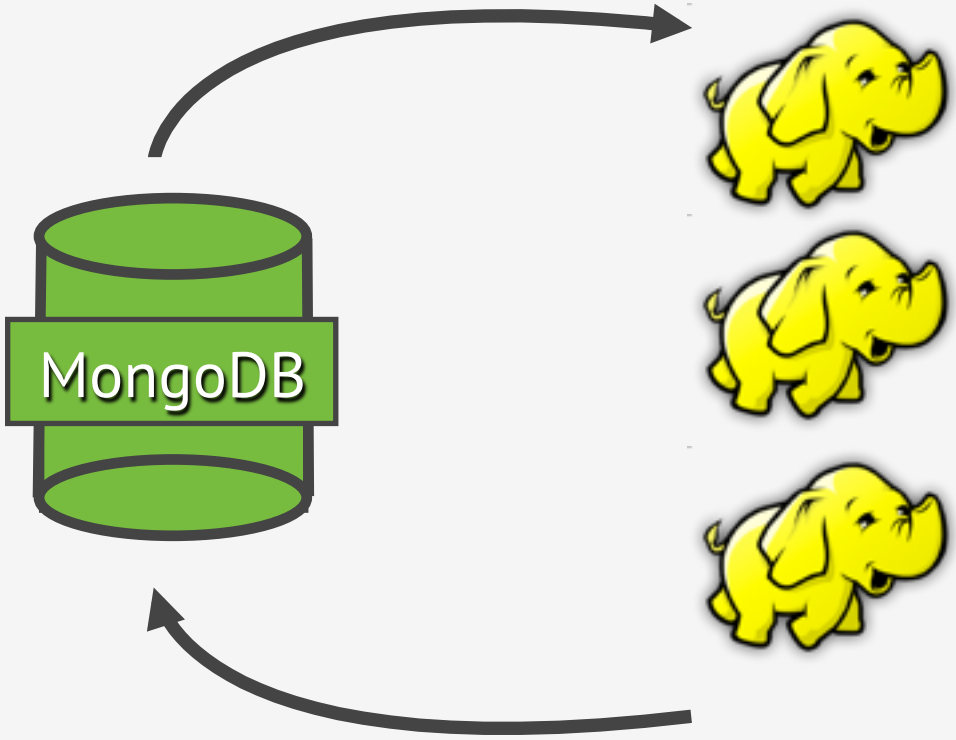
Aggregation Framework Benefits

- Real-time
- Simple yet powerful interface
- Declared in JSON, executes in C++
- Runs inside MongoDB on local data
  - Adds load to your DB
  - Limited Operators
  - Data output is limited

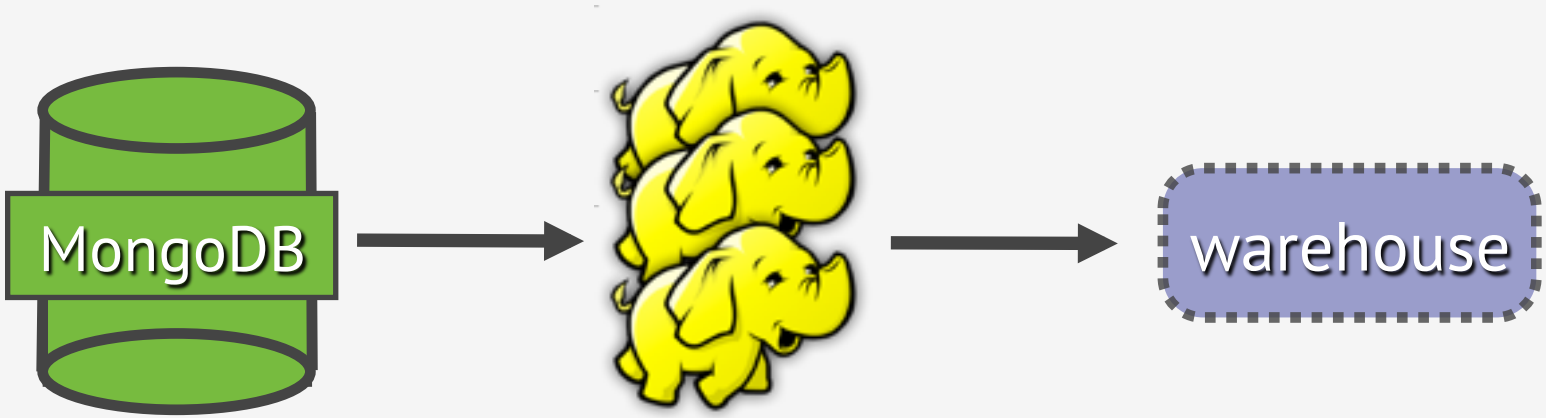
Analyzing MongoDB Data in External Systems

mongoDB

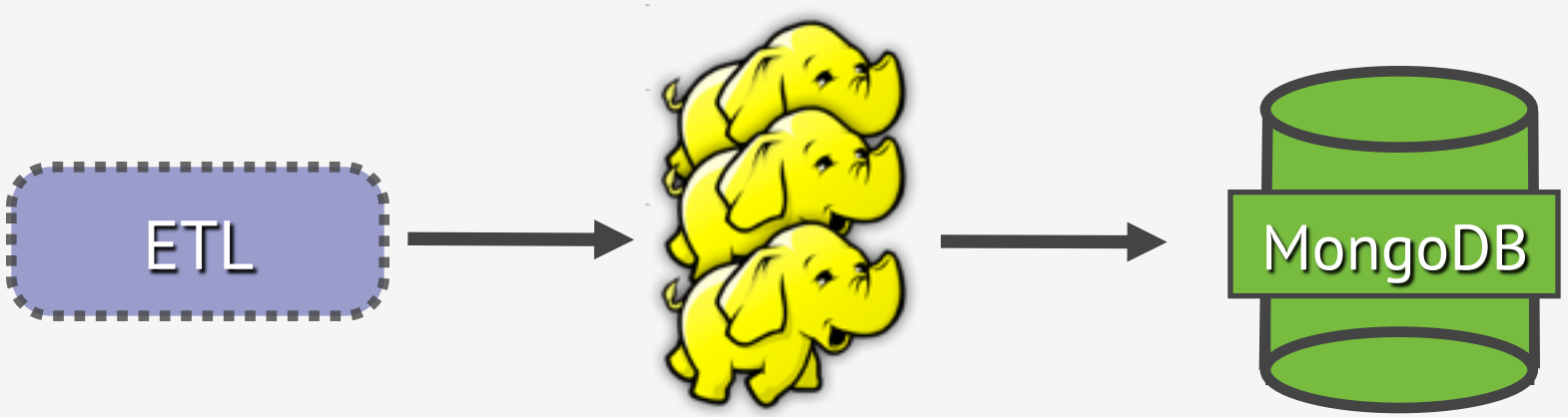
MongoDB with Hadoop



MongoDB with Hadoop



MongoDB with Hadoop



Map Pub Names in Python


```
#!/usr/bin/env python
from pymongo_hadoop import BSONMapper

def mapper(documents):
    bounds = get_bounds() # ~2 mile polygon
    for doc in documents:
        geo = get_geo(doc["location"]) # Convert the geo type
        if not geo:
            continue
        if bounds.intersects(geo):
            yield {'_id': doc['name'], 'count': 1}

BSONMapper(mapper)
print >> sys.stderr, "Done Mapping."
```

Reduce Pub Names in Python


```
#!/usr/bin/env python

from pymongo_hadoop import BSONReducer

def reducer(key, values):
    _count = 0
    for v in values:
        _count += v['count']
    return {'_id': key, 'value': _count}

BSONReducer(reducer)
```

Execute MapReduce

```
hadoop jar target/mongo-hadoop-streaming-assembly-1.1.0-rc0.jar \
-mapper examples/pub/map.py \
-reducer examples/pub/reduce.py \
-mongo mongodb://127.0.0.1/demo.pubs \
-outputURI mongodb://127.0.0.1/demo.pub_names
```

Popular Pub Names Nearby

```
> db.pub_names.find().sort({value: -1}).limit(10)

{ "_id" : "All Bar One", "value" : 11 }
{ "_id" : "The Slug & Lettuce", "value" : 7 }
{ "_id" : "The Coach & Horses", "value" : 6 }
{ "_id" : "The Kings Arms", "value" : 5 }
{ "_id" : "Corney & Barrow", "value" : 4 }
{ "_id" : "O'Neills", "value" : 4 }
{ "_id" : "Pitcher & Piano", "value" : 4 }
{ "_id" : "The Crown", "value" : 4 }
{ "_id" : "The George", "value" : 4 }
{ "_id" : "The Green Man", "value" : 4 }
```

MongoDB and Hadoop

- Away from data store
- Can leverage existing data processing infrastructure
- Can horizontally scale your data processing

- Offline batch processing
- Requires synchronisation between store & processor
- Infrastructure is much more complex

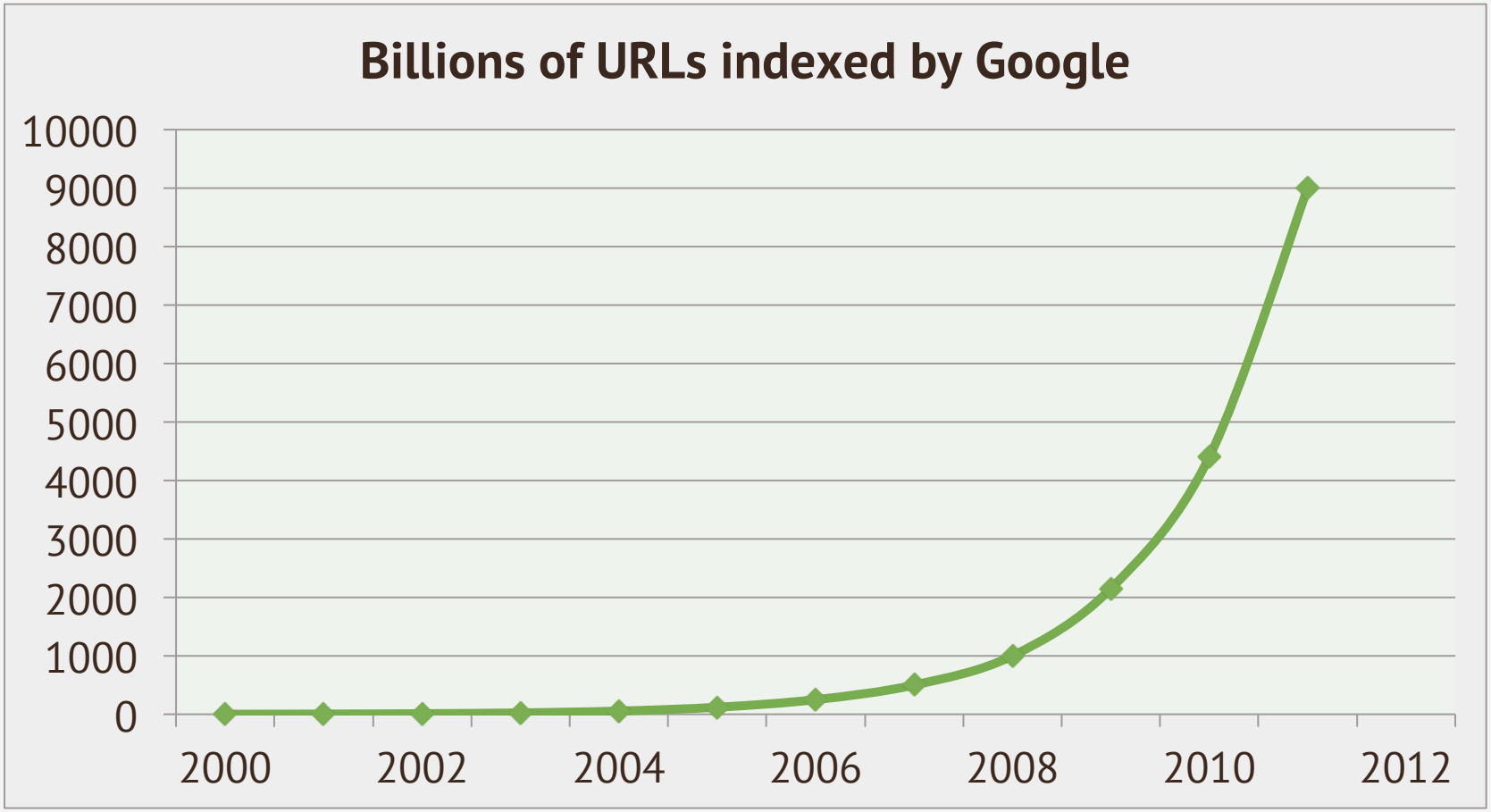
The Future of Big Data and MongoDB

mongoDB

What is Big Data?

Big Data today will be normal tomorrow

Exponential Data Growth



MongoDB enables you to

scale big

MongoDB is evolving

so you can process the big

Data Processing with MongoDB

- Process **in** MongoDB using **Map/Reduce**
- Process **in** MongoDB using **Aggregation Framework**
- Process **outside** MongoDB using **Hadoop** and other external tools

MongoDB Integration

- **Hadoop**
  https://github.com/mongodb/mongo-hadoop
- **Storm**
  https://github.com/christkv/mongo-storm
- **Disco**
  https://github.com/mongodb/mongo-disco

Questions?

mongoDB

#mongodb

Thanks!

Massimo Brignoli

massimo@mongodb.com

mongoDB