

JSON stores (and MongoDB)

**Slides Courtesy (with some
modifications): Ioana Manolescu**

Presented by: Madhulika Mohanty
Inria and Ecole polytechnique

Motivation and outline

- **JSON** (JavaScript Object Notation) allows to describe nested, potentially heterogeneous data
 - Very flexible
 - Hugely adopted for sharing (Open) Data
 - Important to know how to work with JSON
- **MongoDB**: most widely used JSON store
 - Also considered flagship NoSQL system
 - Specific approach for *replication* (and consistency), *distribution*

Sample JSON document: Twitter

```
{ "results": [
  { "text": "@twitterapi http://tinyurl.com/ctrefg",
    "to_user_id": 396524,
    "to_user": "TwitterAPI",
    "from_user": "jkoum",
    "metadata": { "result_type": "popular", "recent_retweets": 109 },
    "id": 1478555574,
    "from_user_id": 1833773,
    "iso_language_code": "nl",
    "source": "twitter< /a>",
    "profile_image_url": http://s3.amazonaws.com/twitter/a155\_b\_normal.jpg,
    "created_at": "Wed, 08 Apr 2009 19:22:10 +0000"},
  ... truncated ... ],
  "refresh_url": "?since_id=1480307926&q=%40twitterapi",
  "results_per_page": 15,
  "next_page": "?page=2&max_id=1480307926&q=%40twitterapi",
  "completed_in": 0.031704,
  "page": 1,
  "query": "%40twitterapi" }
}
```

More JSON from <http://nosdeputes.fr>

```
{
  "deputes": [
    {
      "depute": {
        "id": 1,
        "nom": "Cédric Roussel",
        "nom_de_famille": "Roussel",
        "prenom": "Cédric",
        "sexe": "H",
        "date_naissance": "1972-10-10",
        "lieu_naissance": "Brest (Finistère)",
        "num_deptmt": "06",
        "nom_circo": "Alpes-Maritimes",
        "num_circo": 3,
        "mandat_debut": "2017-06-21",
        "groupe_sigle": "LREM",
        "parti_ratt_financier": "La République en Marche",
        "sites_web": [
          {
            "site": "https://twitter.com/CedricRoussel06",
            "site": "http://cedricroussel.en-marche.fr",
            "site": "https://fr-fr.facebook.com/CRoussel06/",
            "site": "https://www.cedricroussel.fr"
          }
        ],
        "emails": [
          {
            "email": "cedric.roussel@assemblee-nationale.fr",
            "email": "6circo03@en-marche.fr"
          }
        ],
        "adresses": [
          {
            "adresse": "Assemblée nationale, 126 Rue de l'Université, 75355 Paris 07 SP"
          }
        ],
        "collaborateurs": [
          {
            "collaborateur": "M. Yanis Lahmeri",
            "collaborateur": "Mme Justine Birot",
            "collaborateur": "Mme Morgane Reclus",
            "collaborateur": "Mme Caroline Puiss\u00e9gur-Ripet",
            "collaborateur": "M. Yassine Id-Nasser Medjani"
          }
        ],
        "autres_mandats": [],
        "anciens_autres_mandats": [],
        "anciens_mandats": [
          {
            "mandat": "21/06/2017/"
          }
        ],
        "profession": "Conseiller en gestion de patrimoine indépendant",
        "place_en_hemicycle": 309,
        "url_an": "http://www2.assemblee-nationale.fr/deputes/fiche/OMC_PA718902",
        "id_an": "718902",
        "slug": "cedric-roussel",
        "url_nosdeputes": "https://www.nosdeputes.fr/cedric-roussel",
        "url_nosdeputes_api": "https://www.nosdeputes.fr/cedric-roussel/json",
        "nb_mandats": 1,
        "twitter": "CedricRoussel06"
      }
    },
    {
      "depute": {
        "id": 2,
        "nom": "Nadia Hai",
        ...
      }
    }
  ]
}
```

As:

Accueil | Vos députés | Travaux parlés

Accueil > Les députés > Liste alph... > M. Cédric Roussel

M. Cédric Roussel
Alpes-Maritimes (3^e circonscription)
Mandat en cours

Partager

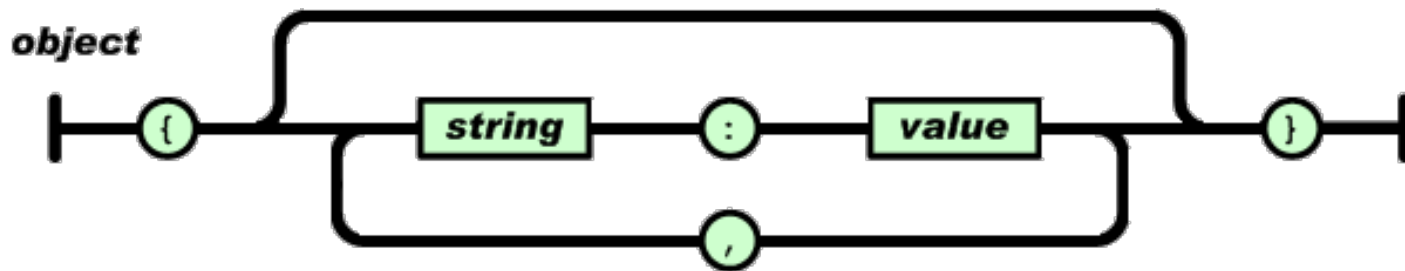
Commission
Biographie
Suppléante
Contact

La République en Marche

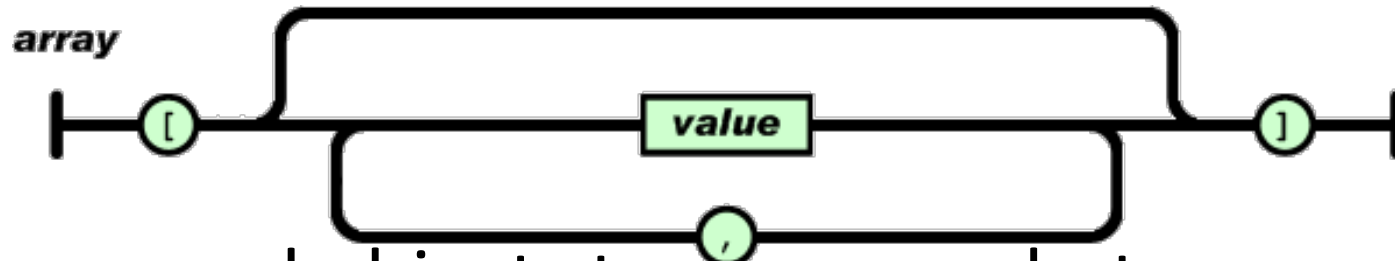
Rattachement au titre du financement de la vie politique
Déclaration d'intérêts et d'activités

JSON document structure

- Object: collection of (name, value) pairs



- Array: collection of values

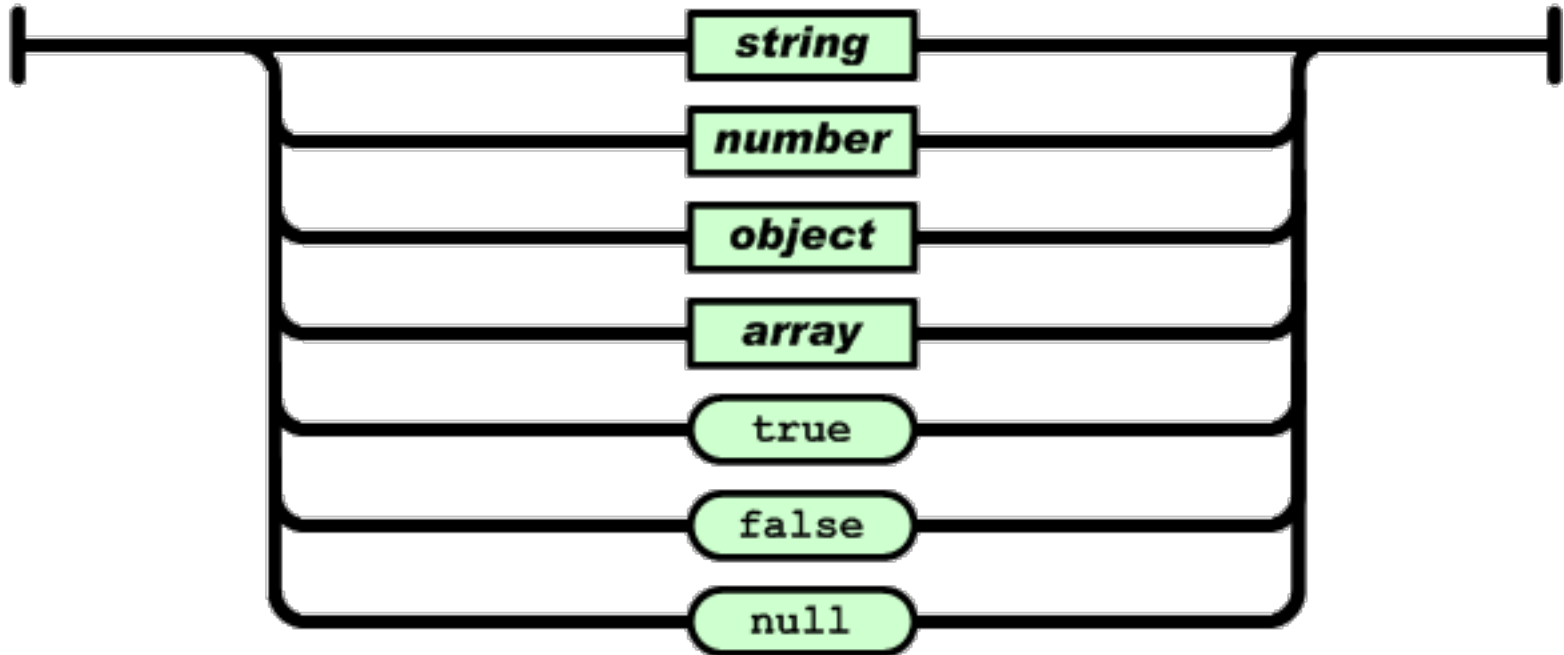


- Arrays and object structure are heterogeneous (no schema)

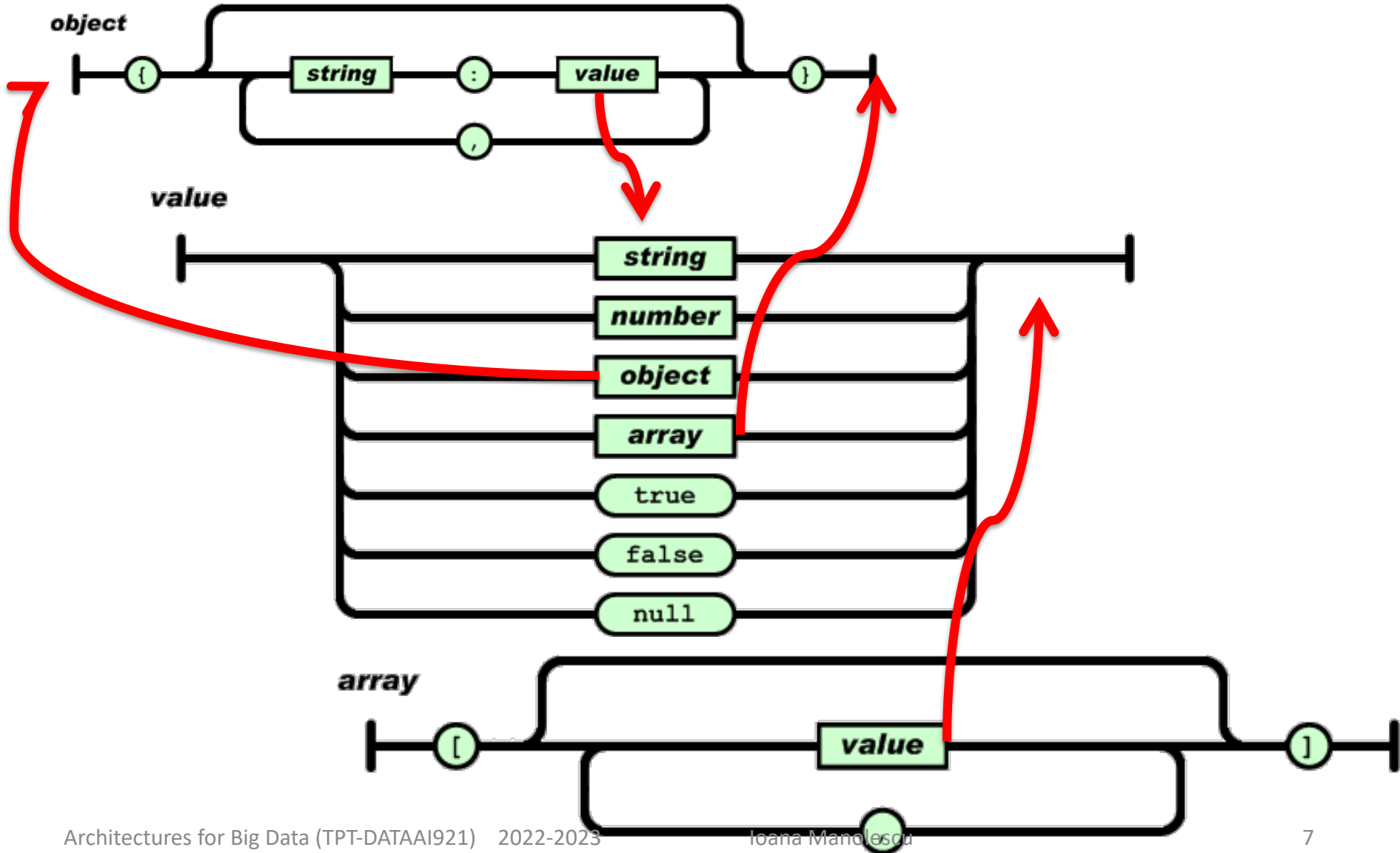
JSON document structure

- Values (allow nesting):

value

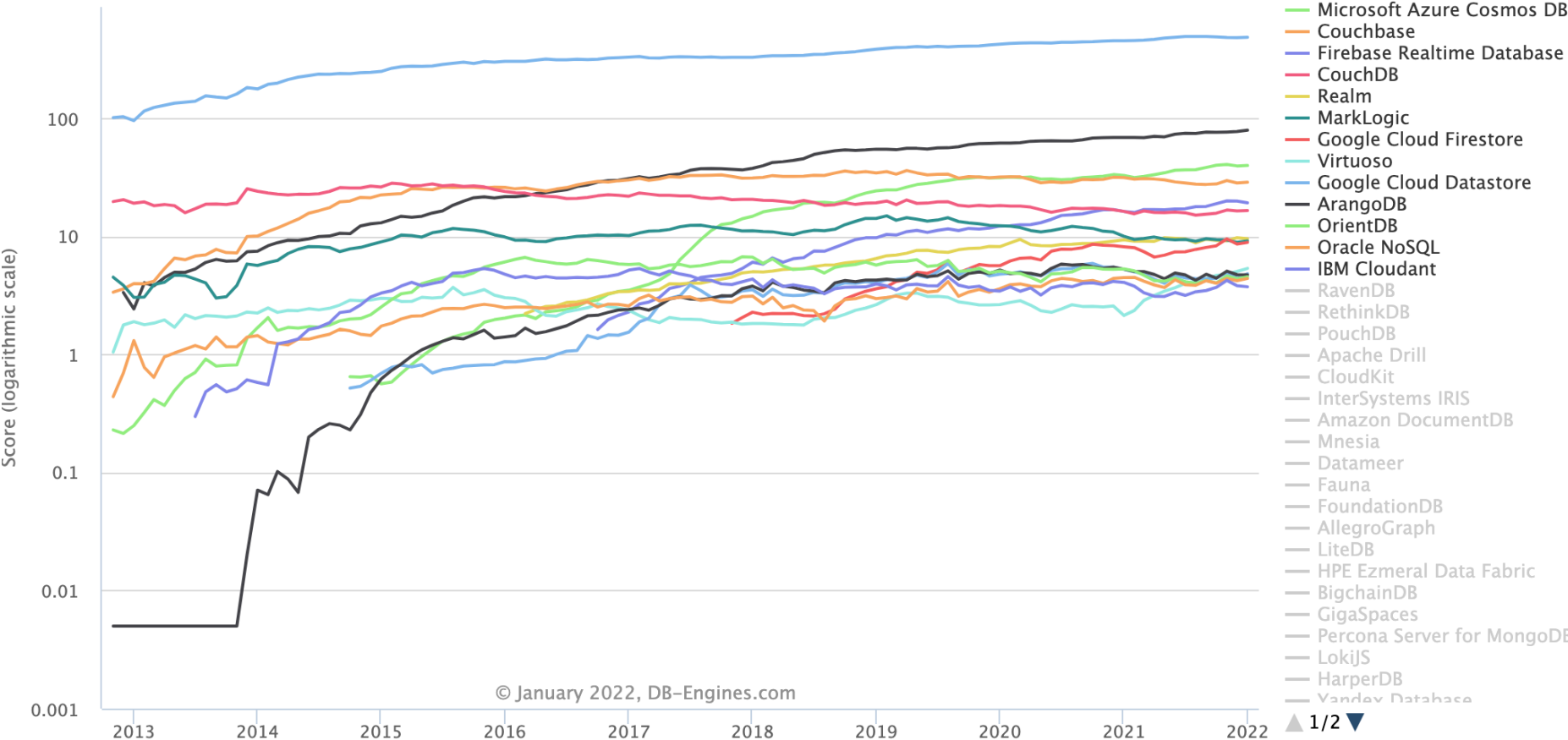


JSON document structure



MongoDB: a JSON document store

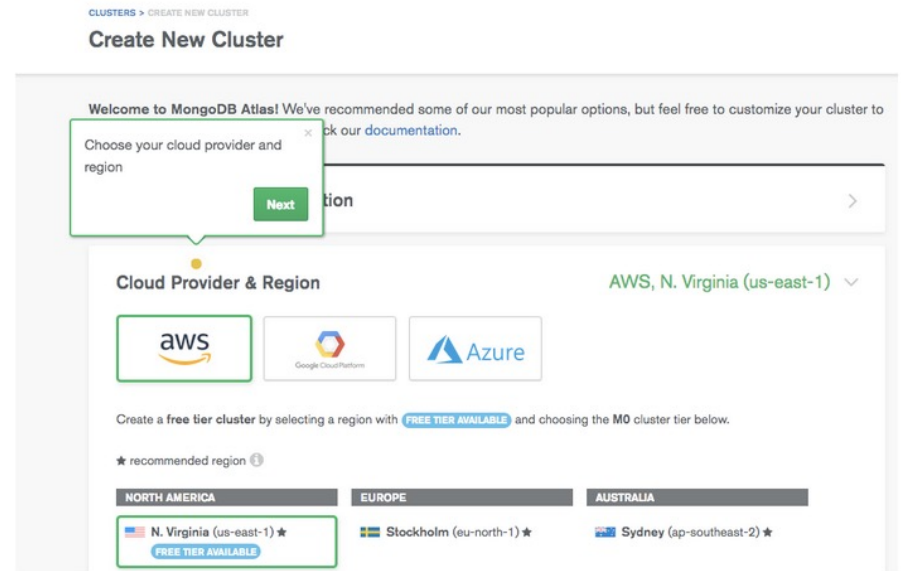
DB-Engines Ranking of Document Stores



Computed based on: popularity in search engine results, queries, job offers, social networks, questions on StackOverflow...

MongoDB tools

- MongoDB **server (mongod)**
 - Installed on top of an operating system (e.g., on a PC); assumes a file system and regular OS support
 - Accessed through (potentially concurrent) clients (mongo)
- MongoDB **Atlas (or cloud edition)**
 - Meant to deploy easily in the cloud (e.g. Amazon, Google, Azure...)
 - Assumes the respective cloud infrastructure services
- Other MongoDB tools:
 - Data lake
 - Charts to generate reporting images
 - Compass GUI to visualize the data
 - Spark connector



MongoDB server

- The server data can be:
 - **Replicated**
 - Several identical copies of the same data
 - **Partitioned** (sharded)
 - Distributed across the machines of a cluster in order to take advantage of the storage and processing capacity
- Document processing
 - Selective access, or
 - Aggregation Pipelines: Map-Reduce mode (deprecated now)

MongoDB storage organization

- **Documents** are stored in **collections** (which may have **indexes**)
 - Collections are part of **databases**
- ```
> use myExample // Creates the database myExample
> db.towns.insertOne({ // Creates the collection towns
 name: "New York", // and inserts a document into it
 population: 22200000,
 last_census: ISODate("2009-07-31"),
 famous_for: ["statue of liberty", "food"],
 mayor : {
 name : "Michael Bloomberg",
 party : "I"
 }
})
```

# Replica sets

**Duplication (replication) to prevent against server failure and data loss**

Example (three servers): mkdir ./mongo1 ./mongo2 ./mongo3

Create replication set:

```
mongod --replSet book --dbpath ./mongo1 --port 27011
```

```
mongod --replSet book --dbpath ./mongo2 --port 27012
```

```
mongod --replSet book --dbpath ./mongo3 --port 27013
```

Connect to each set:

```
mongosh localhost:27011
```

Then in one of the servers, initialize replication set:

```
> rs.initiate({_id: 'book',
 members: [{_id: 1, host: 'localhost:27011'},
 {_id: 2, host: 'localhost:27012'},
 {_id: 3, host: 'localhost:27013'}] })
```

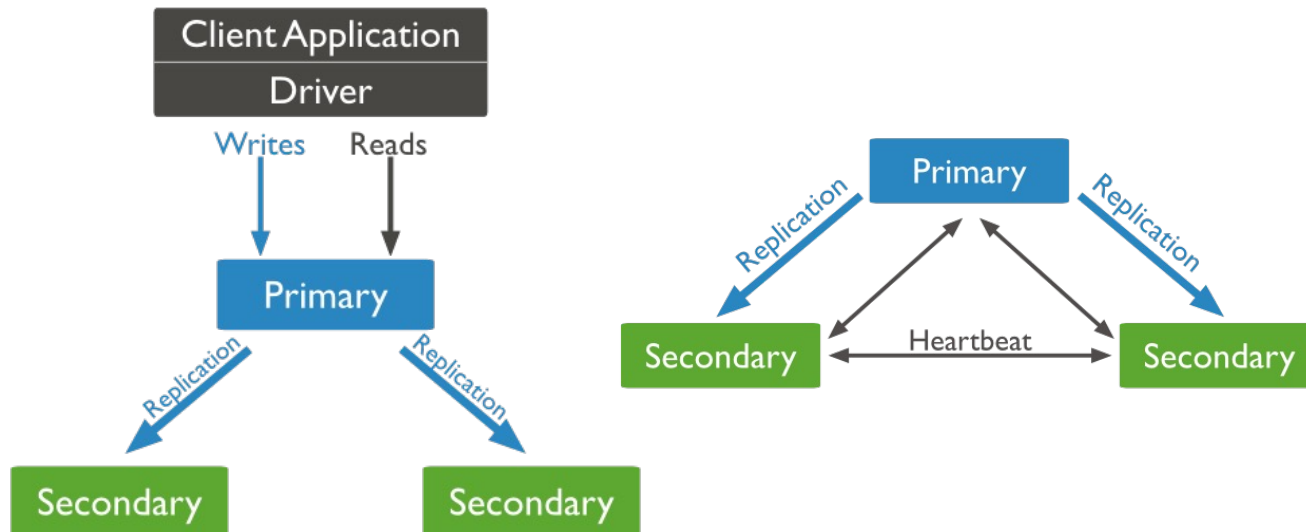
Then one server will output **PRIMARY**

while two will output **SECONDARY**

# Replica sets in MongoDB

The servers held a vote to determine who is the master (primary); the two others are replicas ("secondary")

By default, applications *read/write through the primary*, who pushes updates to secondary servers

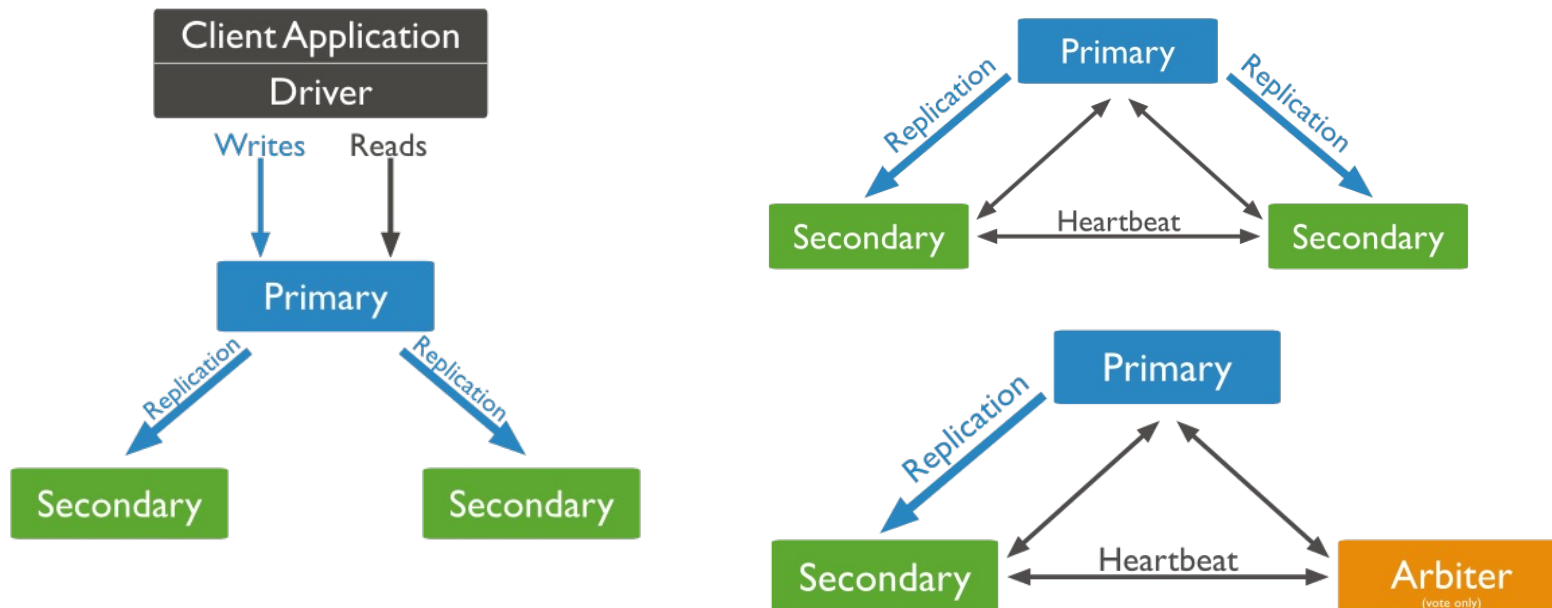


If the master is considered dead, there are new elections

- *Only succeed if more than half of the original replication set votes*
- Operations attempted on a demoted (dead) master are lost
- *A write is considered successful only if > half of the replicas "saw" it*

# Replica sets

By default, write to the primary, which pushes updates to secondary servers



MongoDB recommends an *odd number of servers in a replica set*, to allow a majority in case of network failure. **One arbiter** may be added to the replica set

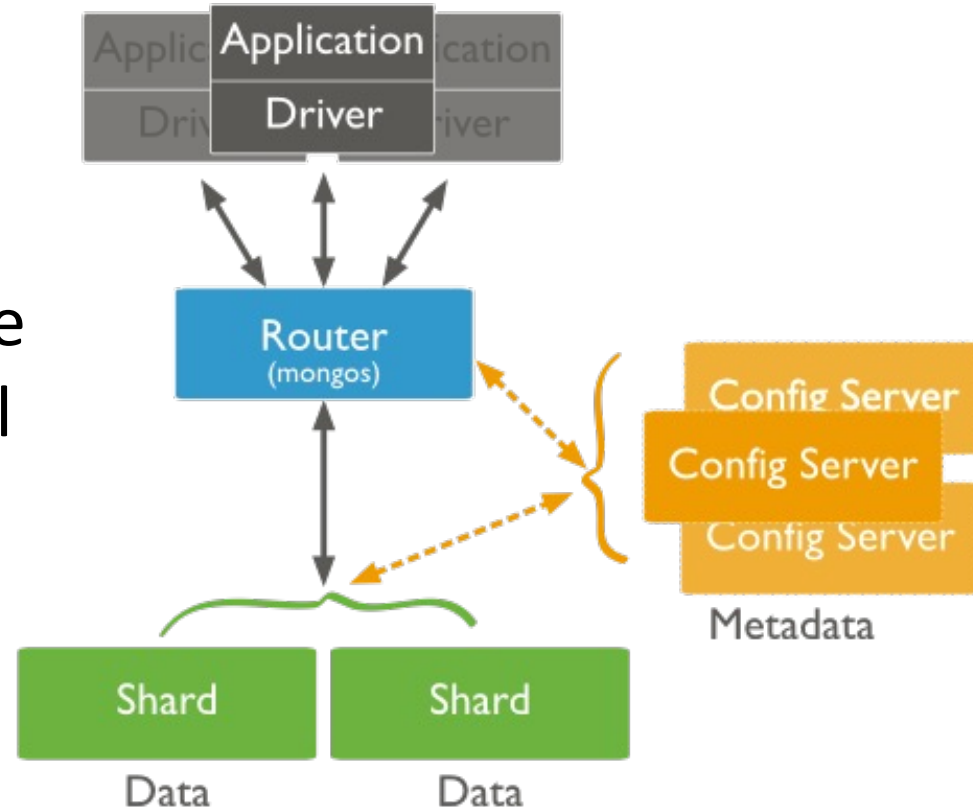
- Strong consistency on read
- Resistance to some partitioning

# MongoDB sharding

*Sharding* = partitioning

1 shard = 1 fragment

- To distribute a very large collection across several servers



# MongoDB sharding

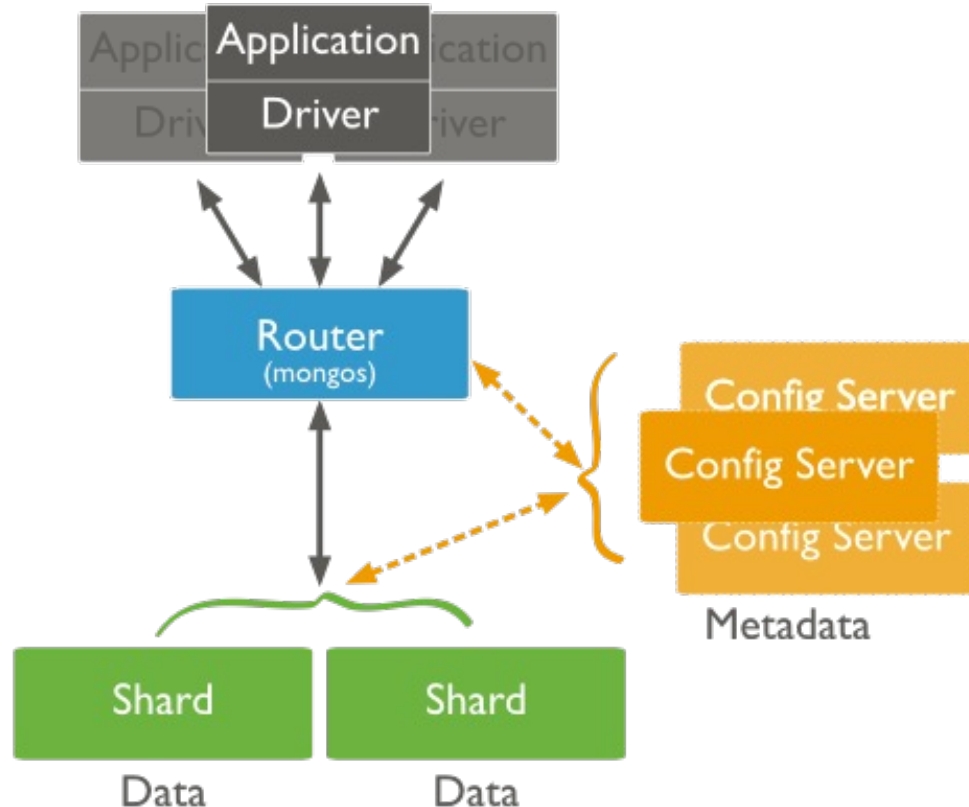
*Sharding* = partitioning

1 shard = 1 fragment

- To distribute a very large collection across several servers

Sharding is logically *on top of replication*

- Each shard server may participate to a replica set

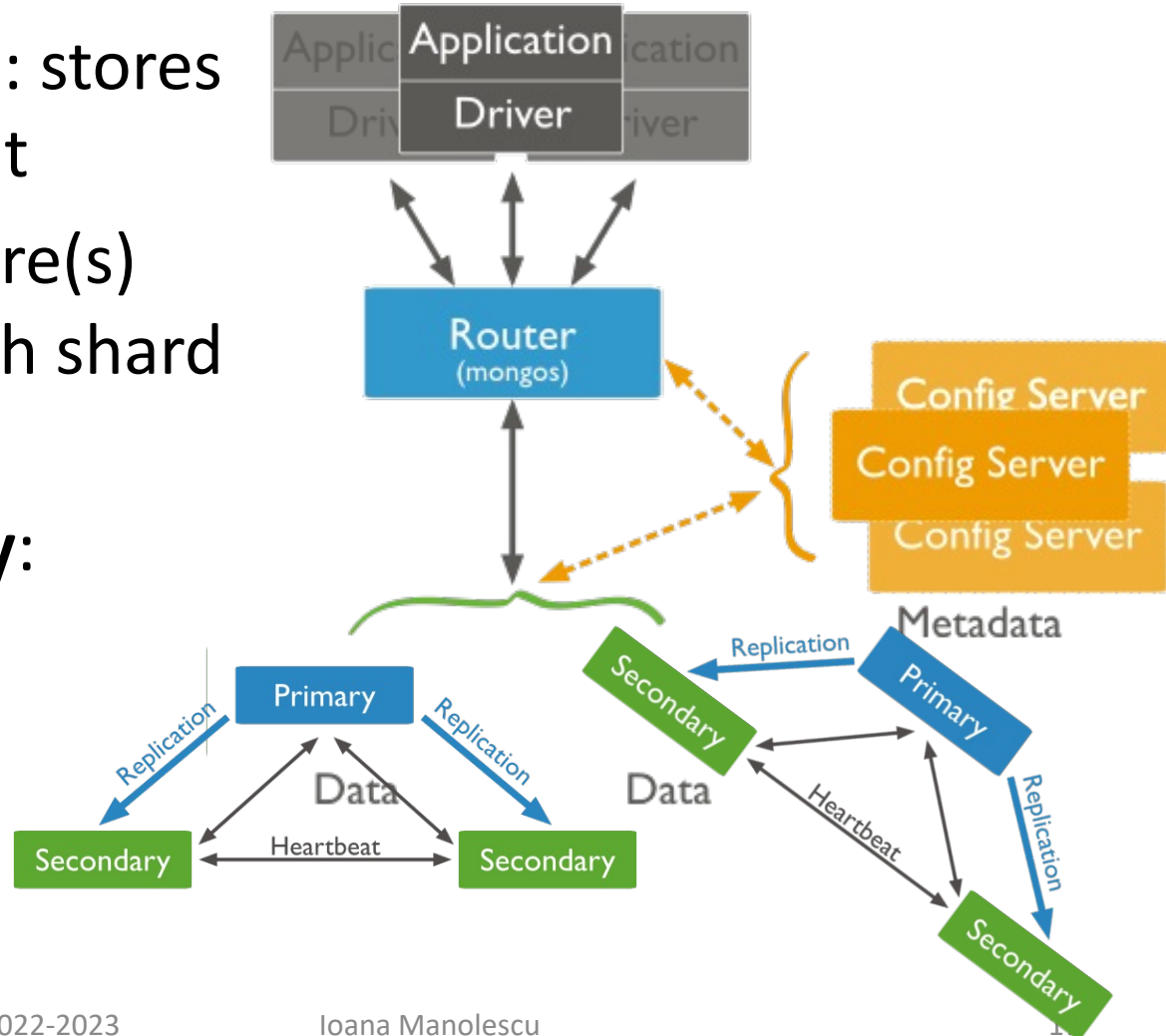




# MongoDB sharding

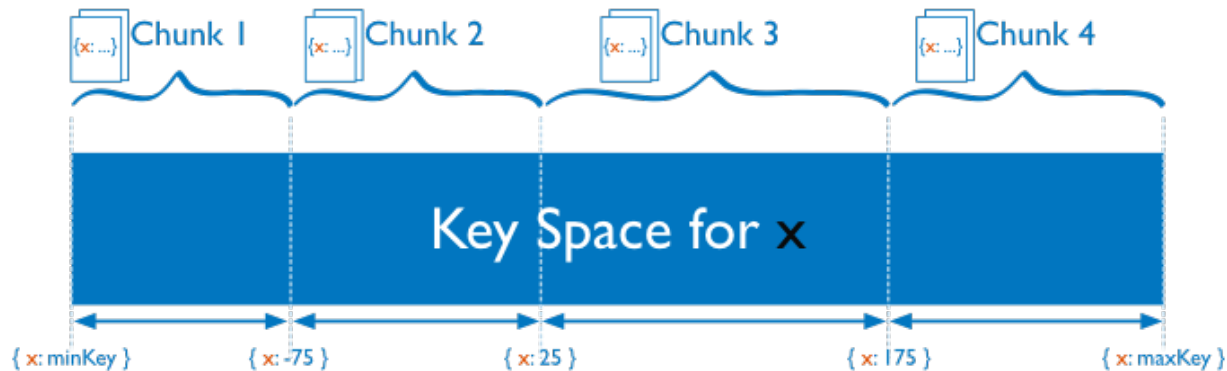
## Roles:

- **Shard** (shard server): stores a collection fragment
- **Config server(s)**: store(s) information on which shard has what (~ catalog)
- **Single point of entry:** mongos



# MongoDB sharding

- A data collection is partitioned into **chunks** based on the value of a **shard key**
- Each chunk covers a key range



- 1 shard = a set of chunks
- mongos routes writes to the appropriate chunk based on the shard key value
- Chunks are split when they grow beyond a fixed chunk size
- MongoDB migrates chunks across shards for load balancing

# Conclusion: JSON stores

- **JSON:** extremely popular data interchange format
- **MongoDB:** (by far) most popular JSON data management system
- **MongoDB query language:**
  - Rich matching within one document (declarative)
  - Pipeline processing for more complex operations (non-declarative; see the documentation)
- **MongoDB replication:**
  - replica group, voting, quorum
- **MongoDB distribution:** sharding
  
- Other JSON stores:
  - Amazon DynamoDB (also serves as K-V store)
  - CouchDB

More document stores: [http://db-engines.com/en/ranking\\_trend/document+store](http://db-engines.com/en/ranking_trend/document+store)