Interacting With MongoDB

- Various APIs
  - Python, Ruby, Perl, Java, Java, Scala…
- MongoDB uses natively javascript
  - Simple language and easy to use
  - No datatypes
  - You can write Javascript in the shell or load a file to run
- UI: a lot off options
  - mongo-express, Edda, HumongouS, Umongo, MongoVision

Starting up MondoDB

- In the command line : **mongod**
- You need to have folder for the database:
  - Linux \data\db
  - Windows C:\data\db
  - You can set an alternative path in the command line:
    mongod --dbpath <full path to folder>
- Also start the ipython notebook

Example 1: People

```
mongoimport --host localhost --port 27017 --db forum --collection
profiles  --file ./profiles.json --jsonArray
```

```
{
        "_id" : ObjectId("5114e0bd42…"),
        "first" : "John",
        "last" : "Doe",
        "age" : 39,
       "interests" : [
               "Reading",
               "Mountain Biking ]
       "favorites": {
              "color": "Blue",
              "sport": "Soccer"}
}
```

Pymongo

- MongoDB commands inside python

```
from pymongo import MongoClient
client = MongoClient()
db=client.forum
```

If the database/collection does not exist it gets created at run time

```
res=db.profiles.find().limit(10)
for r in res:
    favorites=""
    if "favorites" in r:
        favorites=str(r['favorites'])
    print r['name']+"\t"+r['lastname']+"\t"+favorites
```

Simple Queries

```
db.profiles.find({'name':'Billy'})
#OR FORMALLY
db.profiles.find({'name':{'$eq':'Billy'}})
```

```
from datetime import datetime
before = datetime(1990, 1, 30, 0, 0, 0)
db.profiles.find({'name':{'$eq':'Billy'}, 'dob':{‘$lt':before}})
```

Selection and Projection

- db.collection.find(**query, projection**)
  - Query and projection optional

```
db.profiles.find({'name':{'$eq':'John'}},{'favorites':1})
```

```
db.profiles.find({'name':{'$eq':'John'}},{'favorites':0})
```

```
db.profiles.find({},{'_id':0})
```

```
db.profiles.find({'$and':[{'name':{'$eq':'John'}},{'favorites.color':'Black'}]})
```

```
db.profiles.find({'$and':[{'name':{'$eq':'John'}},{'favorites.color':{'$ne':None}}
]},{'favorites':1})
```

Sort and Count

```
import pymongo

db.profiles.find({'name':{'$eq':'John'}},{'name':1,'lastname':1}).sort('lastname', pymongo.ASCENDING)
```

```
db.profiles.find({name:{$eq:'John'}}).count()
```

Insert

```
db.profiles.insert({'name':'Rick','lastname':'Sanchez'})
db.profiles.insert([{'name':'Morty','lastname':'Smith'},
{'name':'Summer','lastname':'Smith'}])
```

- You can insert stuff with completely different fields
  - A completely inconsistent “schema” would make the database management very hard

Arrays in MongoDB (1)

```
#We can set an array
db.profiles.update({'name':"Rick",'lastname':"Sanchez"},{'$set':{'interests'
:['booze','science']}})
#if the document is not found: upsert=True will create it
#if we want to update multiple docs: multi=True
#add new stuff
db.profiles.update({'name':"Rick",'lastname':"Sanchez"},
{'$push':{'interests':'money'}})
```

```
db.profiles.update({'name':"Rick",'lastname':"Sanchez"},
{'$push':{'interests':['unity','rebelion']}})

db.profiles.update({'name':"Rick",'lastname':"Sanchez"},
{'$pushAll':{'interests':['unity','rebelion']}})
```

Arrays in MongoDB (2)

```
#We can query on the values of an array
#e.g. How many people are interested in Reading ?

db.profiles.find({'interests':'Reading'}).count()

#We can also search based on the length of the array
#e.g. How many are interested in Reading  AND have 2 or more interests?
#we need some javascript
db.profiles.find({'interests':'Reading', '$where':"this.interests.length>=2"}).count()

#How many are interested in Reading  OR Walking?
db.profiles.find({'interests':{'$in':['Reading','Walking']}}).count()

#How many are interested in Reading  AND Walking?
db.profiles.find({'interests':{'$all':['Reading','Walking']}}).count()
```

Scenario

- Data dump from stackexchange for the following category of cooking
- File: *data.csv*
  - Comma separated fields:

post id	Post title	keywords	Post type: Question / Answer	text

```
data=pandas.read_csv('data.csv')
rows=data.to_dict(orient='records')
for row in rows:
    row['keywords']=row['keywords'].split(',')
res=db.cooking.insert(rows)
```

Text Search (1)

- Find posts where the question has the word **italian**

```
db.cooking.create_index( [("text" ,pymongo.TEXT)])
```

- We need a special operator to access this index:

```
db.cooking.find({'$text':{'$search': 'italian'}})
```

- The search uses the root of the word
- Many terms inside the string are combined with OR

Text Search (2)

- We can have exact search with terms quoted
- **Italian as is OR burn**

```
>db.index.find({$text:{$search:'\“italian\” burning'}})
```

- More than one quoted terms are combined with AND
- We can create one text type index per collection
  - But it can contain multiple fields
- If we want to use it in an aggregation : the text match has to be first in the pipeline

Aggregation in MongoDB (1)

- db.collection.aggregate([{ <stage> }, ...])

```
db.profiles.aggregate([{'$project':
                       {'age':
                         {'$divide':[
                            {'$subtract':[datetime.now(),'$dob']},
                            31558464000
                            ]
                         },
                       'name':1}
                       },
                     {'$group':{
                                     '_id':'$name',
                                      'avgAge':{'$avg':'$age'}
                            }
                      },
                     {'$match':{'avgAge':{'$gt':18}
                     }}])
```

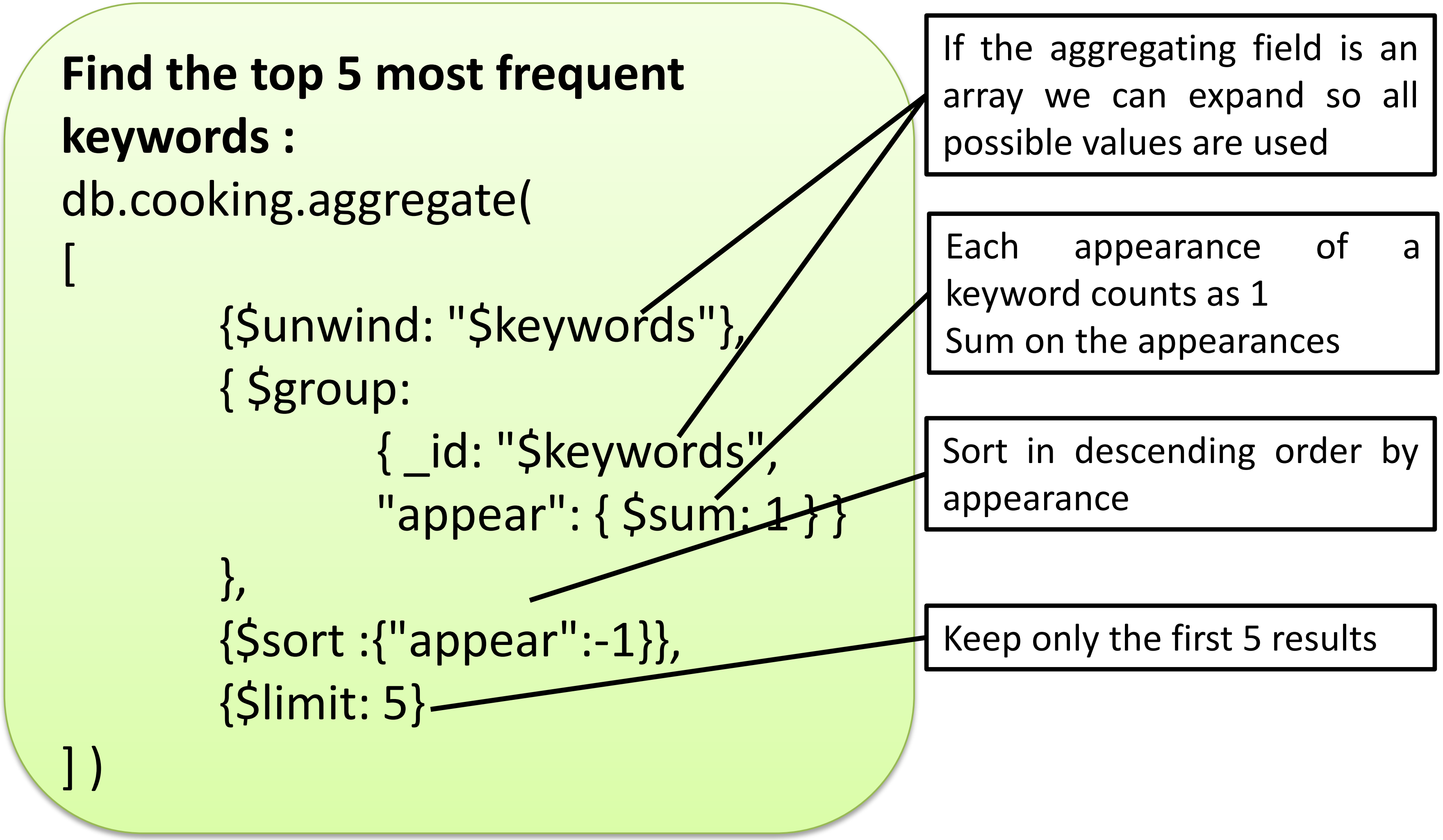
Accumulators

$sum	Returns a sum for each group. Ignores non-numeric values.
$avg	Returns an average for each group. Ignores non-numeric values.
$first	Returns a value from the first document for each group.
$last	Returns a value from the last document for each group.
$max	Returns the highest expression value for each group.
$min	Returns the lowest expression value for each group.
$push	Returns an array of expression values for each group.
$addToSet	Returns an array of unique expression values for each group. Order of the array elements is undefined.

Operator (reminder)

Name	Description
$project	Manage the fields you want to use
$match	Apply a query to filter the data
$limit	Use only the first n documents
$skip	Skip n documents
$unwind	Applied in a array which flattens it .
$group	Groups input documents by a specified identifier expression and applies accumulator expression(s),.
$sort	Reorders the document stream by a specified sort key.
$out	Writes the resulting documents of the aggregation pipeline to a collection.
$split	Splits string into array
$size	During aggregation returns length of array

Aggregation in MongoDB (2)



Find the top 5 most frequent keywords :

```
db.cooking.aggregate(
[
        {$unwind: "$keywords"},
        { $group:
                { _id: "$keywords",
                "appear": { $sum: 1 } }
        },
        {$sort :{"appear":-1}},
        {$limit: 5}
] )
```

If the aggregating field is an array we can expand so all possible values are used

Each appearance of a keyword counts as 1
Sum on the appearances

Sort in descending order by appearance

Keep only the first 5 results

Keyword frequency distribution

```
#Put the pipeline commands of an aggregation as
#dictionaries in a list
pipeline=[{"$project":{"keywords":1}},
          {"$unwind":"$keywords"},
          {"$group":{
              "_id":"$keywords",
               "count":{"$sum":1}}
          }]
```

- You can put the result in pandas dataframe

Exercises

1. Find the top-5 music genres for people who like "Listening to Music“
2. For each favorite sport find the top interest
3. The cooking data are not stored optimally. Re-load the data (in a new collection) by grouping answers
   – Find the most answered keywords in both versions
4. Find the top 5 most frequent keywords where the question has the word italian
5. Find the count frequency of every word in the collection per keyword.
6. Find the top 30 most frequent words

THANK YOU