# Big Data e Business Intelligence

Database
e gestione dei dati

Giulio Angiani - UniPr

# Sistema informativo

# Database NoSQL

- NoSQL
  - Not Only SQL
  - Paradigma non relazionale
- Varie tipologie
  - DB Document-oriented
    - XML databases
  - Graph databases
  - DB Key-Value
  - Wide-column stores
- Esempi DB NoSQL
  - MongoDB, Cassandra, HBase, Neo4j

# Database NoSQL

## Caratteristiche:


- Flessibilità
- Scalabilità
- Elevate prestazioni
- Funzionalità alta

## Vantaggi:

- analisi di dati semi-strutturati
- bassa latenza
- multi-istanza
- quando abbiamo dati eterogenei e senza uno schema

# MongoDB

- database orientato ai documenti
- pensare in JSON
- documenti raggruppati in collezioni (anche non omogenei)
- NB: **non esiste un concetto analogo al vincolo di integrità referenziale**

# Usiamo MongoDB

## lanciamo l'interprete a riga di comando

SHELL

```
$ mongo
MongoDB shell version v4.2.0
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("d6382e28-ca58-41af-b237-a75e52c6993e") }
MongoDB server version: 4.2.0
>
```

MONGO

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
unipr   0.000GB
> use unipr
switched to db unipr
```

# Usiamo MongoDB : Inserimento

## Due metodi

- db.collection.insertOne()
- db.collection.insertMany()
  (la collection viene creata automaticamente se non esiste. No Create!)

MONGO

```
> db.teachers.insertOne({
    "cognome": "Angiani",
    "nome": "Giulio",
    "eta": 46,
    "email": "giulio.angiani@unipr.it",
    "ruolo": "Docente esterno"
})
{
    "acknowledged" : true,
    "insertedId" : ObjectId("5d612377e503102a50f614ea")
}
```

ref: https://docs.mongodb.com/manual/

# Usiamo MongoDB : Inserimento

MONGO

```
> db.teachers.insertMany([
    { "cognome": "Mordonini", "nome": "Monica", "email": "monica.mordonini@unipr.it",
      "ufficio": { "palazzina" : 1,
                   "stanza": 102,
                   "piano": 1,
                   "telefono": "+39 0521 905735"
                 },
      "ruolo": "Ricercatrice"
    },
    { "cognome": "Tomaiuolo", "nome": "Michele", "age": 43,
      "ruolo": "Ricercatore", "email": "michele.tomaiuolo@unipr.it",
      "ufficio" : { "telefono": "+39 0521 905708" }
    }
])
```

ref: https://docs.mongodb.com/manual/

# Usiamo MongoDB : Ricerca

- senza filtro: come *select * from teachers*

MONGO

```
> show collections
teachers
> db.teachers.find()
{ "_id" : ObjectId("5d612377e503102a50f614ea"), "familyname" : "Angiani", "name" : "Giulio",
"age" : 46, "email" : "giulio.angiani@unipr.it" }
{ "_id" : ObjectId("5d6126d3e503102a50f614eb"), "cognome" : "Mordonini", "nome" : "Monica",
"email" : "monica.mordonini@unipr.it", "ufficio" : { "palazzina" : 1, "stanza" : 102,
"piano" : 1, "telefono" : "+39 0521 905735" }, "ruolo" : "Ricercatrice" }
{ "_id" : ObjectId("5d6126d3e503102a50f614ec"), "cognome" : "Tomaiuolo", "nome" : "Michele",
"age" : 43, "email" : "michele.tomaiuolo@unipr.it", "ruolo" : "Ricercatore",
"ufficio" : { "telefono" : "+39 0521 905708" } }
```

ref: https://docs.mongodb.com/manual/

# Usiamo MongoDB : Ricerca

- con filtro: come *select * from teachers where name = 'Giulio'*

MONGO

```
> show collections
teachers
> db.teachers.find({"name": "Giulio"})
{ "_id" : ObjectId("5d612377e503102a50f614ea"), "familyname" : "Angiani", "name" : "Giulio",
"age" : 46, "email" : "giulio.angiani@unipr.it" }

> db.teachers.find({"name": "Giulio"}).pretty()
{
    "_id" : ObjectId("5d612377e503102a50f614ea"),
    "familyname" : "Angiani",
    "name" : "Giulio",
    "age" : 46,
    "email" : "giulio.angiani@unipr.it"
}
```

ref: https://docs.mongodb.com/manual/

# Usiamo MongoDB : Ricerca

- filtro annidato (in SQL solo con join o query nidificate)

MONGO

```
> db.teachers.find({"ufficio.palazzina": 1 }).pretty()
{
    "_id" : ObjectId("5d6126d3e503102a50f614eb"),
    "cognome" : "Mordonini",
    "nome" : "Monica",
    "email" : "monica.mordonini@unipr.it",
    "ufficio" : {
        "palazzina" : 1,
        "stanza" : 102,
        "piano" : 1,
        "telefono" : "+39 0521 905735"
    },
    "ruolo" : "Ricercatrice"
}
```

ref: https://docs.mongodb.com/manual/

# Usiamo MongoDB : Ricerca

- filtro con regexp (in SQL si userebbe like '%9057%')

MONGO

```
> db.teachers.find({"ufficio.telefono": /9057/ })
{ "_id" : ObjectId("5d6126d3e503102a50f614eb"), "cognome" : "Mordonini", "nome" : "Monica",
"email" : "monica.mordonini@unipr.it", "ufficio" : { "palazzina" : 1, "stanza" : 102,
"piano" : 1, "telefono" : "+39 0521 905735" }, "ruolo" : "Ricercatrice" }
{ "_id" : ObjectId("5d6126d3e503102a50f614ec"), "cognome" : "Tomaiuolo", "nome" : "Michele",
"age" : 43, "email" : "michele.tomaiuolo@unipr.it", "ruolo" : "Ricercatore",
"ufficio" : { "telefono" : "+39 0521 905708" } }
```

ref: https://docs.mongodb.com/manual/

# Usiamo MongoDB : Ricerca

- filtro con più clausole

MONGO

```
> db.teachers.find({"ruolo": /^Ricerc/ ,  age : {$lt: 45}}).pretty()
> db.teachers.find({$and : [{"ruolo": /^Ricerc/ },  {age : {$lt: 45}}]} ).pretty()
{
    "_id" : ObjectId("5d6126d3e503102a50f614ec"),
    "cognome" : "Tomaiuolo",
    "nome" : "Michele",
    "age" : 43,
    "email" : "michele.tomaiuolo@unipr.it",
    "ruolo" : "Ricercatore",
    "ufficio" : {
        "telefono" : "+39 0521 905708"
    }
}
```

ref: https://docs.mongodb.com/manual/

# Usiamo MongoDB : Ricerca

- selezione campi principali e annidati

MONGO

```
> db.teachers.find( {$and : [{"ruolo": /^Ricerc/ },  {age : {$lt: 45}}]} ,
                    {"nome" : 1, "cognome" : 1, "age": 1, _id: 0}
                  ).pretty()
{ "cognome" : "Tomaiuolo", "nome" : "Michele", "age" : 43 }

> db.teachers.find( {$and : [{"ruolo": /^Ricerc/ },  {age : {$lt: 45}}]} ,
                    {"nome" : 1, "cognome" : 1, "age": 1, _id: 0,
                     "ufficio.telefono" : 1} ).pretty()
{
    "cognome" : "Tomaiuolo",
    "nome" : "Michele",
    "age" : 43,
    "ufficio" : {
        "telefono" : "+39 0521 905708"
    }
}
```

ref: https://docs.mongodb.com/manual/

# Usiamo MongoDB : Ricerca

- selezione campi non esistenti (principali e annidati)

MONGO

```
> db.teachers.find( {$and : [{"ruolo": /^Ricerc/ },  {age : {$lt: 45}}]} ,
                    {"nome" : 1, "cognome" : 1, "age": 1, _id: 0, altezza: 1,
                     "ufficio.palazzina" : 1} ).pretty()
{
    "cognome" : "Tomaiuolo",
    "nome" : "Michele",
    "age" : 43,
    "ufficio" : {}
}
```

ref: https://docs.mongodb.com/manual/

# Usiamo MongoDB : Update

Tre metodi

- db.collection.updateOne(<filter>, <update>, <options>)
- db.collection.updateMany(<filter>, <update>, <options>)
- db.collection.replaceOne(<filter>, <update>, <options>)

ref: https://docs.mongodb.com/manual/

# Usiamo MongoDB : Update

## Esempio

- in SQL sarebbe "update teachers set age=47 where cognome = 'Angiani'"

MONGO

```
> db.teachers.updateOne(
   { cognome: "Angiani" },
   { $set: { "eta": "47" } }
)
```

ref: https://docs.mongodb.com/manual/

# Usiamo MongoDB : Update

MONGO

```
> db.teachers.find({"cognome": "Angiani"}).pretty()
{
    "_id" : ObjectId("5d7797b0b47376d19599cba4"),
    "cognome" : "Angiani",
    "nome" : "Giulio",
    "eta" : 47,
    "email" : "giulio.angiani@unipr.it",
    "ruolo" : "Docente esterno"
}
```

ref: https://docs.mongodb.com/manual/

# Python e MongoDB

## Installazione:

SHELL

```
$ sudo pip3 install pymongo
# Collecting pymongo
# Downloading https://..../pymongo-3.9.0-cp36-cp36m-manylinux1_x86_64.whl (446kB)
# 100% |████████████████████████████████| 450kB 2.0MB/s
# Installing collected packages: pymongo
# Successfully installed pymongo-3.9.0...
```

# Python e MongoDB

## Accesso al database

PYTHON

```python
from pymongo import MongoClient
import pprint
client = MongoClient()
db = client.unipr
collection = db.teachers # or collection = db["teachers"]
```

ref: https://api.mongodb.com/python/current/

# Python e MongoDB

## Query

PYTHON

```
# primo elemento trovato
t = collection.find_one()
pprint.pprint(t)
```

OUTPUT

```
{u'_id': ObjectId('5d779a009a5f39e0ea5c1bbf'),
 u'cognome': u'Angiani',
 u'email': u'giulio.angiani@unipr.it',
 u'eta': u'47',
 u'nome': u'Giulio',
 u'ruolo': u'Docente esterno'}
```

ref: https://api.mongodb.com/python/current/

# Python e MongoDB

## Query

PYTHON

```
# filtro
t = collection.find_one({"cognome": "Tomaiuolo"})
pprint.pprint(t)
```

OUTPUT

```
{u'_id': ObjectId('5d779a009a5f39e0ea5c1bc1'),
 u'age': 43.0,
 u'cognome': u'Tomaiuolo',
 u'email': u'michele.tomaiuolo@unipr.it',
 u'nome': u'Michele',
 u'ruolo': u'Ricercatore',
 u'ufficio': {u'telefono': u'+39 0521 905708'}}
```

ref: https://api.mongodb.com/python/current/

# Python e MongoDB

## Query

PYTHON

```python
res = collection.find() # resituisce un oggetto pymongo.cursor.Cursor; Va iterato...
for t in res:
    print(t.get('cognome'), t.get('nome'), t.get('email'), t.get("eta"))
```

OUTPUT

```
(u'Angiani', u'Giulio', u'giulio.angiani@unipr.it', u'47')
(u'Tomaiuolo', u'Michele', u'michele.tomaiuolo@unipr.it', 43.0)
(u'Mordonini', u'Monica', u'monica.mordonini@unipr.it', None)
```

ref: https://api.mongodb.com/python/current/

# Python e MongoDB

## Query con filtro e ordinamento

PYTHON

```python
res = collection.find().sort("nome", 1)  # nota: 1 asc [default], -1 desc
for t in res:
    print(t.get('cognome'), t.get('nome'), t.get('email'), t.get("eta"))
```

OUTPUT

```
(u'Angiani', u'Giulio', u'giulio.angiani@unipr.it', u'47')
(u'Tomaiuolo', u'Michele', u'michele.tomaiuolo@unipr.it', 43.0)
(u'Mordonini', u'Monica', u'monica.mordonini@unipr.it', None)
```

ref: https://api.mongodb.com/python/current/

# Python e MongoDB

## Inserimento

PYTHON

```python
doc = {
    "cognome": "Cagnoni",
    "nome": "Stefano",
    "eta" : 58,
    "ruolo": "Professore Associato",
    "email": "stefano.cagnoni@unipr.it"
}
# inserisce
id = collection.insert_one(doc)
# inserisce e restituisce l'id dell'elemento inserito
id = collection.insert_one(doc).inserted_id
```

ref: https://api.mongodb.com/python/current/

# Python e MongoDB

## Query con filtro e ordinamento

PYTHON

```python
res = collection.find().sort("nome", -1)  # nota: 1 asc [default], -1 desc
for t in res:
    print(t.get('cognome'), t.get('nome'), t.get('email'), t.get("eta"))
```

OUTPUT

```
(u'Cagnoni', u'Stefano', u'stefano.cagnoni@unipr.it', 58)
(u'Mordonini', u'Monica', u'monica.mordonini@unipr.it', None)
(u'Tomaiuolo', u'Michele', u'michele.tomaiuolo@unipr.it', 43.0)
(u'Angiani', u'Giulio', u'giulio.angiani@unipr.it', u'47')
```

ref: https://api.mongodb.com/python/current/

# Applicazioni

# Applicazione pratica

- Molti servizi utilizzano dati in formato JSON
- Formato dati spesso diverso
- Dati non normalizzati

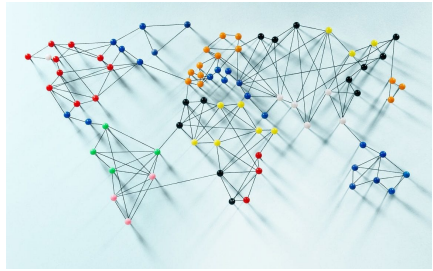
E' sufficiente che siano in formato JSON

- Nessuna trasformazione anche se provenienti sorgenti diverse

# Applicazione pratica - Twitter

## Ricostruire relazioni su Twitter

- Nodi => Utenti/Tweet
- Archi => Relazione di following/Retweeting



## esempio tweet:

JSON

```
{u'contributors': None,
 u'coordinates': None,
 u'created_at': u'Sun Sep 15 16:29:55 +0000 2019',
 u'entities': {u'hashtags': [{u'indices': [20, 33],
                              u'text': u'PresidentPAB'}],
               u'symbols': [],
               u'urls': [],
               u'user_mentions': [{u'id': 462288600,
                                   [...]
```

vedi

# Applicazione pratica - Twitter

- Libreria per python-Twitter (Tweepy)

PYTHON

```python
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy import API
from tweepy import Cursor
from tweepy.error import RateLimitError, TweepError
```

- per info... https://www.tweepy.org
- altre: https://developer.twitter.com/en/docs/developer-utilities/twitter-libraries

# Applicazione pratica - Twitter

- Dati recuperati direttamente in formato JSON

PYTHON

```
d = {
    "consumer_key": "iIkJ52sadasNR2ZX1RWq",
    "consumer_secret": "YZ9ti7sdsqQFu4EfdfaasB6HaEjYDpBhNr6PdKMITfMs5",
    "access_token": "22516223340-xcQz8BuA7XdLyfgfasdsfa334N8D4BwrfDHUjFTWQ",
    "access_token_secret": "4B0XuPZtpMhdfsd455SUqjeUawI2h49ihl27GUTHum8ON2"
}

auth = OAuthHandler(d["consumer_key"], d['consumer_secret'])
auth.set_access_token(d['access_token'], d['access_token_secret'])
api = API(auth)
tweets = api.search(q="Dorian", count=100)
```

# Applicazione pratica - Twitter

- ed inseriti in DB Mongo

PYTHON

```python
for t in tweets:
    #pprint.pprint(t._json)
    #print(t.id)
    if not db.tweets.find_one({"id_str": str(t.id)}):
        print("insert ", t.id)
        db.tweets.insert_one(t._json)
    else:
        print("presente ", t.id)
    #print(dir(t))
    u = t.user
    #pprint.pprint(u._json)
    # se non e' presente lo inserisco nella collection
    if not db.users.find_one({"screen_name" : u.screen_name}):
        db.users.insert_one(u._json)
```

# Esempi di operazioni: tutti gli users

PYTHON

```python
from pymongo import MongoClient, ASCENDING, DESCENDING
import pprint
client = MongoClient()
db = client.unipr
c_users = db.users
c_tweets = db.tweets
# tutti gli utenti
users = c_users.find({}, {"screen_name": 1}).sort([("screen_name", ASCENDING)])
for u in users:
    print(u["screen_name"])
```

OUTPUT

```
Alfredo88461728
AllCruiseNews
BertoDryden
BradfordCotton5
BurrellMinistry
CloseToHomeInc
...
```

# Esempi di operazioni: tweets di un utente

PYTHON

```
pipeline = [
            {"$group": {"_id": "$user.screen_name", "count": {"$sum": 1}}},
            {"$sort": SON([("count", -1), ("_id", -1)])}
        ]

lista = list(db.tweets.aggregate(pipeline))
print("Primi 3 elementi: ")
for elem in lista[0:3]:
    print elem['_id'], elem['count']

# screen_name del piu' attivo
screen_name = lista[0]["_id"]
print("\npiù attivo: " + screen_name + "\n")
# tutti i tweet dell'utente piu' attivo
tws = c_tweets.find({"user.screen_name": screen_name})
for t in tws:
    print(t["lang"] + " : " +  t["text"])
```

# Esempi di operazioni: tweets di un utente

OUTPUT

```
Primi 3 elementi:
Dorian_Gray_L 6
BertoDryden 4
fvl_dorian_ 2

più attivo: Dorian_Gray_L

ko : @Hedwig_ch 그렇죠?(생긋)
ko : @Hedwig_ch 글쎄요. 어느곳에서든 그럴수 있죠. 그게 그다지 중요한게 아니듯이요.
ko : @Hedwig_ch 런던이 늘 그렇죠, 뭐. (천하태평)
ko : @Long_Grift_ (한번 콕 찔러본다)
ko : @Lucy_H_JH 나도요.
ko : @Hedwig_ch 시간이 뒤틀리는 것 정도야 그렇게 드문 일도 아니지 않나요?
```

# Esempi di operazioni: tweets di un utente in inglese

PYTHON

```
pipeline = [
        {"$match": {"lang": "en"} },        # <<<<< IN TESTA ALLA PIPELINE
        {"$group": {"_id": "$user.screen_name", "count": {"$sum": 1}}},
        {"$sort": SON([("count", -1), ("_id", -1)])}
    ]

lista = list(db.tweets.aggregate(pipeline))
# screen_name del piu' attivo
screen_name = lista[0]["_id"]
print("\npiù attivo: " + screen_name + "\n")
# tutti i tweet dell'utente piu' attivo
tws = c_tweets.find({"user.screen_name": screen_name})
for t in tws:
    print(t["lang"] + " : " +  t["text"])
```

# Esempi di operazioni: tweets di un utente in inglese

OUTPUT

```
Primi 3 elementi:
BertoDryden 4
ResilienceAcad 2
CulverCityNIMBY 2

più attivo: BertoDryden

en : RT @TwitterMoments: The Hurricane Dorian death toll has risen to 23 in the Bahamas,[...]
en : RT @nytimesworld: Tropical Storm Humberto is unlikely to be as destructive [...]
en : RT @foxnewsradio: Tropical Storm Humberto  bringing rain to parts of the Bahamas [...]
en : RT @JacobLanierWx: @HUMBERTO...TD Nine has made it to the minor leagues...it's now [...]
```

# Esempi di join in shell mongo

MONGO

```
db.users.aggregate([
   {
      $lookup:
         {
            from: "tweets",
            localField: "screen_name",
            foreignField: "user.screen_name",
            as: "user_tweets"
         }
   }
])
```

- per python *mongojoin*

# Esempi di join in shell mongo

- join + limit + proiezione su entrambe le collections

MONGO

```
db.users.aggregate([
   {
      $lookup:
         {
            from: "tweets",
            localField: "screen_name",
            foreignField: "user.screen_name",
            as: "user_tweets"
          },

   },
   {$limit: 3},
   { $project : { screen_name: 1, "user_tweets.text": 1} }
])
```

# Esercitazione

- scegliamo tre #hashtag e un paio di @utenti da seguire
- scarichiamo 100 tweets per ogni tipologia (con lingue diverse)
- strutturiamo il DB con Mongo
- eseguiamo le seguenti queries
  - i testi dei tweet di un certo utente ordinati per data di pubblicazione
  - testi di tutti i tweet in lingua inglese
  - gli utenti che hanno pubblicato in più lingue

Giulio Angiani
Universita' degli Studi di Parma