# Práctica Bases de Datos no Convencionales

Rubén Rodríguez
Juan Carlos Prieto

# Procesamiento de datos

Para el procesamiento del fichero dblp hemos creado dos scripts en python, “dblp_to_mongoimport.py” y “dblp_to_neo4jimport.py” que transforman el fichero dblp.xml a un fichero adecuado para mongo o neo4j.

Dado el tamaño del fichero de entrada, los datos se van leyendo del archivo según se van necesitando, para evitar quedarnos sin memoria, y posteriormente se escriben en su correspondiente fichero.

El script converter.py  es el encargado de leer el XML y generador un stream de diccionarios (convierte un elemento XML a un diccionario), y es usado por los scripts “dblp_to_mongoimport.py” y “dblp_to_neo4jimport.py”.

# Mongo

## Crear contenedor mongo

Para facilitar la reproducción de los resultados mostrados en el presente trabajo, hemos optado por trabajar con contenedores de docker.

Por lo que primero, debemos crearnos un nuevo contenedor para la práctica, el nombre (--name), puede ser modificado, así como las rutas locales de los volúmenes.

```
sudo docker run --name mongo -p 27017:27017 -p 28017:28017 -v
/home/runix/Projects/docker/mongo/data:/data/db -v
/home/runix/Projects/other/xml2json/output:/data/seeds -d mongo
--httpinterface
```

## Importar JSONs

Para cargar los datos en nuestra base de datos mongo utilizaremos mongoimport como se muestra a continuación.

```
sudo docker exec -i -t mongo bash

mongoimport --db practica_bbdd --collection dblp --type json --file
/data/seeds/dblp.json
```

## Calcular colecciones pre-computadas

Para facilitar las consultas del presente trabajo y disminuir su tiempo de ejecución, hemos creado una serie de índices basados en las consultas que vamos a realizar sobre los datos. Además, hemos definido una colección que se calcula a partir de la colección de entrada 'dblp', y que servirá para facilitar y optimizar algunas de las consultas relacionadas con autores.

```
db.dblp.createIndex({tag: 1})
db.dblp.createIndex({author: 1})
db.dblp.createIndex({year: 1, tag: 1})

db.dblp.aggregate(
 [
   {'$match':{'author': { "$exists" : true }}},
   {$unwind:"$author"},
   {$group:{
     _id: "$author",
     publications: { $push:  { ref: "$_id", type: "$tag" } },
     publications_size: {$sum:1},
     inproceedings_size: {$sum: {$cond: [{$eq: ['$tag',
"inproceedings"]}, 1, 0]}},
     articles_size: {$sum: {$cond: [{$eq: ['$tag', "article"]}, 1, 0]}},
     min_year: {$min: "$year"},
     max_year: {$max: "$year"}
   }},
   { $addFields: {author_age : {$subtract: ['$max_year',
'$min_year']}}},
   { $out: "authors" }
 ],
 {allowDiskUse:true}
);

db.authors.createIndex({ _id: 1})
db.authors.createIndex({ author_age: 1})
db.authors.createIndex({ publications_size: 1})
```

## 1.- Listado de todas las publicaciones de un autor determinado

Para calcular el listado de todas las publicaciones de un determinado autor sólo tenemos que hacer una consulta sobre la tabla dblp filtrando por autor.

```
db.dblp.find({ "author" : "Sanjeev Saxena"})
{
  "_id":ObjectId("590dafc1124a9a2d886444ef"),
  "mdate":   ISODate("2010-04-30T00:00:00   Z"),
  "key":"journals/ipl/Saxena10",
  "author":"Sanjeev Saxena",
  "title":"On finding fundamental cut sets.",
  "pages":"168-170",
  "year":2010,
  "volume":110,
  "journal":"Inf. Process. Lett.",
  "number":4,
  "ee":"http://dx.doi.org/10.1016/j.ipl.2009.11.014",
  "url":"db/journals/ipl/ipl110.html#Saxena10"
},
{
  "_id":ObjectId("590dafc1124a9a2d88644762"),
  "mdate":   ISODate("2012-09-26T00:00:00   Z"),
  "key":"journals/ipl/SajithS95",
  "author":[
     "G. Sajith",
     "Sanjeev Saxena"
  ],
  "title":"Corrigendum: Optimal Parallel Algorithms for Coloring Bounded
Degree Graphs and Finding Maximal Independent Sets in Rooted Trees.",
  "pages":305,
  "year":1995,
  "volume":54,
  "journal":"Inf. Process. Lett.",
  "number":5,
  "url":"db/journals/ipl/ipl54.html#SajithS95",
  "ee":"http://dx.doi.org/10.1016/0020-0190(95)00069-O",
  "note":"see: Inf. Process. Lett. 49(6): 303-308(1994)"
}
```

## 2.- Número de publicaciones de un autor determinado

El número de publicaciones de un determinado autor se puede calcular de dos formas, haciendo un count de la consulta del ejercicio 1 (Primera consulta del código a continuación), o utilizar la colección que hemos precalculado (Segunda consulta del código a continuación)

```
 db.dblp.count({"author": "Sanjeev Saxena"})
 db.authors.find({"_id": "Sanjeev Saxena"}, {_id:1, publications_size:
1})

 { "_id" : "Sanjeev Saxena", "publications_size" : 40 }
```

## 3.- Número de artículos en revista para el año 2016

Para calcular el número de artículos en el año 2016, simplemente tenemos que hacer un count de la colección dblp filtrando por año 2016, y tag “artículo”.

```
db.dblp.count({"year": 2016, tag: "article"})
130607
```

## 4.- Número de autores ocasionales, es decir, que tengan menos de 5 publicaciones en total.

El número de publicaciones de un determinado autor se ha precalculado en la colección de autores, por lo que la consulta se reduce a contar el número de documentos donde el campo “publicacions_size” es menor que 5.

```
 db.authors.count({publications_size: {$lt:5}});
 1425459
```

## 5.- Número de artículos de revista (article) y número de artículos en congresos (inproceedings) de los diez autores con más publicaciones totales.

Al igual que el número de publicaciones, hemos precalculado también el número de artículos y artículos en congresos, por lo que simplemente tenemos que ordenar la colección por “publicacions_size”, limitarla a 10 autores y sacar los campos precalculados.

```
 db.authors.aggregate([
```

```
    {$sort: {publications_size: -1}},
    {$limit: 10},
    {$project: {_id:1, articles_size: 1, inproceedings_size: 1}}
 ])

 { "_id" : "H. Vincent Poor", "inproceedings_size" : 441,
"articles_size" : 931 }
 { "_id" : "Philip S. Yu", "inproceedings_size" : 658, "articles_size" :
353 }
 { "_id" : "Wen Gao 0001", "inproceedings_size" : 717, "articles_size" :
300 }
 { "_id" : "Yang Liu", "inproceedings_size" : 576, "articles_size" : 437
}
 { "_id" : "Mohamed-Slim Alouini", "inproceedings_size" : 471,
"articles_size" : 542 }
 { "_id" : "Yu Zhang", "inproceedings_size" : 563, "articles_size" : 429
}
 { "_id" : "Jing Li", "inproceedings_size" : 557, "articles_size" : 423
}
 { "_id" : "Lajos Hanzo", "inproceedings_size" : 401, "articles_size" :
570 }
 { "_id" : "Li Zhang", "inproceedings_size" : 539, "articles_size" : 387
}
 { "_id" : "Wei Li", "inproceedings_size" : 499, "articles_size" : 415 }
```

## 6.- Número medio de autores de todas las publicaciones que tenga en su conjunto de datos.

Calculamos la media del campo precalculado “num_authors”.

```
db.dblp.aggregate([
  {'$match':{'author': { "$exists" : true }}},
  {"$group": {
    "_id": null,
    "avg_pub": { "$avg": "$num_authors" }
  }}
]);

{ "_id" : null, "avg_pub" : 2.2498590562398673 }
```

## 7.- Listado de coautores de un autor (Se denomina coautor a cualquier persona que haya firmado una publicación).

Para encontrar los coautores de un determinado autor, tenemos que buscar todas las publicaciones en las que aparece dicho autor, después hacemos un unwind del campo autores para conseguir un registro por autor, filtramos el autor en cuestión para que no aparezca en la lista y lo mostramos. (Nota, los coautores aparecerán en lista duplicados siempre que aparezcan en más de una ocasión con el autor en cuestión en una publicación)

```
 db.dblp.aggregate([
   {'$match': {$and: [{'author': 'Sanjeev Saxena'}, {num_authors: {$gt:
1}}]}},
   {'$project': {'author': 1, '_id': 0}},
   {$unwind:"$author"},
   {'$match': {'author': {$ne : 'Sanjeev Saxena'}}},
   {$group:{
     _id: null,
     authors: { $push: "$author"}
   }}
 ])

 {
    "_id":null,
    "authors":[
       "G. Sajith", "Sudarshan Banerjee", "Yijie Han", "Xiaojun Shen",
       "G. Sajith", "Jay Mahadeokar", "Neethi K. S.", "Neethi K. S.",
       "Kurt Mehlhorn", "P. C. P. Bhatt", "V. C. Prasad",
       "K. V. R. C.     N. Kishore", "N. Malahal Rao", "G. Sajith",
        "G. Sajith", "G. Sajith", "P. C. P. Bhatt", "Krzysztof Diks",
       "Torben Hagerup", "V. C. Prasad", "Tomasz Radzik", "G. Sajith",
       "V. Yugandhar", "Pramod Chandra P. Bhatt", "V. C. Prasad",
       "Kurt Mehlhorn", "Neethi K. S.", "Neethi K. S.",
       "Jay Mahadeokar", "G. Sajith", "S. Das", "Anjeneya Swami Kare",
       "Niloy Chakrabarti", "Sagar Kalra", "M. R. Tripathy",
       "Jay Mahadeokar", "Yijie Han", "P. C. P. Bhatt",
       "V. C. Prasad", "Anjeneya Swami Kare", "K. Nandan Babu",
       "K. Jeevan Madhu", "Yijie Han"
    ]}
```

## 8.- Edad de los 5 autores con un periodo de publicaciones más largo (Se considera la Edad de un autor al número de años transcurridos desde la fecha de su  primera publicación hasta la última registrada).

La edad es un campo que hemos calculado previamente, por lo que simplemente tenemos que ordenarlo y filtrar los 10 primeros.

```
 db.authors.aggregate([
   {'$project': {'author_age': 1, '_id': 1}},
   {'$sort': {author_age: -1}},
   {$limit: 10},
 ])

{ "_id" : "Alan M. Turing", "author_age" : 75 }
{ "_id" : "Rudolf Carnap", "author_age" : 71 }
{ "_id" : "David Nelson", "author_age" : 67 }
{ "_id" : "Eric Weiss", "author_age" : 64 }
{ "_id" : "C. West Churchman", "author_age" : 63 }
{ "_id" : "Joseph R. Shoenfield", "author_age" : 63 }
{ "_id" : "George E. Collins", "author_age" : 63 }
{ "_id" : "Martin Davis", "author_age" : 63 }
{ "_id" : "Claude E. Shannon", "author_age" : 63 }
{ "_id" : "Bernard Widrow", "author_age" : 63 }
```

## 9.- Número de autores novatos, es decir, que tengan una Edad menor de 5 años (Se considera la edad de un autor al número de años transcurridos desde la fecha de su primera publicación hasta la última registrada).

La edad es un campo precalculado, por lo que sólo tenemos que filtrar por edad menor que 5.

```
db.authors.aggregate([
  {'$project': {'author_age': 1}},
```

```
  {'$match': {author_age: {$lt: 5}}},
  {$count: "Number of Rookies"}
])

{ "Number of Rookies" : 1448120 }
```

## 10.- Porcentaje de publicaciones en revistas con respecto al total de publicaciones.

Para calcular el porcentaje, calculamos el número de artículos y lo dividimos por el número de publicaciones.

```
db.dblp.count({"tag": "article"}) / db.dblp.count()
0.2893011413430495
```

# Neo4j

## Crear un contenedor de Neo4j

Crearnos un contenedor  de Neo4j para la práctica, el nombre (--name), puede ser modificado, así como las rutas locales de los volúmenes.

```
sudo docker run --name neo4j --publish=7474:7474 --publish=7687:7687
--volume=$HOME/neo4j/data:/data --volume=$HOME/neo4j/logs:/logs -v
/home/runix/Projects/other/xml2json/output:/var/lib/neo4j/import/data/se
eds neo4j
```

## Importar datos

Primeramente, creamos los índices de la tabla autor y publicación para acelerar el proceso y evitar duplicados.

Para evitar que el cargador de Neo4j trate de hacer un único commit al final, y por lo tanto, almacene todo el grafo en memoria hasta el final, utilizamos la directiva “USING PERIODIC COMMIT” para hacer commits intermedios.

El funcionamiento de la carga del grafo es el siguiente, cargamos el fichero csv “publications”, y dividimos el campo autor en un array (autores, cada posición representa un autor) a través de unwind. Una vez hecho esto, tenemos que insertar las publicaciones y autores en el caso de que no estén ya creadas, y crear la relación que los une (WRITE)

```
CREATE CONSTRAINT ON (a:AUTHOR) ASSERT a.author_id is UNIQUE;
CREATE CONSTRAINT ON (p:PUBLICATION) ASSERT p.key is UNIQUE;

// Load.
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS
FROM 'file:///data/seeds/publications.csv' AS line
WITH line, split(line.author, ";") AS authors
LIMIT 100000
WHERE line.author IS NOT NULL
UNWIND authors AS author_id
MERGE (publication: PUBLICATION {key: UPPER(line.key), year:
TOINT(line.year), tag: UPPER(line.tag)})
MERGE (author:AUTHOR { author_id: UPPER(author_id) })
CREATE (author)-[:WRITE]->(publication)
```

Nota: Para reducir el tiempo de carga en Neo4j, el número de publicaciones incluidas se ha reducido a cien mil.

## Diferencias entre Neo4j y Mongo

La principal diferencia entre Neo4j y Mongo es cómo almacenamos la información, en Neo4j la información se guarda como grafos, y por lo tanto, se enfatiza la relación entre las diferentes entidades (nodos), mientras que en Mongo la información se agrupa en torno a documentos, dando más importancia a la estructura de la información en su conjunto.

Dado el carácter de base de datos orientada a grafos de Neo4j, las consultas en las cuales Neo4j obtiene una mejora sobre Mongo son aquellas en las cuales nos estamos centrando en las relaciones que hay entre autores y publicaciones, y no en elementos individuales.

## 1.- Número de publicaciones de un autor determinado

En Neo4j, calcular el número de publicaciones de un determinado autor (nodo), se reduce a calcular el grado del nodo, que se puede calcular de la siguiente forma.

```
MATCH (p:AUTHOR)-[r:WRITE]->(pub:PUBLICATION)
WITH p, count(pub) as pub_num
WHERE p.author_id = 'SANJEEV SAXENA'
RETURN p.author_id, pub_num
```

## 2.- Número de autores ocasionales, es decir, que tengan menos de 5 publicaciones en total.

Al igual que la consulta anterior, la consulta se reduce a contar el número de autores cuyo grado sea mayor o igual que 5.

```
MATCH (p:AUTHOR)-[r:WRITE]->(pub:PUBLICATION)
WITH p, count(pub) as pub_num
WHERE pub_num < 5
RETURN count(p)
```

## 3.- Número medio de autores de todas las publicaciones que tenga en su conjunto de datos.

Para calcular el número de autores medios, simplemente tenemos que calcular el grado medio de todas las publicaciones.

```
MATCH (pub:PUBLICATION) <- [r:WRITE] - (p:AUTHOR)
WITH pub, count(p) as pub_num
RETURN avg(pub_num)
```