PRÁCTICA I BBDD

MongoDB

Carlos Grande Núñez, Pablo Olmos Martínez y Verónica Gómez Gómez

UNIVERSIDAD REY JUAN CARLOS

[ÍNDICE]

01 Introducción a la práctica

Descripción

La práctica propuesta consiste en la creación de una base de datos en Mongo a partir de la base de datos DBLP Computer Science Bibliography (*https://dblp.uni-trier.de/db/*) y realizar las 10 consultas propuestas para la extracción de´información.

Objetivos

- Generar un base de datos en MongoDB a partir de la base de datos DBLP.
- Procesar de los datos originales.
- Crear una nueva base de datos en MongoDB mediante los datos procesados.
- Realizar las consultas propuestas para la práctica.

Metodología:

1. Captura y procesamiento de datos: descarga del fichero *dblp.xml* y procesado al formato JSON mediante el script *01_xmlParser.py*. En este procesamiento se seleccionan as variables de necesarias.
2. Almacenamiento de datos: tras la generación del fichero JSON se genera una nueva base de datos en MongoDB donde se cargan y se estructuran los datos del fichero. Para este ejercicio se van a crear dos colecciones dentro de la base de datos, la colección de 'publicaciones' y la colección de 'autores'.
3. Finalmente se van a realizar las 10 consultas propuestas en la práctica.

02 Elaboración de la práctica

2.1 Situación de los archivos originales y generados

Los archivos usados para la elaboración de la práctica son:

01

- *dblp.xml* (Fichero XML de la base de datos propuesta DBLP).
- _README.txt (Fichero con las instrucciones para la reproducción del ejercicio).
- *01_xmlParser*.py (Script para la lectura y el procesado de los datos a un fichero json).
- *02_MongoLoader* (Script para la creación y carga de la base de datos en MongoDB).
- *03_MongoFinder* (Script para realizar las consultas propuestas).
- *MDS_Memoria_GrandeCarlosYOlmosPabloYGomezVeronica.pdf* (Memoria de la práctica).
- *Resultados.txt* (Resultado final de las consultas llevadas a cabo).

02

2.2 Lectura y procesado de DBLP

El primer paso para la elaboración de la práctica es el procesado y filtrado del fichero dblp.xml, para poder realizar este paso se ha generado el script 01_xmlParser.py que permite leer y generar un fichero JSON con la información necesaria para la práctica.

01_xmlParser.py Permite parsear el fichero dblp.xml y generar un archivo JSON con la información procesada.

2.1

```
# all of the element types in dblp
selected_elements = {"article", "inproceedings", "incollection"}
# all of the feature types in dblp
features = {"author", "booktitle", "editor", "isbn", "pages", "publisher", "title", "year"}

json_path = "dblp_parsed.json"
f = open(json_path, "w", encoding="utf8")
tree = etree.iterparse(source=dblp_path, dtd_validation=True, load_dtd=True)
counter = 0
for _ , branch in tree:
    if branch.tag in selected_elements:
        if counter%100000 == 0:
            print(counter)
        counter += 1
        attributes = extractor(branch, features)
        f.write(str(attributes) + "\n")
        clear_branch(branch)
f.close()
# Ejemplo de un branch
print(attributes)
```

2.2

2.3

{"id": ["journals/eswa/HussainLCB15"], "branch": ["article"], "pages": ["8"], "editor": [], "title": ["An expert system for acoustic diagnosis of power circuit breakers and on-load tap changers."], "year": ["2015"], "isbn": [], "publisher": [], "booktitle": [], "author": ["Akhtar Hussain", "Seung-Jae Lee", "Myeon-Song Choi", "Fouad Brikci"]}

Mediante el siguiente fragmento de código se va iterando por los diferentes documentos del xml procesando cada una de las entradas mediante la función extractor en un diccionario para su posterior guardado en el fichero JSON.

2.4

En este primer script se guardaran aquellos branch que contengan las siguientes variables "author", "booktitle", "editor", "isbn", "pages", "publisher", "title" y "year" y que pertenezcan a las categorías "article", "improceedings" e "incollection". El número total de entradas procesadas es de 4 884 720.

2.3 Cargado de datos en MongoDB

En este apartado se crea una nueva base de datos en MongoDB y se añaden los datos del fichero JSON. El fichero JSON contiene todas las entradas procesadas anteriormente con la siguiente estructura.

dblp_parsed.json Fragmento de la estructura del fichero JSON generado.

01


```
{"id": ["tr/meltdown/s18"], "branch": ["article"], "pages": [], "editor": [], "title": ["Spectre Attacks: Exploiting Speculative Execution."], "year": ["2018"], "isbn": [], "publisher": [], "booktitle": [], "author": ["Paul Kocher", "Daniel Genkin", "Daniel Gruss", "Werner Haas", "Mike Hamburg", "Moritz Lipp", "Stefan Mangard", "Thomas Prescher 0002", "Michael Schwarz 0001", "Yuval Yarom"]}
```


El fragmento del siguiente script crea y carga las 4 millones de entradas en la nueva base de datos generada en MongoDB. Tras la carga de datos de la publicaciones se genera un nueva colección de autores que nos permita agilizar las búsquedas del apartado final de la práctica.

02 *02_MongoLoader.py Permite crear una base de datos en MongoDB y cargar la información del fichero JSON.*

```
# Connecting to mongo
uri = 'mongodb://127.0.0.1:27017'
client = pm.MongoClient(uri)
print('Connection stablished...')

# Creating database
db = client['dblp']
collection = 'publicaciones'
print('Database created')

# load database from json
json_path = 'dblp_parsed.json'
load_json(json_path, db, collection)
print('Database loaded')

# Creating new collection authors
pipe_authors = [{"$unwind": "$author"},
        {"$group": {"_id": "$author","publications": {"$push": "$_id"}}},
        {"$out": "autores"}]
db.publicaciones.aggregate(pipe_authors,  allowDiskUse = True)
```

2.1

2.2

Colección publicaciones

2.3 [{'_id': ObjectId('5e93b1036055d774f6fd6707'), 'id': ['tr/meltdown/s18'], 'branch': ['article'], 'publisher': [], 'title': ['Spectre Attacks: Exploiting Speculative Execution.'], 'pages': [], 'author': ['Paul Kocher', 'Daniel Genkin', 'Daniel Gruss', 'Werner Haas', 'Mike Hamburg', 'Moritz Lipp', 'Stefan Mangard', 'Thomas Prescher 0002', 'Michael Schwarz 0001', 'Yuval Yarom'], 'editor': [], 'isbn': [], 'booktitle': [], 'year': ['2018']}]

Coleccions autores

[{'_id': '"Johann" Sebastian Rudolph', 'publications': [ObjectId('5e93b46f6055d774f62b6f14')]}, {'_id': "'Anau Mesui", 'publications': [ObjectId('5e93b4276055d774f627c261')]}, {'_id': "'Maseka Lesaoana", 'publications': [ObjectId('5e93b1796055d774f603b0bc'), ObjectId('5e93b2526055d774f60ee3bc')]}]

2.4

2.4 Ejecución de las consultas

En este apartado se mostrarán los pipes de las consultas realizadas con sus resultados. El resto del código puede encontrarse en el script *03_MongoFinder.py*

Query 01. Listado de todas las publicaciones de un autor determinado..

```
pipe = [{'$match': {'author': 'A-Nasser Ansari'}},
        {'$project': {'_id': 0, 'title': 1}}]
```

	title
0	[Tracking multiple people with recovery from partial and total occlusion.]
1	[Automatic facial feature extraction and 3D face modeling using two orthogonal views with applic...
2	[Improved Active Shape Model for Facial Feature Extraction in Color Images.]
3	[A multimodal approach for 3D face modeling and recognition using 3D deformable facial mask.]
4	[3D Face Mesh Modeling from Range Images for 3D Face Recognition.]
5	[Disparity-Based 3D Face Modeling for 3D Face Recognition.]
6	[Multi-modal (2-D and 3-D) face modeling and recognition using Attributed Relational Graph.]
7	[3-D Face Modeling Using Two Views and a Generic Face Model with Application to 3-D Face Recogni...
8	[Disparity-Based 3D Face Modeling using 3D Deformable Facial Mask for 3D Face Recognition.]
9	[3D face modeling using two orthogonal views and a generic face model.]

Query 02. Número de publicaciones de un autor determinado.

```
pipe = [{'$match': {'author': 'A-Nasser Ansari'}},
        {'$project': {'_id': 0, 'title': 1}},
        {'$count': 'Anasari_publications'}]
```

[{'Anasari_publications': 10}]

Query 03. Número de artículos en revista para el año 2018.

```
pipe = [{'$match': {'branch': 'article', 'year': '2018'}},
        {'$count': 'Articles_2018'}]
```

[{'Articles_2018': 179954}]

Query 04. Número de autores ocasionales, es decir, que tengan menos de 5 publicaciones en total.

```
pipe = [{'$match': {'branch': 'article', 'year': '2018'}},
        {'$count': 'Articles_2018'}]
```

[{'Authors < 5 publications': 14961375}]

Query 05. Número de artículos de revista y de artículos en congresos de los diez autores con más publicaciones totales.

```
pipe = [{'$project': {'_id': '$_id', 'n_publications': {'$size': '$publications'}, 'publications': '$publications'}},
        {'$sort': {'n_publications': -1}},
        {'$limit': 11},
        {'$unwind': '$publications'},
        {'$lookup':{'from': 'publicaciones','localField': 'publications', 'foreignField': '_id', 'as': 'document'}},
        {'$project': {'_id': '$_id', 'type': '$document.branch'}},
        {'$unwind': '$type'}, {'$unwind': '$type'},
        {'$match': {'$or': [{'type': 'article'}, {'type': 'inproceedings'}]}},
        {'$group': {'_id': '$type', 'n': {'$sum': 1}}}]
```

	_id	number
0	article	7434
1	inproceedings	7268

Query 06. Número medio de autores de todas las publicaciones que tenga en su conjunto de datos.

```
pipe = [{'$unwind': '$author'},
        {'$group': {'_id': {'id': '$_id', 'branch': '$branch'}, 'n_authors': {'$sum': 1}}},
        {'$project': {'_id': '$_id.id', 'branch': '$_id.branch', 'n_authors': '$n_authors'}},
        {'$group': {'_id': '$branch', 'avg_authors': {'$avg': '$n_authors'}}}]
```

01

	_id	avg_authors
0	[incollection]	2.338645
1	[article]	2.981176
2	[inproceedings]	3.179692

Query 07. Listado de coautores de un autor.

```
pipe = [{'$unwind': '$publications'},
        {'$match': {'_id': 'A-Nasser Ansari'}},
        {'$lookup': {'from': 'publicaciones', 'localField': 'publications', 'foreignField': '_id', 'as': 'document'}},
        {'$project': {'_id': '$_id', 'coauthor': '$document.author'}},
        {'$unwind': '$coauthor'},
        {'$unwind': '$coauthor'},
        {'$match': {'coauthor': {'$ne': 'A-Nasser Ansari'}}},
        {'$group': {'_id': '$_id', 'coauthor': {'$addToSet': '$coauthor'}}}]
```

02

```
[{'_id': 'A-Nasser Ansari', 'coauthor': ['Mohamed Abdel-Mottaleb', 'Charay Lerdsudwichai', 'Mohammad H. Mahoor']}]
```

2.1 *Query 08. Edad de los 5 autores con un periodo de publicaciones más largo.*

```
pipe = [{'$unwind': '$publications'},
        {'$lookup':{'from': 'publicaciones', 'localField': 'publications', 'foreignField': '_id', 'as': 'document'}},
        {'$project': {'_id': '$_id', 'year': '$document.year'}},
        {'$unwind': '$year'}, {'$unwind': '$year'},
        {'$limit': 10000},
        {'$group': {'_id': '$_id', 'max_year': {'$max': '$year'}, 'min_year': {'$min': '$year'}}},
        {'$project': {'_id': '$_id', 'age': {'$subtract': [{'$toInt': '$max_year'}, {'$toInt': '$min_year'}]}}},
        {'$sort': {'age': -1}},
        {'$limit': 5}]
```

2.2

	_id	age
0	A. Denisov	55
1	A. C. Smith	50
2	A. Bensoussan	49
3	A. Alan B. Pritsker	48
4	A. Chattopadhyay	46

2.3 *Query 09. Número de autores novatos.*

```
# Same pipe as query 8 adding
        {'$match': {'age': {'$lt': 5}}},
        {'$count': 'Novels authors'}]
```

```
[{'Novels authors': 4337}]
```

2.4 *Query 10. Número de autores novatos.*

```
pipe = [{'$project': {'_id': '1', 'magazines': {'$cond': [{'$eq': ['$branch', ['article']]}, 1, 0]}}},
        {'$group': {'_id': '$_id', 'total_magazines': {'$sum': '$magazines'}, 'total_docs': {'$sum': 1}}},
        {'$project': {'Magazines percentage': {'$round': [{'$multiply': [{'$divide': ['$total_magazines', '$total_docs']}, 100]}, 0]}}}]
```

```
[{'_id': '1', 'Magazines percentage': 46.0}]
```

ANEXO: LIBRERÍAS Y FUNCIONES USADAS

Librerías usadas para la práctica

```
from lxml import etree
import json
import re
import pymongo as pm
import pandas as pd
```

Otras funciones usadas para la práctica

```
def clear_branch(element):
    """Free up memory for temporary element tree after processing the element"""
    while element.getprevious() is not None:
        del element.getparent()[0]

def extractor(branch, features):
    """Extract the value of each feature"""
    attribs = {"id": [branch.attrib["key"]], "branch": [branch.tag]}
    for feature in features:
        attribs[feature] = []
    for sub in branch:
        if sub.tag not in features:
            continue
        if sub.tag == "title":
            text = re.sub("<.*?>", "", etree.tostring(sub).decode("utf-8")) if sub.text is None else sub.text
        elif sub.tag == "pages":
            text = count_pages(sub.text)
        else:
            text = sub.text
        if text is not None and len(text) > 0:
            attribs[sub.tag] = attribs.get(sub.tag) + [text]
    attribs = json.dumps(attribs)
    return attribs

def count_pages(pages):
    cnt = 0
    for part in re.compile(r",").split(pages):
        subparts = re.compile(r"-").split(part)
        if len(subparts) > 2:
            continue
        else:
            try:
                re_digits = re.compile(r"[\d]+")
                subparts = [int(re_digits.findall(sub)[-1]) for sub in subparts]
            except IndexError:
                continue
            cnt += 1 if len(subparts) == 1 else subparts[1] - subparts[0] + 1
    return "" if cnt == 0 else str(cnt)
```

```
def load_json(path, db, collection = 'test'):
    collection = db[collection]
    with open(path, 'r') as handle:
        for line in handle:
            data = json.loads(line)
            collection.insert_one(data)
    client.close()

def head(cursor, n = 3):
    hd = list(cursor.limit(n))
    print(hd)

def printer(lst):
        pd.set_option('max_colwidth', 100)
        df = pd.DataFrame({})
        keys = lst[0].keys()
        values = [list(i.values()) for i in lst]
        for key, col in zip(keys, range(len(keys))):
                df[key] = [i[col] for i in values]
        print(df)
```

01

02

2.1

2.2

2.3

2.4

A