POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Stefano Baroni**
Luca Brembilla
Alessia Menozzi
Caterina Giardi
Rocco Agnello

Group Number: **33**

Academic Year: 2022-2023

Contents

Part I

Delivery 1

1 | Introduction

The aim of this project is to build a model for a database that contains bibliography concerning scientific papers.
First of all we have modeled the bibliographic database creating an E-R model to better model the data. Then we downloaded a json file containing all the information about the papers, authors, venue and the different fields of study.
We decided to decrease the dimension of the json file because Neo4J was becoming too slow and after the data import we worked on different queries to retrieve information and play with the nodes.
In this document we will further explain the procedures applied to import a given dataset on Neo4j, then we will discuss the structure of the dataset itself and finally we will test different types of queries and show the obtained results using screenshots.

2 | Entities and Relations

2.1. Introduction

We modeled the E-R model using erdplus.com. This model is based on the json file containing all the fields of the AMiner database given by the specifications of the project. It is modeled in order to be a general overview of what a bibliographic database might look like.

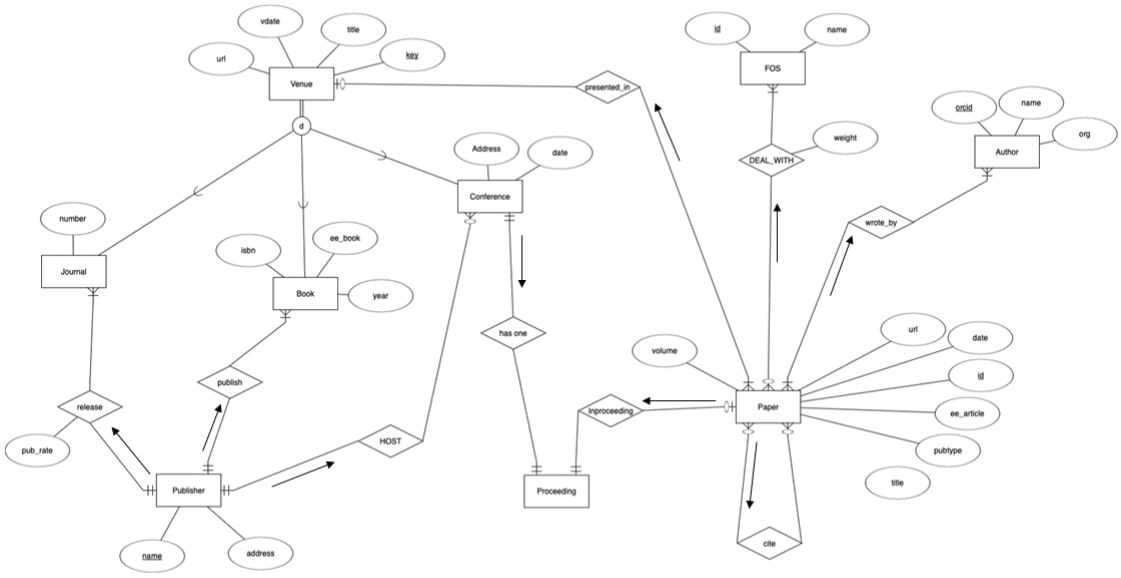


Figure 2.1: E-R Model

2.2. Description

We created nine entities:

- **Paper**, this entity is the core of our graph and that's why it contains the highest number of attributes: id (the primary key), title, date, URL, volume, pub_type (publication type f.i. informal, scientific, ...), and ee_article, that represents the position of the electronic edition. One paper can also contains citations of other papers and we represented it by inserting the *cite* relationship that connects two papers and has a 0-n cardinality for both.
- **Author**, characterized by attributes such as name, organization and orcid (primary key) that is an international unique identifier for scientific and academic authors. It is connected to the entity Paper through the relationship wrote_by: every author has written from 1 to n papers and every paper has from 1 to n authors.
- **FOS** (Field of Study) which is the topic of the paper, it has an id as primary key and a name. It's connected to it through the relationship *DEAL_WITH* that has cardinality 0-n on the paper's side and cardinality 1-n on the FOS's side. This relationship has the attribute *weight* that captures how much a topic is present inside the article.
- **Venue**, this is a supertype that includes 4 attributes: URL, vdate, title and the primary key. A Venue is connected to the Paper with the relationship *presented_in* that has cardinality 1-n on the Venue's side and 0-1 on the Paper's side The attributes are inherited by the 3 disjoined subtypes.
  - **Journal**: since it can be compared to a periodic newspaper, we added the attribute *number*. Every journal must be connected to one Publisher using the relationship *release* that has the attribute *pub_rate* which describes the frequency of publication of every journal (weekly, daily, monthly...). On the other side, each publisher is connected to at least one journal.
  - **Book**: its attributes are the *year* in which the book was published, the *ISBN* code (an International Standard Book Number) and *ee_book*, which contains the position of the electronic edition. It's connected to the entity Publisher with the relationship *publish* and with the same cardinalities as the entity above.
  - **Conference**: its attributes are the date in which the conference took place and the address. This entity is connected to Publisher: a Publisher can host

from 0 to multiple Conference(s), and a Conference must be hosted by one Publisher.

- **Publisher**, it's the one that publishes books or journals, both containing one or more papers. Otherwise, it can host a conference, in which one or more papers are presented. Its attributes are name (primary key), and address.
- **Proceeding**, it represents a collection of papers presented in a conference. It's connected to the entity Conference with a one-to-one mandatory relation on both sides. It's also connected to the entity Paper through the relationship *inproceeding* which is optional: one proceeding can contain from 0 to multiple papers and one paper can be present in at most one conference. Note: This entity was inserted on the E-R diagram in order to make it the more general as possible but it doesn't appear on our specific dataset.

3 | Import

3.1. Choice of the dataset

We were given more choices about the dataset. So we have tried many of them and we then choose the best one.

First of all, we used the dataset by dblp. The file provided here is an xml. However, the bad thing about it is that a paper isn't linked to the papers cited, and we thought this is something not convenient for a project like this, because in bibliography the citations are one of the most important things.

For example, one paper in this file is:

```
<article mdate="2020-06-25" key="tr/meltdown/s18" publtype="informal">
    <author>Paul Kocher</author>
    <author>Daniel Genkin</author>
    <author>Daniel Gruss</author>
    <author>Werner Haas 0004</author>
    <author>Mike Hamburg</author>
    <author>Moritz Lipp</author>
    <author>Stefan Mangard</author>
    <author>Thomas Prescher 0002</author>
    <author>Michael Schwarz 0001</author>
    <author>Yuval Yarom</author>
    <title>Spectre Attacks: Exploiting Speculative Execution.</title>
    <journal>meltdownattack.com</journal>
    <year>2018</year>
    <ee type="oa">https://spectreattack.com/spectre.pdf</ee>
</article>
```

Moreover, as you can see, the authors are not identified by an id, so we would have to add the constraint that the key was their name and that it was not possible to have homonyms.

The other datasets tried are given by AMiner. They were all in json format. The citation data is extracted from DBLP (where we found the dataset above), ACM, MAG (Microsoft Academic Graph), and other sources. The first versions (till version 10) were too simple and spare. So, we opted for version 10. An example of a paper in that dataset is provided below. We now have references with other papers.


```
{
    "abstract": "The purpose of this study is to develop a learning tool
    for high school students studying the scientific aspects of
   information and communication net- works. More specifically, we focus
    on the basic principles of network proto- cols as the aim to develop
    our learning tool. Our tool gives students hands-on experience to
   help understand the basic principles of network protocols.",
    "authors": ["Makoto Satoh", "Ryo Muramatsu", "Mizue Kayama", "
   Kazunori Itoh", "Masami Hashimoto", "Makoto Otani", "Michio Shimizu",
    "Masahiko Sugimoto"],
    "n_citation": 0,
    "references": ["51c7e02e-f5ed-431a-8cf5-f761f266d4be", "69b625b9-
   ebc5-4b60-b385-8a07945f5de9"],
    "title": "Preliminary Design of a Network Protocol Learning Tool
   Based on the Comprehension of High School Students: Design by an
   Empirical Study Using a Simple Mind Map",
    "venue": "international conference on human-computer interaction",
    "year": 2013,
    "id": "00127ee2-cb05-48ce-bc49-9de556b93346"
}
```


Again, one thing not provided is the id of an author, but that's good enough.

However, to increase the number of nodes and edges, we should use version 12, where there are also "fos", i.e. fields of study, publishers, id of authors and many other properties.

3.2. Query import

We have used 3 different scripts to import the dataset. Obviously, we had to create the nodes (first script), then we had to delete some nodes that were "mistaken" (second and third scripts).

```
// apoc.load.json is used to load a json into Neo4j
CALL apoc.load.json("file:///dbUsed.json")
YIELD value
// Creation of papers, with the property id, title, year and n_citation
UNWIND value as p
MERGE (paper:Paper {id: p.id})
ON CREATE SET paper.title = p.title,
        paper.id = p.id,
              paper.year = p.year,
              paper.n_citation = p.n_citation

// For each paper, we create the venue linked
MERGE (venue:Venue {raw: p.venue.raw})
ON CREATE SET venue.id = p.venue.id,
        venue.type = p.venue.type
MERGE (paper)-[:PRESENTED_IN]->(venue)

// For each paper, we create the publisher linked
MERGE (publisher:Publisher {id: p.publisher})
MERGE (paper)-[:PUBLISHED_BY]->(publisher)

// Each paper has a list of references. We create each reference in that list and link it with the paper
FOREACH (ref IN p.references |
  MERGE (paper2:Paper {id: ref})
  MERGE (paper)-[:CITE]->(paper2)
)

// Each paper has a list of authors. We create each author in that list and link it with the paper
FOREACH (a IN p.authors |
  MERGE (author:Author {id:a.id})
  ON CREATE SET author.name = a.name,
              author.org = a.org
  MERGE (author)-[:WROTE]->(paper)
)

```

```
36    // Each paper has a list of "fos" (fields of study). We create each
      author in that list and link it with the paper
37    FOREACH (f IN p.fos |
38      MERGE (fos:Fos {id: f.name})
39      MERGE (paper)-[:DEAL_WITH {weight:f.w}]->(fos)
40    )
```

Listing 3.1: Query used to import and create the database

Some papers don't have a publisher but that's provided anyways with empty string. So we need to delete it.

```
1    MATCH (p:Publisher)
2    WHERE p.id = ""
3    DETACH DELETE p
```

Listing 3.2: To delete the Publishers without name

In the same way, some paper don't have a venue but that's provided anyway with empty string. So we need to delete it.

```
1    MATCH (v:Venue)
2    WHERE v.raw=""
3    DETACH DELETE v
```

Listing 3.3: To delete the Venues without name

3.3. Graph DB

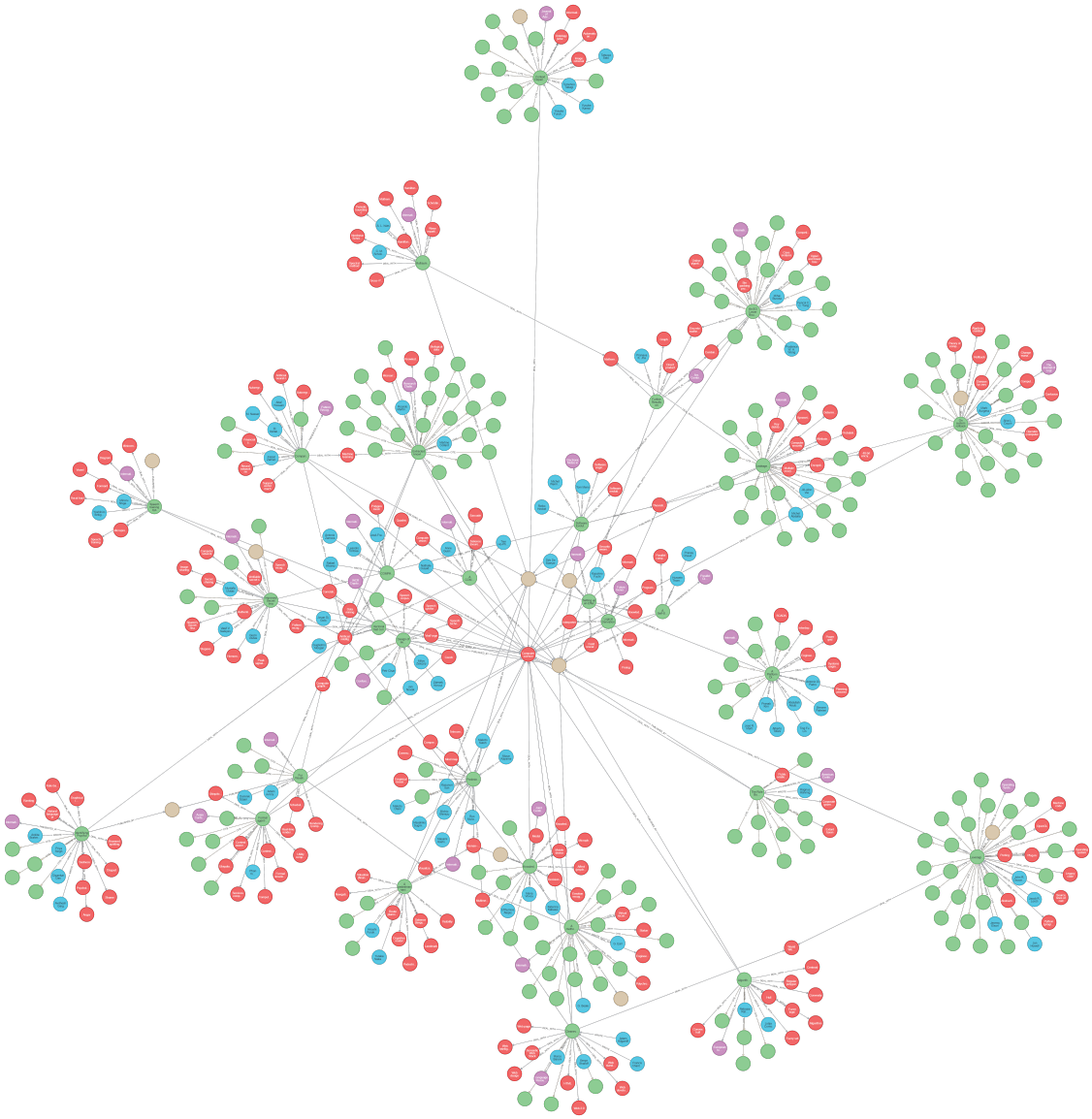


Figure 3.1: Graph Database

Description of the graph:
ENTITIES:

- Venue, whose attributes are id, raw(the name of the venue), numpages, type: J (for journal), C (for conference) and B (for Book)
- Author, whose attributes are id, name and org (optional)
- Paper, whose attributes are id, n_citations (number of citations included in that paper), title and year

- FOS, identified by an id which is their name
- Publisher, identified by an id which is their name

RELATIONS:

- CITE, links two papers that contains a citation on each other
- PUBLISHED_BY, links a paper to a publisher
- PRESENTED_IN, links a paper to a venue
- WROTE, links an author to a paper
- DEAL_WITH, links a paper to a field of study. This is the only relation that contains an attribute (weight). This attribute indicates how much that topic is present in the overall content of that specific paper.

4 | Query

4.1. Creation/Update

In the following queries we will create a new paper, associate it with an already existing author, and connect it with the fields of study that characterize the paper. These queries only represent an example of creation and update on a node, other queries regarding data creation have been used for the import.

4.1.1. Query 1

```
1    CREATE (p:Paper {title:"prova", id: 5000, year: 2022, n_citation:
     0})
```

Listing 4.1: Creation of a new Paper

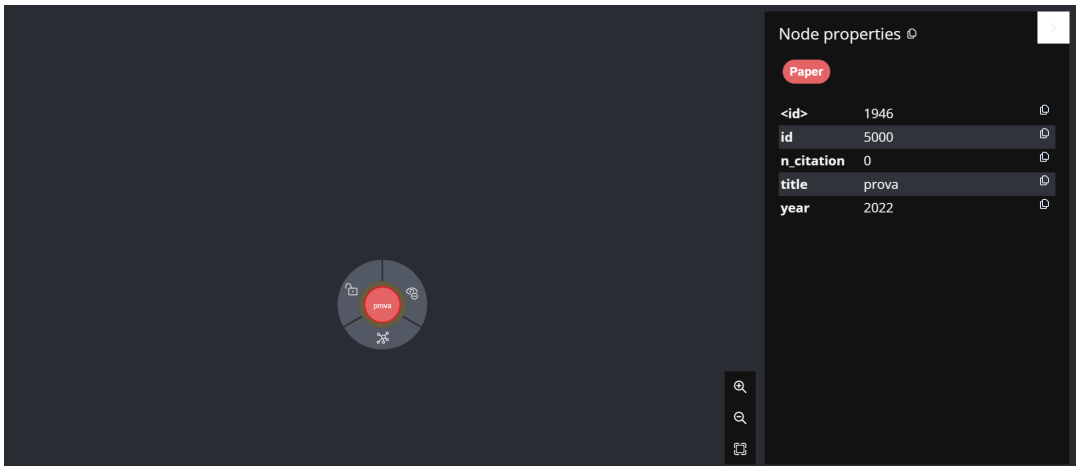


Figure 4.1: Result of new paper query

4.1.2. Query 2

```
1    MATCH (p:Paper), (fos:Fos)
2    WHERE p.title = 'prova' AND (fos.id = 'Mind map' OR fos.id = "
     Machine learning")
```

```
3    CREATE (p)-[r:DEAL_WITH {weight: 0.5}]->(fos)
```

Listing 4.2: Creation of a relationship between the new paper and its FOS

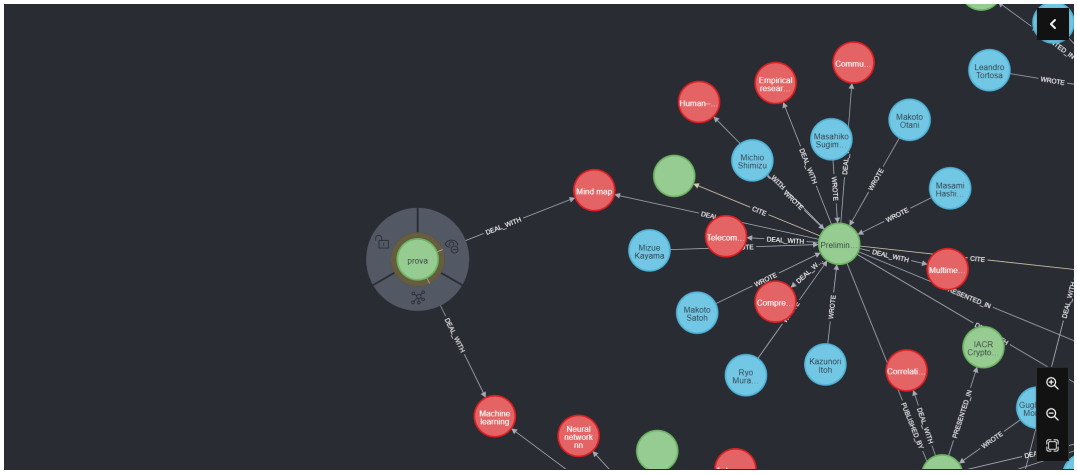


Figure 4.2: Result of creation query 2

4.1.3. Query 3

```
1    MATCH (a:Author), (p:Paper)
2    WHERE p.title = 'prova' AND (a.name = 'Rafael Alvarez' OR a.name = '
     Masahiko Sugimoto')
3    CREATE (a)-[r:WROTE]->(p)
```

Listing 4.3: Creation of a relationship between the new paper and its authors

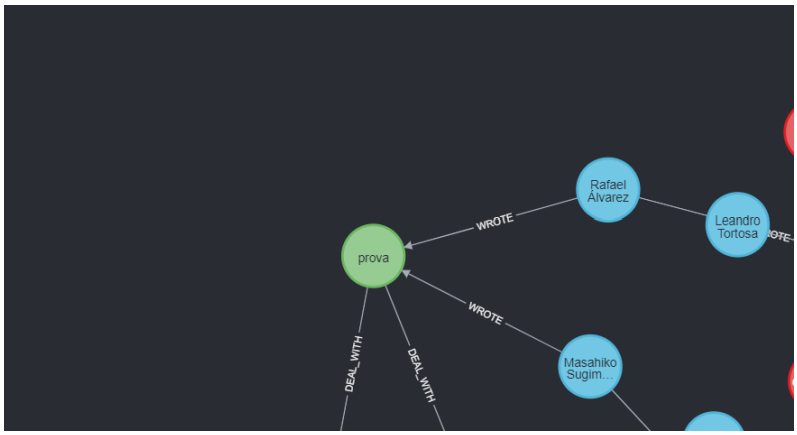


Figure 4.3: Result of creation query 3

4.1.4. Query 4

The first update query will perform the insertion of a new attribute numpages set to 20 in the case of the new paper we have just added to the database with the previous queries.

```
MATCH (p:Paper {title: 'prova'})
SET p.numpages = 20
RETURN p
```

Listing 4.4: Update Query 1

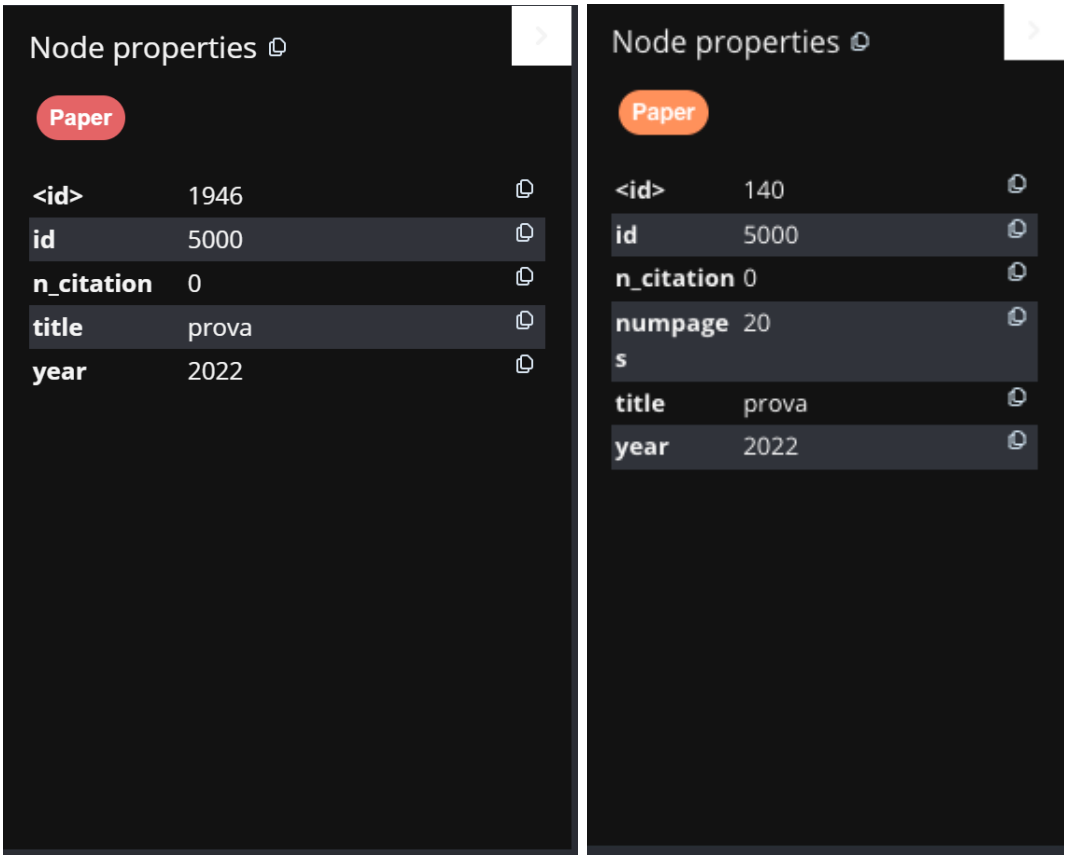


Figure 4.4: Result of the update query, before and after

4.1.5. Query 5

If this update query finds a match based on the title of the paper, then the attribute *year* is incremented by 1, otherwise a new paper with that title and the year set on 2020 is created.

```
MERGE (p:Paper{title:"Fur Visualisation for Computer Game
Engines and Real-Time Rendering"})
ON CREATE SET p.year="2020"
```

```
        ON MATCH SET p.year= p.year +1
        return p
```

Listing 4.5: Update Query 2

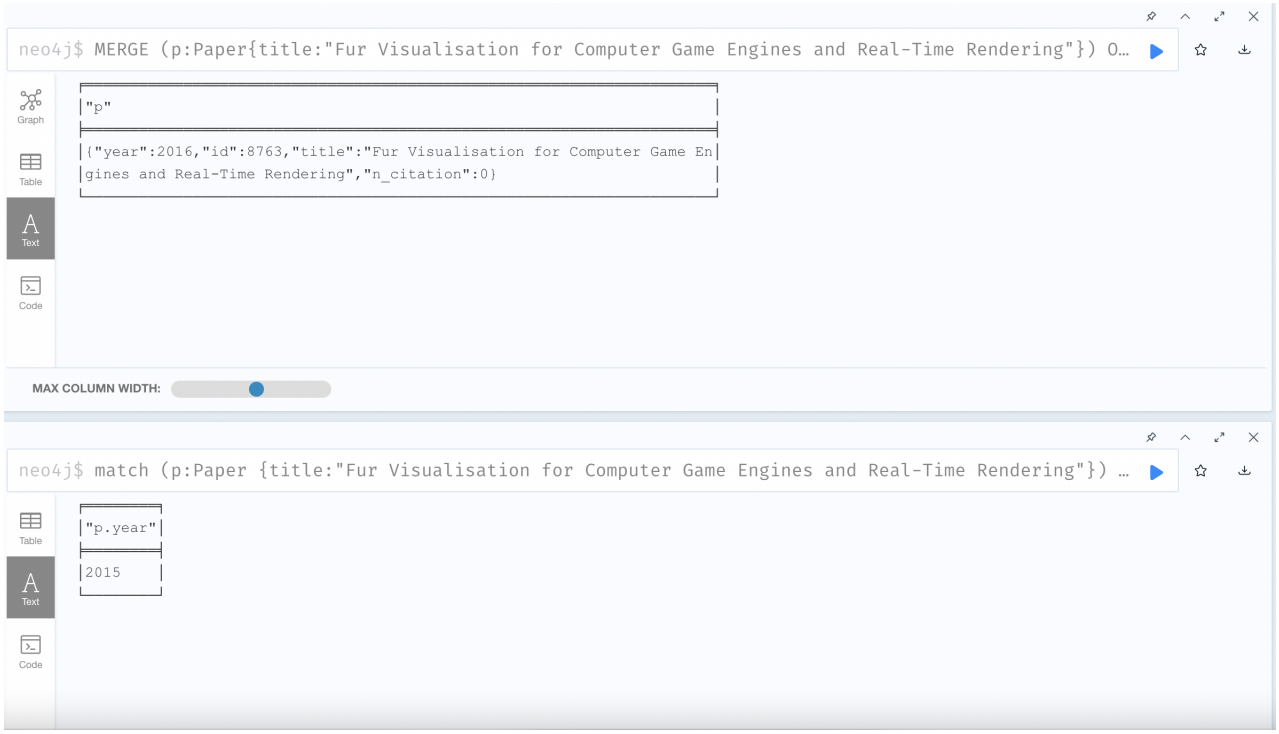


Figure 4.5: In the second row the non modified Paper, in the first one the modified one

4.2. Information Retrieval

In this section we will show queries that retrieve different kind of data. We tried to involve different nodes, using constraints and limitations.
Note that for some queries we have created datasets using other data different from those used in the graph reported in chapter 3.3. All queries would work if the database were created using all the data in the original file.

4.2.1. Query 1

This query returns the Venues of type C (Conference) and the Papers presented there. There is a constraint on the Papers: they must have been produced by at least 4 authors.

```
    MATCH (v:Venue)<-[:PRESENTED_IN]-(p:Paper)<-[:WROTE]-(a:Author)
    WHERE v.type = "C"
    WITH v.raw AS vtitle, p.title AS ptitle, collect(a.name) AS authors
```

```
4       WHERE size(authors) > 3
5       RETURN vtitle, ptitle, authors
```

Listing 4.6: Query 1

"vtitle"	"ptitle"	"authors"
"International Conference on Human-Computer Interaction"	"Preliminary Design of a Network Protocol Learning Tool Based on the Comprehension of High School Students: Design by an Empirical Study Using a Simple Mind Map"	["Kazunori Itoh","Ryo Muramatsu","Michio Shimizu","Masahiko Sugimoto","Makoto Otani","Mizue Kayama","Makoto Satoh","Masami Hashimoto"]
"Pattern Recognition and Machine Intelligence"	"Comparison of GARCH, Neural Network and Support Vector Machine in Financial Time Series Prediction"	["M. Nasser","Altaf Hossain","M. Mufakkharul Islam","Faisal Zaman"]
"International Conference on Computer Graphics Theory and Applications"	"COMPARING GNG3D AND QUADRIC ERROR METRICS METHODS TO SIMPLIFY 3D MESHES"	["José-Francisco Vicent","Leandro Tortosa","Antonio Zamora","Rafael Álvarez"]
"International Conference on Mining Intelligence and Knowledge Exploration"	"Identifying Psychological Theme Words from Emotion Annotated Interviews"	["Ankita Brahmachari","Priya Singh","Dipankar Das","Avdhesh Garg"]
"Conference of the International Speech Communication Association"	"Design of an audio-visual speech corpus for the czech audio-visual speech synthesis."	["Petr Císar","Zdenek Krnoul","Milos Železný","Jan Novák"]
"International Conference on Critical Infrastructure Protection"	"A Platform for Disaster Response Planning with Interdependency Simulation Functionality"	["José R. Martí","Pranab Kini","Abdullah Alsubaie","Alberto Tofani","Simone Palmieri","Ting Fu Lin","Antonio Di Pietro"]
"Language Resources and Evaluation"	"Cleaneval: a Competition for Cleaning Web Pages"	["Marco Baroni","Adam Kilgarriff","Francis Chantree","Serge Sharoff"]
"Operating Systems Design and Implementation"	"Leveraging legacy code to deploy desktop applications on the web"	["Jeremy Elson","Jacob R. Lorch","John R. Douceur","Jon Howell"]

Figure 4.6: Result of query 1

4.2.2. Query 2

For every paper, the query returns the list of all FOS having a weight of at least 0.5 and having a number of citations that is less that 5.

```
1       MATCH(p:Paper)-[d:DEAL_WITH]->(f:Fos)
2       WHERE d.weight >= 0.5 and p.n_citation < 5
3       WITH p.title as title, collect(f.id) as topics, p.n_citation as
        citation
4       RETURN title, topics, citation
```

Listing 4.7: Query 2

"title"	"topics"	"citation"
"Preliminary Design of a Network Protocol Learning Tool Based on the Comprehension of High School Students: Design by an Empirical Study Using a Simple Mind Map"	["Mind map","Communications protocol"]	1
"Further Results on Independence in Direct-Product Graphs."	["Direct product"]	1
"COMPARING GNG3D AND QUADRIC ERROR METRICS METHODS TO SIMPLIFY 3D MESHES"	["Quadric","Polygon mesh"]	0
"Vectorial fast correlation attacks."	["Correlation attack"]	2
"Improved Secret Image Sharing Method By Encoding Shared Values With Authentication Bits"	["Peak signal-to-noise ratio","Shamir's Secret Sharing","Verifiable secret sharing","Secret sharing","Homomorphic secret sharing"]	0
"Formal agent-oriented ubiquitous computing: a computational intelligence support for information and services integration"	["Computational intelligence","Context-aware pervasive systems","Ubiquitous commerce","Context awareness","Utility computing","Services computing","Ubiquitous computing"]	0
"Fur Visualisation for Computer Game Engines and Real-Time Rendering"	["Visualization","Rendering (computer graphics)","Real-time rendering"]	0
"Identifying Psychological Theme Words from Emotion Annotated Interviews"	["Keyword spotting","Sadness"]	0
"Multisymplectic Spectral Methods for the Gross-Pitaevskii Equation"	["Gross-Pitaevskii equation","Spectral method","Schrödinger equation","Wave equation","Nonlinear Schrödinger equation"]	2
"Speech training systems using lateral shapes of vocal tract and F1-F2 diagram for hearing-impaired children"	["Formant","Vocal tract","Vowel"]	1
"Software Evolution through Transformations."	["Software evolution"]	2

Figure 4.7: Result of query 2

4.2.3. Query 3

This query returns for every paper the list of papers that appear between his citations, but these papers must have at least a FOS in common.
For this particular query we created a graph using the data of a dataset in order to create the relation *CITE* (as represented below).

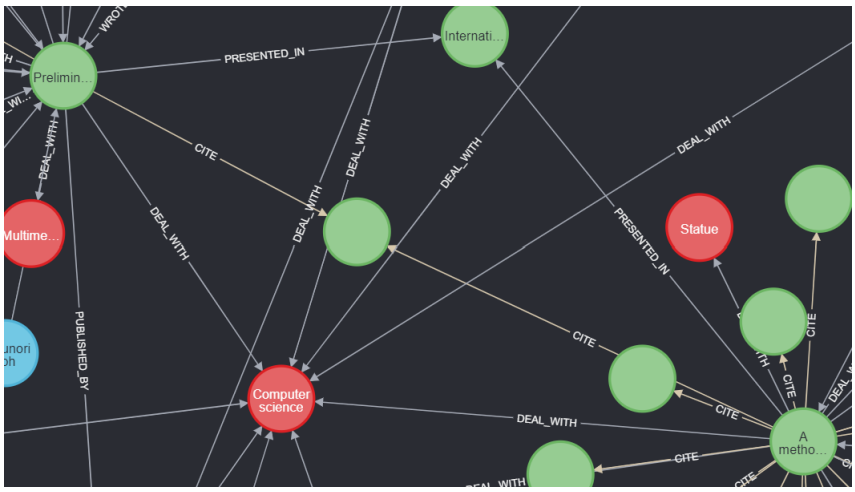


Figure 4.8: Example of the relation CITE

```
1  MATCH (f1:Fos)<-[:DEAL_WITH]-(p1:Paper)-[:CITE]->()<-[:CITE]-(p2:
   Paper)-[:DEAL_WITH]->(f2:Fos)
2  WITH p1.title AS original, collect(p2.title) as cited, f1.id AS
   name1, f2.id AS name2
```

```
3     WHERE name1 = name2
4     RETURN original, cited, name1
```

Listing 4.8: Query 3

"original"	"cited"	"name1"
"A methodology for the physically accurate visualisation of roman polychrome statuary"	["Preliminary Design of a Network Protocol Learning Tool Based on the Comprehension of High School Students: Design by an Empirical Study Using a Simple Mind Map"]	"Computer science"
"Preliminary Design of a Network Protocol Learning Tool Based on the Comprehension of High School Students: Design by an Empirical Study Using a Simple Mind Map"	["A methodology for the physically accurate visualisation of roman polychrome statuary"]	"Computer science"

Figure 4.9: Result of query 3

4.2.4. Query 4

The query 4 returns the papers written before the year 2000 and containing at least 6 citations. These papers must have been presented in a Venue where at least other 3 papers have been presented. In order to execute this query we created a graph "AD HOC" always based on the same dataset we imported.

```
1     MATCH (p1:Paper)-[:PRESENTED_IN]->(v:Venue)<-[r:PRESENTED_IN]-(:
    Paper)-[:DEAL_WITH]->(fos:Fos)
2     WHERE p1.year<2000 and p1.n_citation>5 and fos.id="Machine learning"
3     WITH  p1.title as paper,  v.raw as raw, count(r) as
    other_pap_presented
4     WHERE other_pap_presented>2
5     RETURN paper, raw, other_pap_presented
```

Listing 4.9: Query 4

"paper"	"raw"	"other_pap_presented"
"Cooperative learning over composite search spaces experiences with a multi-agent design system"	"National Conference on Artificial Intelligence"	3
"Search lessons learned from crossword puzzles"	"National Conference on Artificial Intelligence"	3

Figure 4.10: Result of query 4

4.2.5. Query 5

The query returns the list of authors that wrote a paper before the 2000 which was presented in a conference (for which we return the attribute *raw* which is the name of the

conference itself). This conference must have presented at least 5 papers (and returns the number of the corresponding papers, shown in a decreasing order).

```
1  MATCH (a:Author)-[:WROTE]->(p1:Paper)-[r1:PRESENTED_IN]->(v:Venue)
   <-[r2:PRESENTED_IN]-(p2:Paper)
2  WHERE v.type = "C" and p1.year<2000
3  WITH collect(distinct a.name) as authors, v.raw as raw, count(
   distinct r2) AS num_papers
4  WHERE num_papers>5
5  RETURN  raw, num_papers, authors
6  ORDER BY num_papers DESC
```

Listing 4.10: Query 5

"raw"	"num_papers"	"authors"
"Conference of the International Speech Communication Association"	27	["Julian H. Page","Mark Tatham","Andy Tams","Julia Hirschberg","Richard Sproat","David Yarowsky","Jose Maria Elvira","Javier Caminero","Juan Carlos Torrecilla","Mats Ljungqvist","Anders Lindström","Kjell Gustafson","Junichi Yamazaki","Zhiyong Peng","Makoto J. Hirayama","Taro Sugahara","Lisa-Jane Brown","Sandra P. Whiteside","John Locke","Peter Jones","Satoshi Nakamura","Kiyohiro Shikano","Shigeki Sagayama","Hiroaki Hattori","Klaus Beulen","Hermann Ney","Lutz Welling","Annemie Vorstermans","Nick Cremelie","Jean-Pierre Martens","Yoshinori Sagisaka","Nobuyoshi Kaiki","Katsuhiko Mimura","Owen P. Kenny","Douglas J. Nelson","Li-Qun Xu","Tie-Cheng Yu","G. D. Tattersall","Laure Charonnat","Michel Glutton","Joel Crestel","Herve Chuberre","Arun C. Surendran","Chin-Hui Lee","Shuji Doshita","Tatsuya Kawahara","Toru Ogawa","Shigeyoshi Kitazawa","Cheol-Jun Hwang","Hyun-Yeol Chung","Shi-wook Lee"]
"Parallel and Distributed Processing Techniques and Applications"	8	["I. M. Ikram","Paul S. Wang","Hong H. Ong"]
"International Conference on Human-Computer Interaction"	6	["Ann Doubleday","Alistair G. Sutcliffe","Ian Bennett","Michele Ryan","Anthony Savidis","Pier Luigi Emiliani","Constantine Stephanidis"]
"International Joint Conference on Artificial Intelligence"	6	["K. Sparck Jones","B. K. Bog","Minoru Shigenaga","Yoshihiro Sekiguchi","Stephan Busemann","P. Bonzon"]

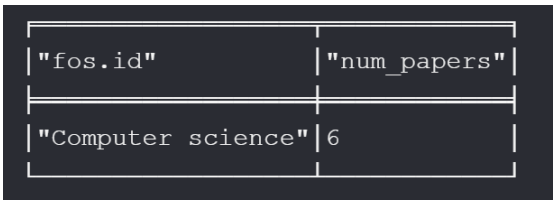
Figure 4.11: Result of query 5

4.2.6. Query 6

The query returns the name of the FOS that has more papers published on the journals and the corresponding number of paper that deal with that field.

```
1  MATCH (fos:Fos)<-[r:DEAL_WITH]-(p:Paper)-[:PRESENTED_IN]->(:Venue {
   type:"J"})
2  WITH fos, count(r) as num_papers
3  ORDER BY num_papers desc
4  LIMIT 1
5  RETURN fos.id, num_papers
```

Listing 4.11: Query 6



"fos.id"	"num_papers"
"Computer science"	6

Figure 4.12: Result of query 6

4.2.7. Query 7

Since the dataset has several different "Springer" publishers from different cities, this query aims to catalog all of them, counting the number of papers they published. Moreover, it retrieves a list of the authors that wrote those articles, created using the function collect(). They are finally shown in decreasing order

```
1  MATCH (pub:Publisher)<-[r1:PUBLISHED_BY]-(p:Paper)<-[:WROTE]-(a:
   Author)
2  WITH pub, count(distinct r1) as num_papers, collect(distinct a.name)
   as authors
3  WHERE substring(pub.id, 0, 8)="Springer"
4  RETURN pub.id, num_papers, authors
5  ORDER BY num_papers desc
```

Listing 4.12: Query 7

"pub.id"	"num_papers"	"authors"
"Springer, Berlin, Heidelberg"	8	["Fencol C. C. Yung","Prudence W. H. Wong","Mihai Burcea","M. Mufakhkharul Islam","Faisal Zaman","Altaf Hossain","M. Nasser","Pranab Kini","José R. Martí","Alberto Tofani","Simone Palmieri","Antonio Di Pietro","Abdullah Alsubaie","Ting Fu Lin","Yutaka Nakamura","Hiroshi Furukawa","C. M. Schober","A. L. Islas","Jill-Jênn Vie","Michel Abdalla","Phan Cong Vinh","Mizue Kayama","Makoto Satoh","Makoto Otani","Michio Shimizu","Masami Hashimoto","Ryo Muramatsu","Masahiko Sugimoto","Kazunori Itoh"]
"Springer, Cham"	2	["Adam Jurczyński","Dominik Szajerman","Avdhesh Garg","Priya Singh","Ankita Brahmachari","Dipankar Das"]
"Springer, London"	1	["Güzin Ulutas","Mustafa Ulutas","Vasif V. Nabiyev"]

Figure 4.13: Result of query 7

4.2.8. Query 8

This query returns the authors that have written at least two different papers published by two different publishers. The authors, limited to 2, are then returned by order of the

total citations of all the papers in which they appears as authors. The result given is performed on an "AD HOC" dataset.

```
MATCH (p1:Publisher)<-[:PUBLISHED_BY]-(:Paper)<-[:WROTE]-(a:Author)
-[:WROTE]->(:Paper)-[:PUBLISHED_BY]->(p2:Publisher)
WHERE p1<>p2
MATCH (a)-[:WROTE]->(p:Paper)
WITH a, SUM(p.n_citation) as numCitations
ORDER BY numCitations DESC
LIMIT 2
RETURN (a)
```

Listing 4.13: Query 8

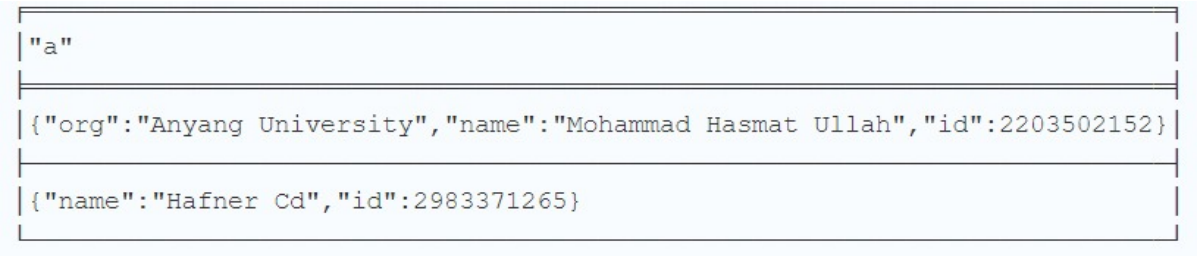


Figure 4.14: Result of query 8

4.2.9. Query 9

This query returns all the papers written by at least two authors (and also return them). One of these authors must have written at least one paper concerning the FOS "Artificial Intelligence" with weight ≥ 0.5, the other author must have written one paper about "Management" with weight ≥ 0.5.

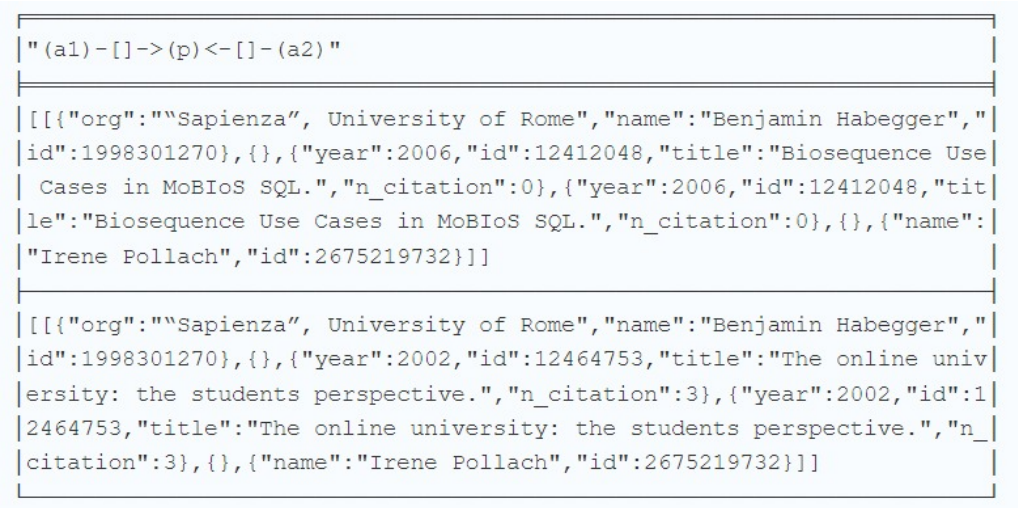
```
MATCH (p1:Paper)-[d1:DEAL_WITH]->(f1:Fos),
(p2:Paper)-[d2:DEAL_WITH]->(f2:Fos),
(a1:Author)-[:WROTE]->(p:Paper)<-[:WROTE]-(a2:Author)
WHERE
(a1)-[:WROTE]->(p1) AND
(a2)-[:WROTE]->(p2) AND
f1.id = "Artificial intelligence" AND f2.id = "Management" AND
d1.weight >= 0.5 AND d2.weight >= 0.5
RETURN (a1)-[]->(p)<-[]-(a2)
```

Listing 4.14: Query 9 less performing version

The following version is more efficient. That's because the above version builds a Cartesian product between disconnected patterns.

```
1     MATCH
2   (:Fos {id: "Management"})<-[d2:DEAL_WITH]-(p2:Paper)<-[:WROTE]-(a1:
     Author)-[:WROTE]->(p:Paper)<-[:WROTE]-(a2:Author)-[:WROTE]->(p1:Paper
     )-[d1:DEAL_WITH]->(:Fos {id: "Artificial intelligence"})
3     WHERE d1.weight>=0.5 AND d2.weight>=0.5
4     RETURN (a1)-[]->(p)<-[]-(a2)
```

Listing 4.15: Query 9 more performing version



```
"(a1)-[]->(p)<-[]-(a2)"
[[{"org":"\"Sapienza", University of Rome","name":"Benjamin Habegger","
id":1998301270},{},{"year":2006,"id":12412048,"title":"Biosequence Use
 Cases in MoBIoS SQL.","n_citation":0},{"year":2006,"id":12412048,"tit
le":"Biosequence Use Cases in MoBIoS SQL.","n_citation":0},{},{"name":
"Irene Pollach","id":2675219732}]]
[[{"org":"\"Sapienza", University of Rome","name":"Benjamin Habegger","
id":1998301270},{},{"year":2002,"id":12464753,"title":"The online univ
ersity: the students perspective.","n_citation":3},{"year":2002,"id":1
2464753,"title":"The online university: the students perspective.","n_
citation":3},{},{"name":"Irene Pollach","id":2675219732}]]
```


Figure 4.15: Result of query 9

4.2.10. Query 10

This query finds the shortest Path from an author whose first name is "Michel" to the topic "Artificial intelligence" with the constraint that the author can't have written any Paper concerning Machine Learning. Since in our database the number of nodes is limited I haven't put any upper bound, but in other cases it would have been necessary in order not to result in very long execution times.

```
1     MATCH (FirstAuthor:Author),
2     (Topic:Fos{id:"Artificial intelligence"}),
3     p=shortestPath((FirstAuthor)-[*]-(Topic))
4     WHERE substring(FirstAuthor.name, 0, 6)="Michel" and not (
    FirstAuthor)-[:WROTE]-()-[:DEAL_WITH]-(:Fos{id:"Machine learning"})
5     RETURN p,FirstAuthor.name
```

Listing 4.16: Query 10

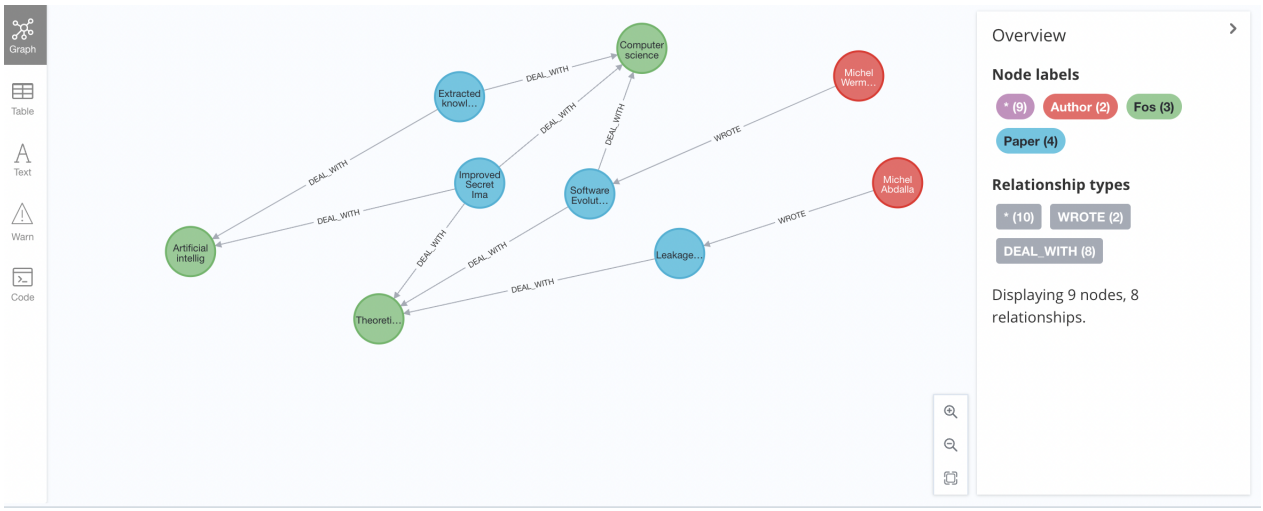


Figure 4.16: Result of query 10

4.3. Performance

Here we can see the performance of the queries described in a particular execution on a local database of 7926 nodes.

4.3.1. Creation/Update queries

Query	Time (ms)
Query 1	7
Query 2	48
Query 3	17
Query 4	13
Query 5	16

Table 4.1: Table of execution times of data creation/upload

4.3.2. Information Retrieval queries

Query	Time (ms)
Query 1	57
Query 2	63
Query 3	126
Query 4	32
Query 5	33
Query 6	21
Query 7	36
Query 8	21
Query 9 version 1	31
Query 9 version 2	25
Query 10	88

Table 4.2: Table of execution times of information retrieval queries

Part II

Delivery 2

5 | Introduction and description of the model

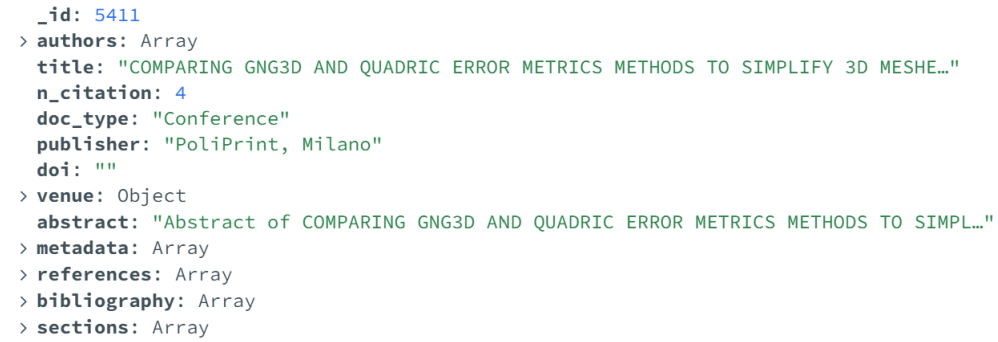


Figure 5.1: Example of a paper seen on MongoDb

Each paper contains information regarding:

- **Title**: contains the title of the paper;
- **Author**: the array *authors* contains all the paper authors and their details such as
  - *name*, which contains the name and surname of the author;
  - *email*, which contains the author's email;
  - *affiliation*, which contains the University affiliated to the author;
  - *bio*, which should contain a brief description of what the author does for a living, as well as their qualifications and education details (in our json file we use the package lorem to create random texts);

```
authors: Array
  0: Object
     name: "Altaf Hossain"
     id: 2300589394
     email: "Altaf.Hossain@mail.com"
     bio: "Bio of Altaf Hossain: ullam ea quo suscipit ex officiis expedita conse…"
     affiliation: "Department of Statistics , Rajshahi University , Rajshahi , Bangladesh"
  1: Object
     name: "Faisal Zaman"
     id: 2308774408
     email: "Faisal.Zaman@mail.com"
     bio: "Bio of Faisal Zaman: labore ullam illum voluptatibus amet repellendus …"
     affiliation: "Department of System Design and Informatics, Kyushu Institute of Techn…"
  2: Object
  3: Object
```


Figure 5.2: Example document - author array

- **Number of citations**: contains the number of papers that are citated in the actual paper (this value can also be checked by counting the number of elements in the array *References*);
- **Doctype**: it explains the kind of publication in which the paper first appears;
- **Publisher**: it indicates the name of the publisher of the paper or the publisher of the journal of the paper;
- **Digital Object Identifier** (doi): it's a digital identifier;
- **Abstract**: it should contain the abstract of the paper, but we set this field using the pattern: "Abstract of" + title of the paper + random words using *lorem.words()*;
- **Venue**: this object describes where the paper was presented and can be either a Conference, a Repository or a Journal (*doc_ type*);


```
venue: Object
   raw: "Pattern Recognition and Machine Intelligence"
   id: 1136274694
   type: "C"
   volume: 3
   number: 17
   date: "2009-03-16"
   pages: 4
```


Figure 5.3: Example document - venue object

- **Metadata**: it contains an array with all the keywords (i.e. the topics discussed) and the corresponding weight (the importance they have in the meaning of the overall document);

- **References**: it's an array that contains all the Ids of the paper citated in the current paper;
- **Bibliography**: it's an array used to complete the *references* array by giving a title to the referenced Id;


```
references: Array
  0: 511601
  1: 672965
  2: 681007
  3: 512839
  4: 671169
venue: Object
abstract: "Abstract of Comparison of GARCH, Neural Network and Support Vector Mac…"
metadata: Array
  0: Object
    keyword: "Autoregressive-moving-average model"
    weight: 0.49811
  1: Object
    keyword: "Computer science"
    weight: 0.41114
  2: Object
  3: Object
  4: Object
  5: Object
  6: Object
  7: Object
  8: Object
bibliography: Array
  0: "An OSF/1 UNIX for Massively Parallel Multicomputers. Id: 511601"
  1: "Using Argumentation to Overcome Hypertext's HCI Failings. Id: 672965"
  2: "Improving User Stereotypes through Machine Learning Techniques Id: 681…"
  3: "Use of spatial adaptation for image rendering based on an extension of…"
  4: "Virtual coiling of intracranial aneurysms based on dynamic path planni…"
```


Figure 5.4: Example document - references, bibliography and metadata arrays

- **Sections** is a multiple array; every element is an object that contains: the field *title*, an array *textbyparagraph* containing the paragraphs of the section (before begininning with the subsections) again created with lorem.words()), another array *subsections* that has a title and an array of paragraphs, and the array *figures* that contains all the urls and captions of the figures of that section.

```
▾ sections: Array
  ▾ 0: Object
      title: "Title of section 1"
    ▾ textbyparagraph: Array
      ▾ 0: Object
          paragraph: "modi rem eos quis voluptates obcaecati saepe necessitatibus ipsum dolo…"
    ▾ figures: Array
      ▾ 0: Object
          url: "https://image.com/?paperid=1688&section=1&image=1"
          caption: "Caption of image 1.0"
    ▾ subsections: Array
      ▾ 0: Object
          title: "Title of subsection 1"
        ▾ textbyparagraph: Array
          ▾ 0: Object
              paragraph: "error asperiores exercitationem repudiandae corporis minus tenetur ut …"
          ▾ 1: Object
              paragraph: "dolorem sapiente possimus quo ullam eveniet minima recusandae inventor…"
      › 1: Object
      › 2: Object
      › 3: Object
  › 1: Object
  › 2: Object
  › 3: Object
```


Figure 5.5: Example document - sections array

6 | Import

For the import phase, we used the same dataset of the first delivery, but this time we had to add some new fields to the structure of a paper. In fact, they were not present in the json file already used. Examples went from the simpler ones, like the absence of the *email* and *bio* for the authors, to the more difficult ones, like the generation from scratch of the *sections*, with their title, text by paragraph, subsections and list of figures.

To cope with these problems we wrote a Python script which reads the file of the original dataset, modifies it and deploys another json file with all the changes required, both in adding new fields and in deleting not interesting/useful ones. In the following sections we will present the issues encountered, together with the solutions we came up with.

6.1. Libraries used

To work on the raw dataset, we imported the following libraries:

- **json** for the methods of read/write operations on a .json file;
- **lorem**[1] to fill with random strings the new fields (wg. the paragraphs of the sections);
- **random** for the functions of picking a random number over an interval of numbers;
- **date** to work with dates, in particular to assign a date to a paper (the year was already provided);
- **urllib** to generate a random URL for the images.

6.2. Abstract and Authors

For the abstract, we simply inserted a new field for each paper and put some words from the *lorem* library.

[1] https://pypi.org/project/python-lorem/

```
data[i]['abstract'] = "Abstract of " + data[i]['title'] + ": " + lorem.
    words(20)
```

In particular, `data[i]` is the array containing the papers extracted from the raw dataset.

As for the authors, we did not have neither the mail nor the bio. For having a more consistent database, we reused the data we have on the author's name and we created a mail like *name.surname@mail.com* using the method `.split()`. Next, for the bio we adopted the same way as for the abstract for the paper. Finally, we assigned an *affiliation* to the authors which did not have it, taken randomly from a vector of affiliations we created (`vect_org[]` in the code).

```
for author in data[i]['authors']:
    author['email'] = author['name'].split()[0] + '.' + \
                      author['name'].split()[len(author['name'].split())
    -1] + '@mail.com'
    author['bio'] = 'Bio of ' + author['name'] + ': ' + lorem.words(20)
    if 'org' in author.keys():
        author['affiliation'] = author['org']
    else:
        author['affiliation'] = vect_org[random.randint(0, len(vect_org)
    - 1)]
```

6.3. Metadata

For the required field *metadata*, we used the *fos* that the dataset already provided, renaming the name of the *fos* in the *keyword* of the metadata.

```
data[i]['metadata'] = []
if 'fos' in data[i]:
    for j in range(len(data[i]['fos'])):
        data[i]['metadata'].append({'keyword': str(data[i]['fos'][j]['
    name']), 'weight': data[i]['fos'][j]['w']})
```

6.4. Publication details

For the publication details, we partially used the field *venue*, assuming that each venue produces a journal, with the name of its raw, in which these papers are collected. For the *date*, we only had the attribute *year* of the paper; anyway, for the same reason of giving a minimum of consistency to data, we extracted a random date in that year. Then, for the integer values of *number*, *volume* and *pages* we assigned a random one. Finally, we set a fixed publisher randomly to some papers that didn't have it.

```
data[i]['venue']['date'] = date.fromordinal(
    random.randint(date(day=1, month=1, year=data[i]['year']).toordinal
    (), date(day=31, month=12, year=data[i]['year']).toordinal())
    ).isoformat()
data[i]['venue']['volume'] = random.randint(1, 3)
data[i]['venue']['number'] = random.randint(1, 20)
data[i]['venue']['pages'] = random.randint(1, 20)
if data[i]['publisher']=="" and i%2 == 0:
    data[i]['publisher']="PoliPrint, Milano"
```

6.5. Sections

As said before, we had nothing about sections in the dataset, so we inserted them after creating from the ground up. A section has been modeled as an array composed of:

- A title;
- An array with the texts of the paragraphs;
- An array with a list of the figures, which are objects consisting in:
  - URL of the image, which has as values passed the id of the paper, the section in which it is present and the number of the image;
  - Caption of the image;
- An array with the subsections, which are objects consisting in:
  - Title of the subsection;
  - Array with the text of the various paragraphs;

```
data[i]['sections'] = []
for j in range(random.randint(1, 5)):
    data[i]['sections'].append({'title': "Title of section " + str(j+1),
                                'textbyparagraph': [],
                                'figures': []})
    data[i]['sections'][j]['subsections'] = []
    for p in range(random.randint(1, 5)):
        data[i]['sections'][j]['subsections'].append({'title': "Title of
    subsection " + str(p+1),
                                                      '
    textbyparagraph': []})
        for k in range(random.randint(1,5)):
            data[i]['sections'][j]['subsections'][p]['textbyparagraph'].
    append({'paragraph': lorem.words(80)})
```

```
12
13      for p in range(random.randint(0,4)):
14          data[i]['sections'][j]['textbyparagraph'].append({'paragraph':
    lorem.words(80)})
15
16      for p in range(random.randint(0, 3)):
17          data[i]['sections'][j]['figures'].append({"url": 'https://image.
    com/?' + urllib.parse.urlencode( {'paperid': data[i]['id'], 'section'
    : j+1, 'image': p+1}),
18          "caption": 'Caption of image ' + str(j+1) + '.' + str(p+1)})
```

6.6. Bibliography

For the *bibligraphy*, we created a *references* array with some random ids of the papers present in the dataset (excluding its own). `vect_id` is the array containing all the ids of the dataset.

```
1 data[i]['references'] = []
2     for j in range(random.randint(1, 6)):
3         number = vect_id[random.randint(0, n_doc-1)]
4         while number in data[i]["references"]:
5                 number = vect_id[random.randint(0, n_doc-1)]
6         data[i]["references"].append(number)
```

7 | Queries in MongoDB

7.1. Data creation/update

Here are the five data creation/update commands.

7.1.1. Query 1

Simple query for the creation of a new document. In this example only some fields are filled for simplicity.

```
db.papers.insertOne({
  "_id" : 1,
  "title" : "New Paper",
  "n_citation" : 0,
  "doc_type" : "Conference",
  "authors": [{
    "name": "Caterina Giardi",
    "id": 1,
    "email": "caterina.giardi@mail.polimi.it",
    "bio": "I'm a Master Degree student at Politecnico di Milano",
    "affiliation": "Politecnico di Milano"
  },
  {
    "name": "Stefano Baroni",
    "id": 2,
    "email": "stefano.baroni@mail.polimi.it",
    "bio": "I'm a Master Degree student at Politecnico di Milano",
    "affiliation": "Politecnico di Milano"
  },
  {
    "name": "Alessia Menozzi",
    "id": 3,
    "email": "alessia.menozzi@mail.polimi.it",
    "bio": "I'm a Master Degree student at Politecnico di Milano",
    "affiliation": "Politecnico di Milano"
  },
```

```
  {
    "name": "Rocco Agnello",
    "id": 4,
    "email": "rocco.agnello@mail.polimi.it",
    "bio": "I'm a Master Degree student at Politecnico di Milano",
    "affiliation": "Politecnico di Milano"
  },
  {
    "name": "Luca Brembilla",
    "id": 5,
    "email": "luca.brembilla@mail.polimi.it",
    "bio": "I'm a Master Degree student at Politecnico di Milano",
    "affiliation": "Politecnico di Milano"
  }],
  "venue" : {
    "raw": "New Conference",
    "id": 1,
    "type": "C",
    "volume": 4,
    "number": 1,
    "date": "2022-11-21",
    "pages": 10
  },
  "abstract": "Abstract of the new paper",
  "references": []
})
```

```
_id: 1
title: "New Paper"
n_citation: 0
doc_type: "Conference"
authors: Array
  0: Object
     name: "Caterina Giardi"
     id: 1
     email: "caterina.giardi@mail.polimi.it"
     bio: "I'm a Master Degree student at Politecnico di Milano"
     affiliation: "Politecnico di Milano"
  1: Object
  2: Object
  3: Object
  4: Object
venue: Object
   raw: "New Conference"
   id: 1
   type: "C"
   volume: 4
   number: 1
   date: "2022-11-21"
   pages: 10
abstract: "Abstract of the new paper"
references: Array
```


Figure 7.1: Paper **after** the execution

7.1.2. Query 2

It is a simple update query in which the publisher of a specific paper, found by id, it's changed in *"PoliPrint, Milano"*.

```
db.project.updateOne(
  {"_id":8373},
  {"$set": {"publisher":"PoliPrint, Milano"}}
)
```

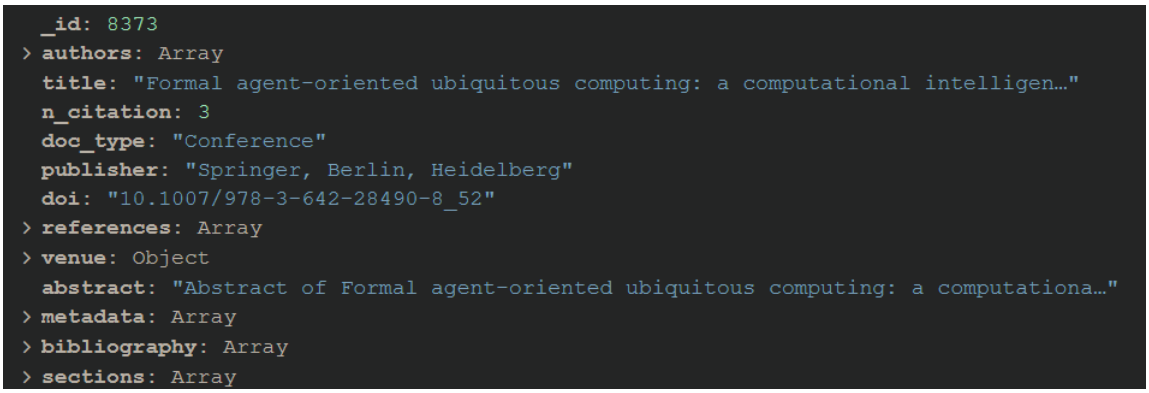


Figure 7.2: Paper **before** the changes

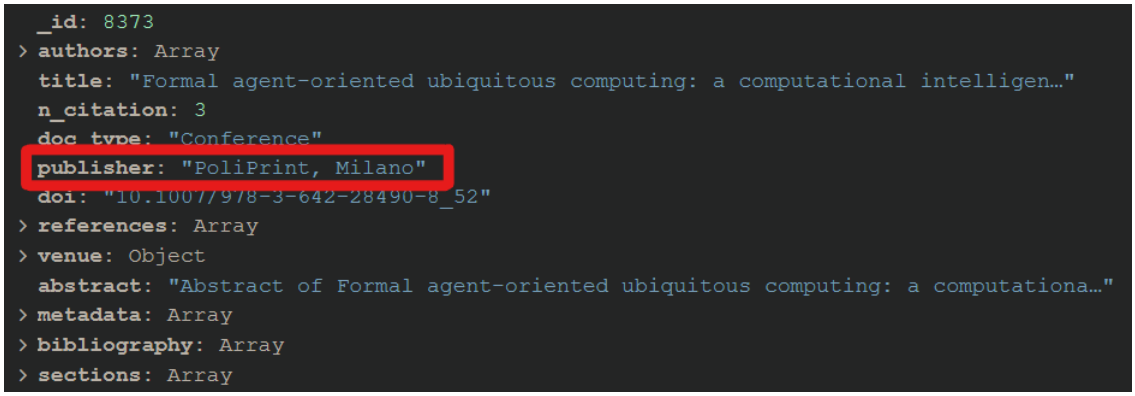


Figure 7.3: Paper **after** the changes

7.1.3. Query 3

In this update query we check for every keyword whose corresponding weight is equal to 0. If found the entry is removed from the metadata array.

```
db.papers.updateMany(
  {
    "metadata.weight": 0
  },
  {
  $pull: {metadata: {weight:0}}
})
```

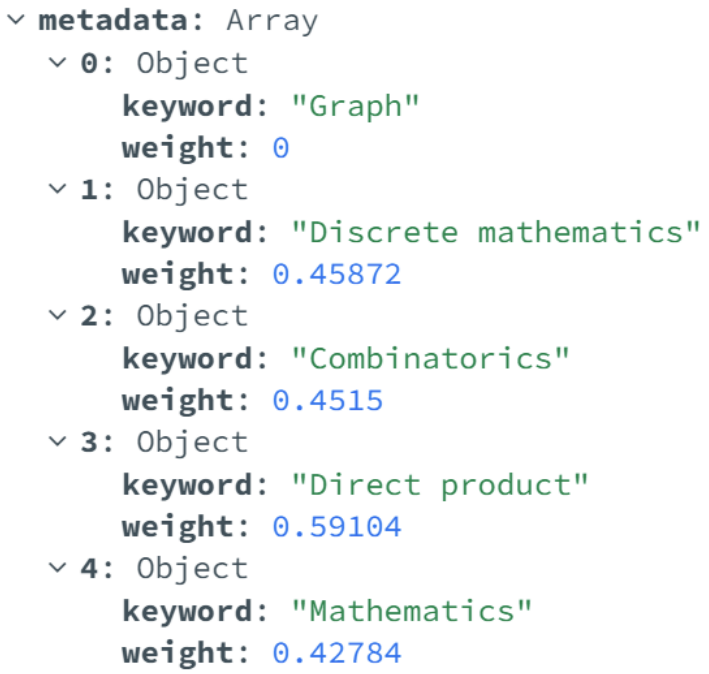


Figure 7.4: Before executing query 3

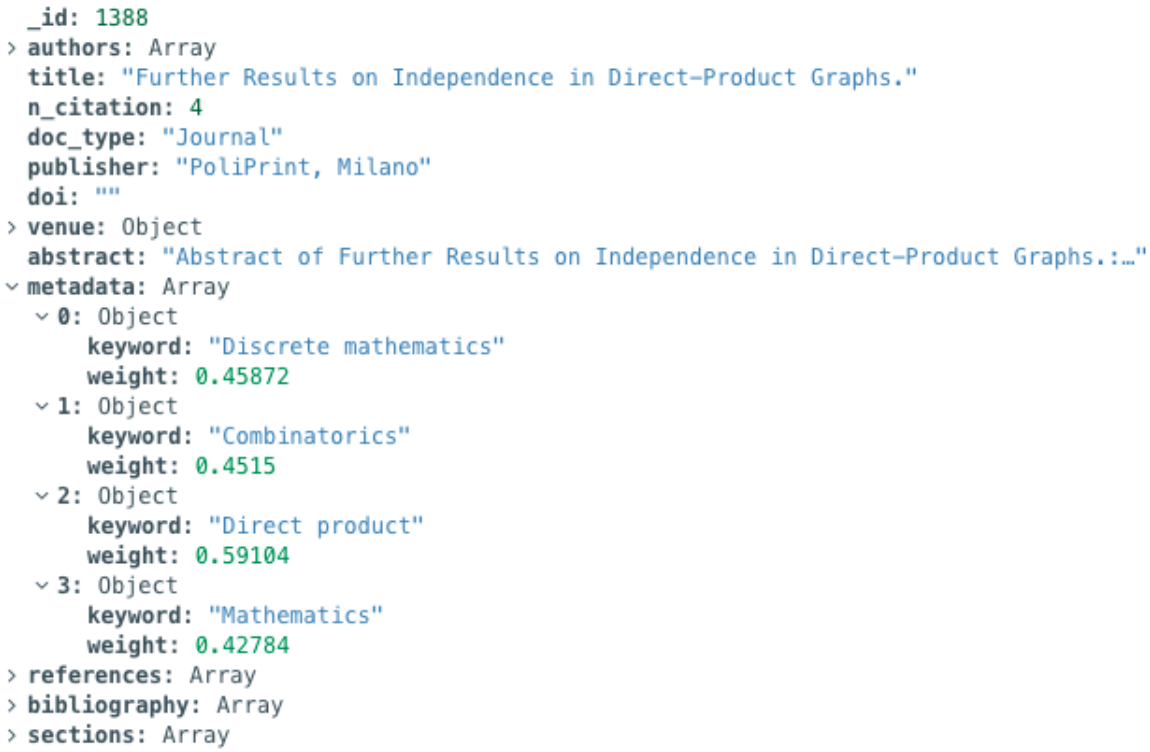


Figure 7.5: Result of Query 3

7.1.4. Query 4

The following update query removes the field doi when it is empty.

```
db.papers.updateMany({
  "doi":""
},
{
    $unset: {doi:""}
})
```

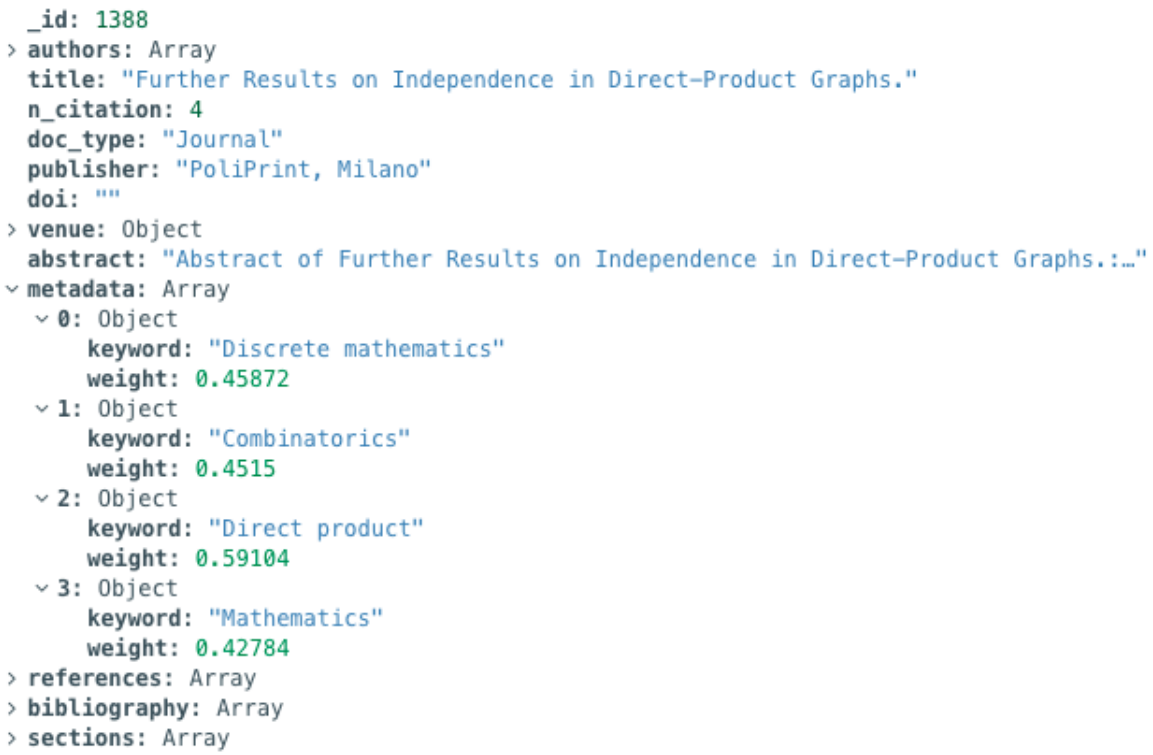


Figure 7.6: Before executing update query 4

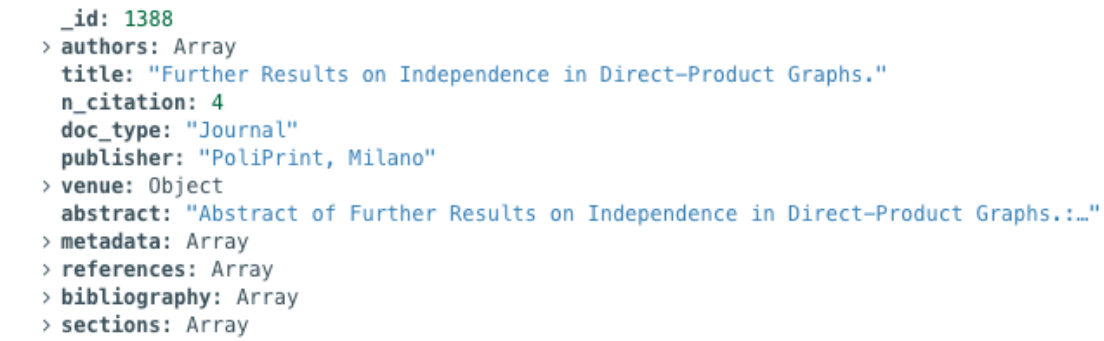


Figure 7.7: Result of update query 4

7.1.5. Query 5

The last query is an example of delete. The query removes every paper that has a number of citation equals to 1.

```
db.papers.deleteMany({
  "n_citation": 1
})
```

	_id Int32	authors Array	title String	n_citation Int32	doc_type String
1	1091	[] 8 elements	"Preliminary Design of a Ne…	6	"Conference"
2	1388	[] 1 elements	"Further Results on Indepen…	4	"Journal"
3	1674	[] 2 elements	"A methodology for the phys…	1	"Conference"
4	1688	[] 4 elements	"Comparison of GARCH, Neura…	5	"Conference"
5	5411	[] 4 elements	"COMPARING GNG3D AND QUADRI…	4	"Conference"
6	5781	[] 2 elements	"Vectorial fast correlation…	4	"Repository"
7	6522	[] 3 elements	"Improved Secret Image Shar…	6	"Conference"
8	6762	[] 2 elements	"A Self-Stabilizing Algorit…	4	"Conference"
9	8373	[] 1 elements	"Formal agent-oriented ubiq…	3	"Conference"
10	8763	[] 2 elements	"Fur Visualisation for Comp…	1	"Conference"
11	382732	[] 2 elements	"Bridging the Gap Between S…	4	"Conference"
12	385572	[] 3 elements	"A Generic Framework Based …	3	"Conference"
13	385606	[] 3 elements	"Assessing the State of Too…	2	"Journal"

Figure 7.8: Before executing update query 5

	_id Int32	authors Array	title String	n_citation Int32	doc_type String
1	1091	[] 8 elements	"Preliminary Design of a Ne…	6	"Conference"
2	1388	[] 1 elements	"Further Results on Indepen…	4	"Journal"
3	1688	[] 4 elements	"Comparison of GARCH, Neura…	5	"Conference"
4	5411	[] 4 elements	"COMPARING GNG3D AND QUADRI…	4	"Conference"
5	5781	[] 2 elements	"Vectorial fast correlation…	4	"Repository"
6	6522	[] 3 elements	"Improved Secret Image Shar…	6	"Conference"
7	6762	[] 2 elements	"A Self-Stabilizing Algorit…	4	"Conference"
8	8373	[] 1 elements	"Formal agent-oriented ubiq…	3	"Conference"
9	382732	[] 2 elements	"Bridging the Gap Between S…	4	"Conference"
10	385572	[] 3 elements	"A Generic Framework Based …	3	"Conference"
11	385606	[] 3 elements	"Assessing the State of Too…	2	"Journal"
12	385951	[] 3 elements	"A bimodal Korean address e…	3	"Conference"
13	385984	[] 3 elements	"The Encourage Operators to…	2	"Conference"

Figure 7.9: Result of update query 5

7.2. Queries

7.2.1. Filtering conditions

Query 1

This query matches two conditions that have to be both satisfied: having 564427 between the array of referenced IDs and having at least 5 sections. We show the title, the references array and the titles of the sections (to show that section 5 is present).

```
db.papers.find({
  $and: [{references: { $in: [564427]}}, { "sections.4": {$exists: true
   }}]
}, {
  "title":1,
  "references":1,
  "sections.title":1
})
```

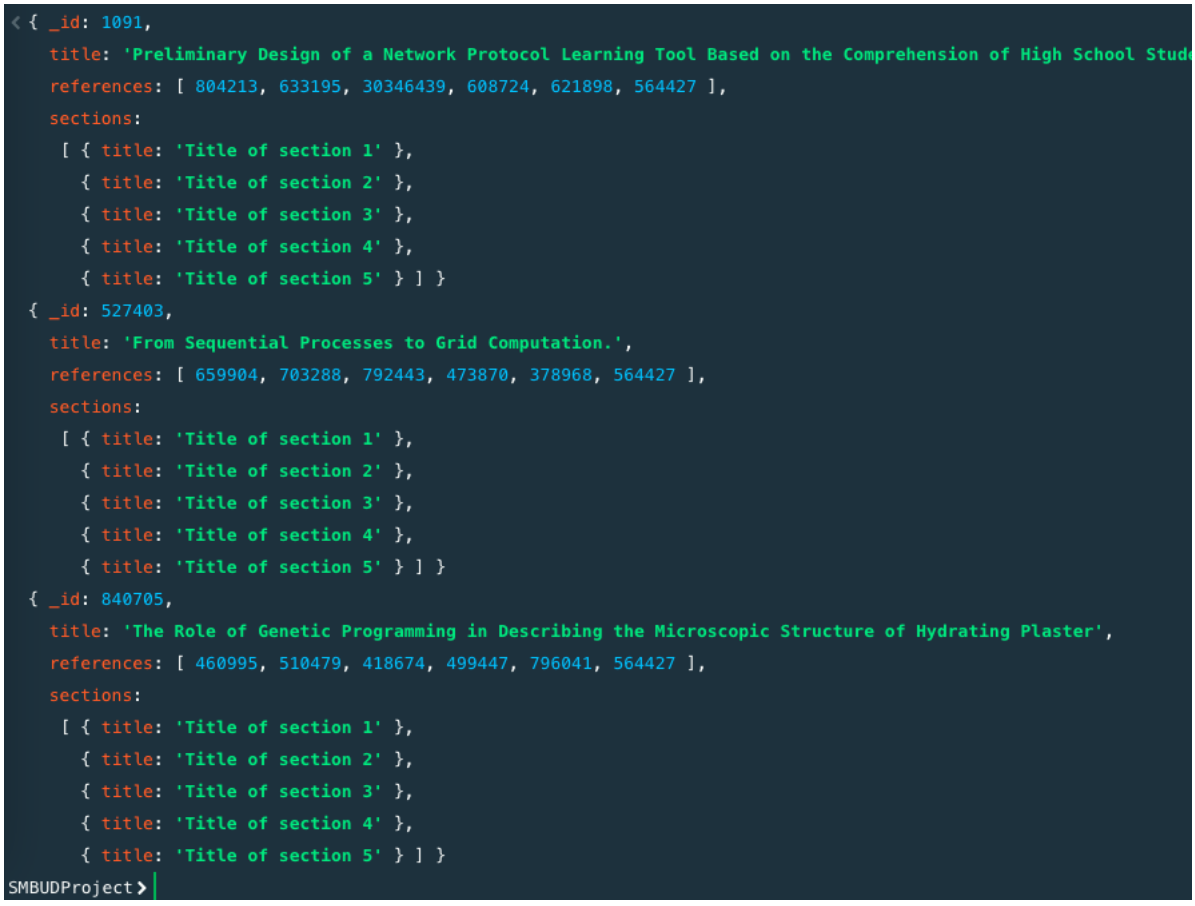


Figure 7.10: Result of query 1

Query 2

The following query filters the documents by checking if they have more than 18 pages and that the second section has at least 5 subsections. Once a document is found, the

query returns the title of the paper, the title of the fifth subsection and the number of pages.

```
db.papers.find({
  "venue.pages": {$gt: 18},
  "sections.1.subsections.4": {$exists: true}
},
{
  "title": 1,
  "sections[1].subsections[4].title": 1,
  "venue.pages": 1
})
```

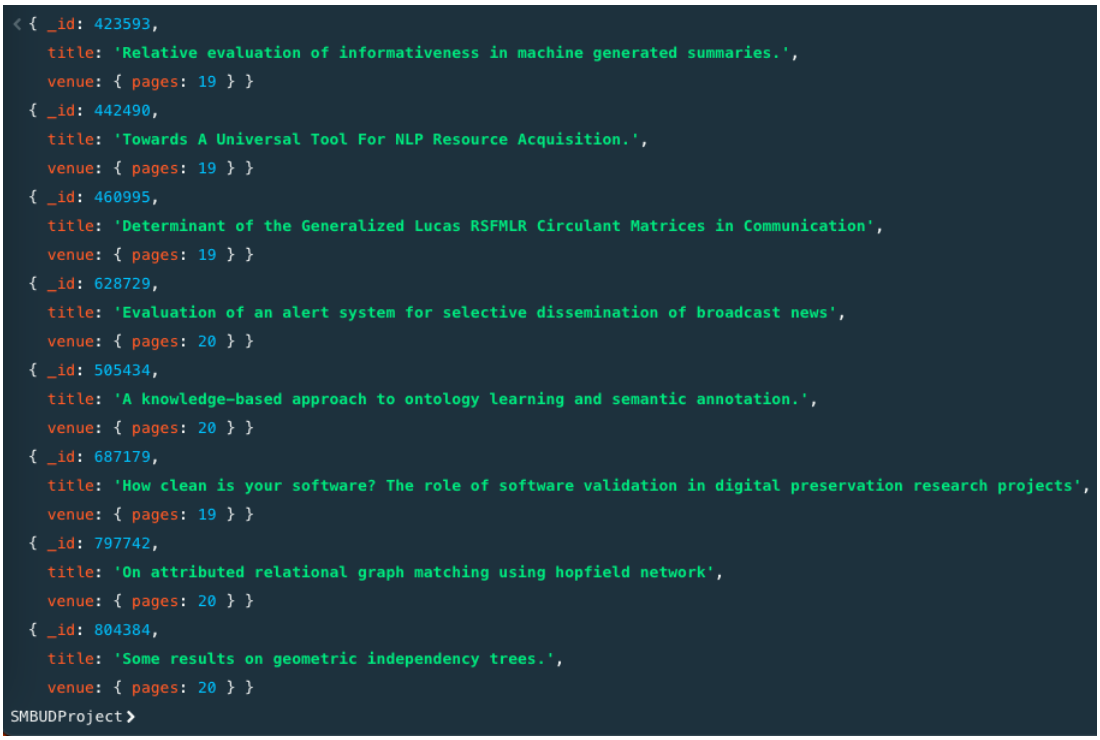


Figure 7.11: Result of query 2

7.2.2. Filtering conditions and aggregation

Query 1

Retrieve the maximum top 5 publishers per amount of citations of their most cited paper. The filter is applied on publisher, which has to exist and not to be equal to the empty string.

```
db.papers.aggregate([
  {
```

```
  /*We make sure that the publisher field is not empty and exists*/
    "$match": {"publisher" : {"$exists" : true, "$ne" : ""}}
  },
  {
  /*We group by the publisher and we record the biggest number of
   citations in one of the papers the publisher appears in*/
    "$group": {
      "_id": "$publisher",
      "maxNumOfCitations": { "$max": "$n_citation" }
    }
  },
  /*Descending order*/
  { "$sort": { "maxNumOfCitations": -1 } },
  /* Showing top 5*/
  { "$limit": 5 }
])
```

```
{ _id: 'Springer', maxNumOfCitations: 6 }
{ _id: 'Springer, Berlin, Heidelberg', maxNumOfCitations: 6 }
{ _id: 'American Medical Informatics Association',
  maxNumOfCitations: 6 }
{ _id: 'University of California, Los Angeles',
  maxNumOfCitations: 6 }
{ _id: 'Rinton Press, Incorporated', maxNumOfCitations: 6 }
```

Figure 7.12: Result of the query 1 with aggregation

Query 2

Retrieve the average authors for each publisher (the publisher must exist and not be the empty string) of a document of type *Conference*, then sort the results in descending order.

```
db.papers.aggregate([
/*Matching conditions*/
  { "$match":
            {"$and":
                [{ "doc_type": "Conference" },{"publisher":{"$exists" :
   true, "$ne" : ""}}]}},
/*Group by publisher and we compute the average number of authors*/
  { "$group":
            { "_id": "$publisher",
```

```
                "avgAuthorPerPublisher": {"$avg": { $size: "$authors"
    }}}},
/*Descending order*/
  { "$sort": { "avgAuthorPerPublisher": -1 }}])
```

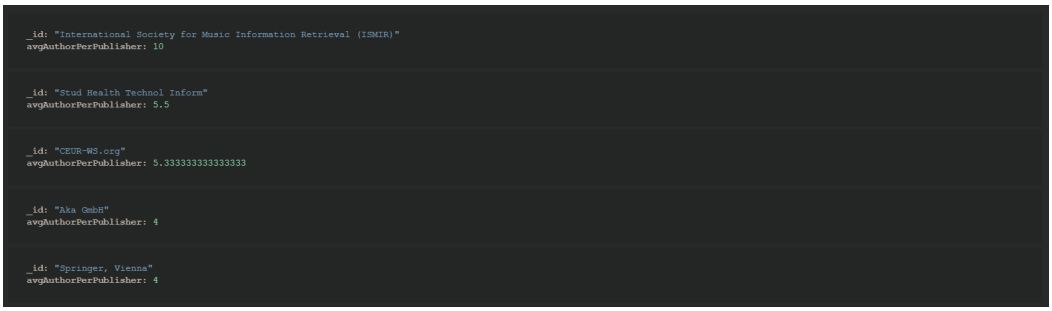


Figure 7.13: Top 5 results retrieved

7.2.3. Filtering conditions also on internal sub-documents

Query 1

Retrieve the papers written by "Pranay Chaudhuri" (at least, it may have other authors), they need to include the keyword "Computer science" and in the title they need to have the string "Algorithm". It returns only the title, the authors' name and the metadata of each document.

```
db.papers.find({
/*Filters the three conditions explained above */
  "$and": [
    {
      "authors": {
        "$elemMatch": {"name":"Pranay Chaudhuri"}
      }
    },
    {
      "metadata": {
        "$elemMatch": {"keyword": "Computer science"}
      }
    },
    {
    "title": { $regex : "Algorithm" }
    }
  ]
},
/*Display the fields*/
```

```
{
  "title": 1,
  "authors.name": 1,
  "metadata": 1
})
```

```
{ _id: 6762,
  authors: [ { name: 'Pranay Chaudhuri' }, { name: 'Hussein Thompson' } ],
  title: 'A Self-Stabilizing Algorithm for Finding the Cutting Center of a Tree.',
  metadata:
   [ { keyword: 'Computer science', weight: '0.44568' },
     { keyword: 'Parallel computing', weight: '0.46131' } ] }
```

Figure 7.14: A document retrieved by the query 1 with filters on sub-documents

Query 2

Retrieve the papers which have exactly 2 authors, the date of the venue in which the paper was presented is 2014 and in the metadata we have the keyword "Computer Science" with a weight greater or equal than 0.40. The query doesn't retrieve all the document but also some properties (i.e. title, authors, etc.).

```
db.papers.find({
/*Filters the three conditions explained above */
  "$and": [
    {
      "authors": { "$size": 2 }
    },
    {
      "metadata": {
        "$elemMatch" : {
          "keyword": {$regex: "Artificial intelligence"},
          "weight": { "$gte": 0.40 }
        }
      }
    },
    {
      "venue.date" : { "$regex" : "2014" }
    }
  ]},
  {
/*Display the fields*/
    "title": 1,
   "authors.name": 1,
   "venue.raw": 1,
```

```
    "venue.date": 1,
    "n_citation": 1,
    "metadata": 1
  }
).sort({"n_citations" : -1}).limit(3)
```

```
{ _id: 1674,
  authors: [ { name: 'G. Beale' }, { name: 'G. Earl' } ],
  title: 'A methodology for the physically accurate visualisation of roman polychrome statuary',
  n_citation: 6,
  venue:
   { raw: 'International Conference on Virtual Reality',
     date: '2014-04-12' },
  metadata:
   [ { keyword: 'Statue', weight: 0.40216 },
     { keyword: 'Engineering drawing', weight: 0.43427 },
     { keyword: 'Virtual reconstruction', weight: 0 },
     { keyword: 'Computer science', weight: 0.42062 },
     { keyword: 'Visualization', weight: 0.4595 },
     { keyword: 'Polychrome', weight: 0.4474 },
     { keyword: 'Artificial intelligence', weight: 0.40496 } ] }
{ _id: 840705,
  authors:
   [ { name: 'Judith Ellen Devaney' },
     { name: 'John G. Hagedorn' } ],
  title: 'The Role of Genetic Programming in Describing the Microscopic Structure of Hydrating Plaster',
  n_citation: 3,
  venue:
   { raw: 'Genetic and Evolutionary Computation Conference',
     date: '2014-05-17' },
  metadata:
```

Figure 7.15: Result of query 2 with filters on sub-documents

7.2.4. Filtering conditions through unwind

Query 1

This query first selects only the papers whose type is *Journal* and then, through **$unwind** it shows the top 10 keywords that are the most used (appear in the *metadata* array) in these kind of papers.

```
db.papers.aggregate([
{
/*We filter the doc_type*/
    $match: { "doc_type": "Journal" }
},
{
/*We unwind metadata array*/
```

```
    $unwind: { path: "$metadata" }
},
{
/*We count for each keyword the number of times it appears inside a
   paper*/
    $group: {
        _id: "$metadata.keyword",
        count: { $sum: 1 }
    }
},
{
/*Descending order*/
    $sort: { count: -1 }
},
{
/*Showing the top 10*/
    $limit: 10
}
])
```

```
{ _id: 'Computer science', count: 44 }
{ _id: 'Mathematics', count: 20 }
{ _id: 'Artificial intelligence', count: 12 }
{ _id: 'Discrete mathematics', count: 11 }
{ _id: 'Knowledge management', count: 10 }
{ _id: 'Mathematical optimization', count: 6 }
{ _id: 'Software engineering', count: 6 }
{ _id: 'Algorithm', count: 6 }
{ _id: 'Distributed computing', count: 6 }
{ _id: 'Combinatorics', count: 5 }
```

Figure 7.16: Result of filtering using unwind 1

Query 2

This query find the percentages of the presence inside the whole documents of the different types of papers (*Repository*, *Journal*, *Conference*) or when it's left blank.

```
db.papers.aggregate([
{
/*We group and count for each type the number of corresponding papers*/
 $group: {
```

```
    _id: '$doc_type',
    counttypes: {
     $sum: 1
    }
   }
},{
/*We dinamically count the total number of papers and we push into an
    array the name of the type and the corresponding quantity*/
   $group: {
    _id: null,
    total: {
     $sum: '$counttypes'
    },
    Typequantity: {
     $push: {
      name: '$_id',
      npapers: '$counttypes'
     }
    }
   }
},{
/*We unwind the array of types also adding an index field*/
   $unwind: {
    path: '$Typequantity',
    includeArrayIndex: 'IndexType',
    preserveNullAndEmptyArrays: false
   }
},{
/*We calculate the percentage and we project the result*/
   $project: {
    _id: '$Typequantity.name',
    percentage: {
     $multiply: [
      {
       $divide: [
        '$Typequantity.npapers',
        '$total'
       ]
      },
      100
     ]
    }
   }
}
```

```
48 ]
49 )
```

```
_id: "Journal"
percentage: 14.6
```

```
_id: "Repository"
percentage: 0.4
```

```
_id: ""
percentage: 20
```

```
_id: "Conference"
percentage: 65
```

Figure 7.17: Result of unwind query 2

7.2.5. Filtering condition through refs

Query 1

It retrieves, for each paper which has at least three references, the percentage of them which were published by *"PoliPrint, Milano"*. The results are shown in decreasing order with respect to the percentage.

It is a pipeline of several operators; the first **$match** is useful to find all the papers with at least four references and it is done as first operation in order to have less document on which do the **$lookup**. After the **$lookup**, it finds the referenced papers with that publisher and soon after, through the first **$project** it counts the total number of the references and the ones with the publisher required. Then, through another **$project** it makes the division between them and multiplying it by 100 we obtain the percentages and finally we sort the results.

```
db.papers.aggregate([
/*Matches the papers with at least 3 references*/
  {"$match": {"references.3": {"$exists": true}}},
/*Perform the lookup on the same collection papers */
  {"$lookup": {
    from: "papers",
    localField: "references",
    foreignField: "_id",
    as: "refs"
  }},
/*Matches the papers whose publisher is PoliPrint, Milano*/
  {"$match": {"refs.publisher": "PoliPrint, Milano"}},
  {"$project": {
/*Compute the total number of references(n_total_references) and the
    total references with the publisher matched before(n_refs_poliprint)
    */
    "_id": 1,
    "n_total_references": {"$size": "$refs"},
    "n_refs_poliprint": {"$size": {"$filter":{input: "$refs", cond: {"
    $eq": ["$$this.publisher","PoliPrint, Milano"]}}}}},
  {"$project": {
/*Compute the percentages through a project*/
    "_id": 1,
    "percentage_of_poliprint_refs": {
      "$trunc":[ {"$multiply":[ {"$divide": ["$n_refs_poliprint","
    $n_total_references"]}, 100]}, 2]}}},
/*Descending order*/
  {"$sort": {"percentage_of_poliprint_refs": -1}}
])
```

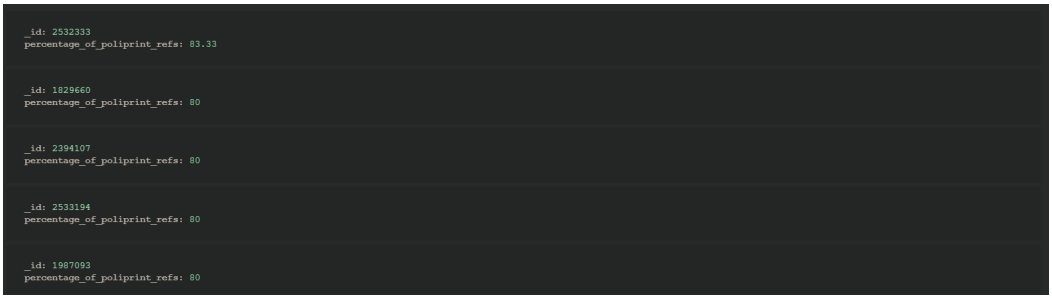


Figure 7.18: Top 5 results retrieved

Query 2

For this query we are using a lookup on two different collections. Since we decided at the beginning to create only one collection for the entire db, we manually created a json called *country* that contains the fields : *country*, *affiliation* and *continent*. We took some of the affiliations and added the country and the continent in order to performs aggregations. So this query performs the **$lookup** between the collection paper and Country and returns, for each country listed in country.json, the list of the authors whose affiliation belongs to that country and the title of the paper they wrote.

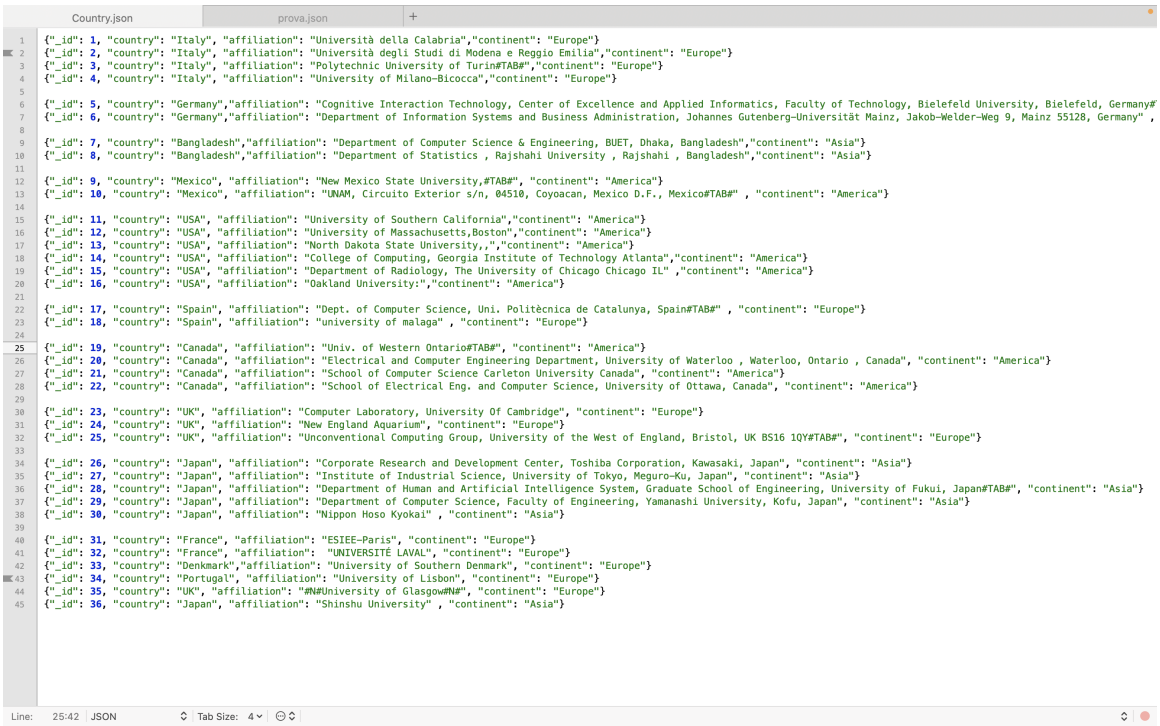


Figure 7.19: Screenshot of the Country.json file created

```
db.papers.aggregate([{
 /* needed for the lookup because authors is an array*/
 $unwind: {
  path: '$authors'
 }
}, {
/*lookup between the two collections based on the field affiliation*/
 $lookup: {
  from: 'Country',
  localField: 'authors.affiliation',
  foreignField: 'affiliation',
  as: 'affiliationDescription'
```

```
 }
}, {
/*Since not all the affiliations are present in the json document, I
    only consider the papers whose authors  found a match*/
 $match: {
  'affiliationDescription.0': {
   $exists: true
  }
 }
}, {
/*Group by country and push into an array the informations we need to
    project*/
 $group: {
  _id: '$affiliationDescription.country',
  authorsnames: {
   $push: {
    name: '$authors.name',
    affiliation: '$authors.affiliation',
    PaperTitle: '$title'
   }
  }
 }
}, {
 $project: {
  _id: 1,
  authorsnames: 1
 }
}])
```

Figure 7.20: Result of lookup query 2

7.3. Performance of the queries

Here's are the performance for a particular execution of the queries on a database hosted on the local machine (obviously times change if repeated, but we can see here how they differ in a simple way). The database contains 1000 documents.

Query	Time (ms)
Filtering conditions - query 1	3
Filtering conditions - query 2	8
Filtering conditions and aggregation - query 1	19
Filtering conditions and aggregation - query 2	10
Filtering conditions on sub-documents - query 1	3
Filtering conditions on sub-documents - query 2	10
Filtering conditions through unwind - query 1	9
Filtering conditions through unwind - query 2	11
Filtering condition through refs - query 1	174
Filtering condition through refs - query 2	605

Table 7.1: Table of execution times of the queries

7.3.1. A closer look at performance with pipelines

Let's take a closer look at performance for the query 7.2.2. We can write the same query in other two different way, using two stages for the filters, instead of using one with a logical and. So, the first way can be:

```
db.papers.aggregate([
  { "$match": { "publisher" : {"$exists" : true, "$ne" : ""} } },
  { "$match": { "doc_type": "Conference" } },
  { "$group": {
    "_id": "$publisher",
    "avgAuthorPerPublisher": {
      "$avg": { $size: "$authors" } }
    }
  },
  { "$sort": { "avgAuthorPerPublisher": -1 } }
])
```

and the second one:

```
db.papers.aggregate([
  { "$match": { "doc_type": "Conference" } },
  { "$match": { "publisher" : {"$exists" : true, "$ne" : ""} } },
  { "$group": {
    "_id": "$publisher",
    "avgAuthorPerPublisher": {
      "$avg": { $size: "$authors" } }
    }
```

```
  },
  { "$sort": { "avgAuthorPerPublisher": -1 } }
])
```

Reminding aggregations are pipelines, i.e. operations included in the query are performed sequentially, we expect different performance. For instance, for the second query, if we exclude nearly all the documents matching only the papers that are presented in a conference, we have then to apply only to these papers the filter on the publisher (a paper can be presented in a conference even without a specified publisher).
The execution times of the first query for three different executions are 8, 10 and 13. For the second one we get 11, 9 and 7. So the second one is more efficient (31>27).
To see the reason, we can see how many papers there are with the filters. For the Conference filter we get

```
db.papers.find({ "doc_type": "Conference" }).count()
```

equals to 613 out of 1000, while we get

```
db.papers.find({ "publisher" : {"$exists" : true, "$ne" : ""} }).count()
```

equals to 759, so the first filter is more restrictive. Thus, the query executing it firstly is more efficient (the second query seen above).

Another aspect that we can highlight is the time for the queries that performs the lookup operation. That operation is like a join for the relational databases, so it's obviously slower than the others. That's also the reason why in MongoDB are used sub-documents.