



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Alessio Buda (10675063)**

Leonardo Cesani (10674905)

Fausto Lasca (10661818)

Gabriele Munafò (10654531)

Matteo Paraboschi (10681473)

Group Number: **20**

Academic Year: 2022-2023

Contents

Contents	i
1 Problem specification	1
1.1 Description	1
1.2 Assumptions	1
2 ER model	3
3 Neo4j	7
3.1 Creation of the database	7
3.1.1 Data set	7
3.1.2 Data importation in Neo4j	7
3.1.3 First schema	8
3.1.4 Differences with the ER diagram	10
3.2 Queries	11
3.2.1 Creation and update queries	11
3.2.2 Request queries	15
4 MongoDB	23
4.1 Document structure	23
4.1.1 Example of a document	24
4.1.2 Differences with the ER model	25
4.2 Data upload and transformation	25
4.2.1 Creation of the dataset	25
4.2.2 Data modification	26
4.2.3 Data upload	26
4.3 Queries	27
4.3.1 Data creation and update	27

4.3.2	Data manipulation	32
A	Appendix A	45
A.1	Article type 1	45
A.2	Article type 2	48
A.3	Article type 3	51
B	Appendix B	55
B.1	Python code	55
List of Figures		63
List of Tables		67

1 | Problem specification

1.1. Description

The aim of the project is to implement a database storage solution for a large-scale dataset recording scientific publications. The database contains various types of scientific publications, including journal articles, books, thesis, conference and workshop papers. Each publication is stored together with important information regarding the publication itself (e.g., its author(s), title, year of publication ...).

The system should allow users to search for specific publications or to find publications based on some of their characteristics.

1.2. Assumptions

The assumptions considered are the following:

- Every conference edition is a separate entity, since a conference can be held after variable time spans. For journals however a journal entity is needed because it is a stable institution, that publishes the journal edition at a fixed pace.
- The author cannot be affiliated to no organization, and he can be affiliated to at most one organization. That means that there is not a record of all organizations where the author worked during his life.
- The names of organizations, publishers and schools are considered to be unique.

2 | ER model

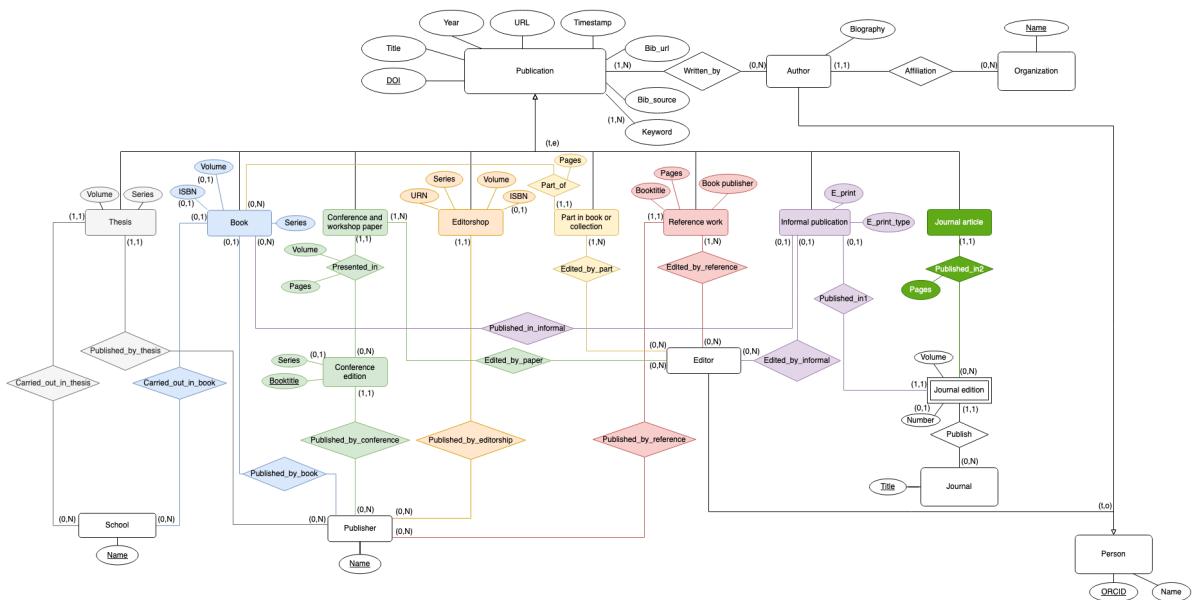


Figure 2.1: ER diagram.

Note: The relationships should be read as follows: a publication is written by at least one author and at most N authors. An author may have written from 0 to N publications.

The ER model presents the following entities:

- **Publication**: represents a generic scientific publication. It is identified by DOI (primary key), the title, the year of publication, the URL of the publication, the timestamp, the Bib URL, the Bib source and potentially multiple keywords associated to the publication. Publications are divided into the subgenres:
 - **Book**: specifically represents a book. Apart from the publication attributes, it is also characterized by an optional ISBN, an optional volume and a series. A book can be carried out in a school (**Carried_out_in_book**) and it is published by a publisher (**Published _by _book**).

- **Journal article:** specifically represents a journal article. It is published in a journal edition (`Published _in2`), specifically in a certain interval of pages.
- **Conference and workshop paper:** specifically represents a conference or workshop paper. It is presented in a conference edition (`Presented _in`), specifically in a certain volume and in a certain interval of pages, and it is edited by an editor (`Edited _by _paper`).
- **Part in book or collection:** specifically represents a part of a book or collection. It is part of a book (`Part _of`), specifically it is a certain interval of pages of that book, and it is edited by an editor (`Edited _by _part`).
- **Editorship:** specifically represents an editorship. Apart from the publication attributes, it is also characterized by a series, a URN, a volume and an optional ISBN. An editorship is published by a publisher (`Published _by _editorship`).
- **Reference work:** specifically represents a reference work. Apart from the publication attributes, it is also characterized by a book title which it refers to, a certain interval of pages and the book publisher. A reference work is published by a publisher (`Published _by _reference`) and it is edited by an editor (`Edited _by _reference`).
- **Informal publication:** specifically represents an informal publication. Apart from the publication attributes, it is also characterized by the E print and the E print type. It can also be published in multiple books (`Published _in _informal`), it can be edited by an editor (`Edited _by _informal`) and it can be published in a journal edition (`Published _in1`).
- **Author:** represents the author who wrote the publication (`Written _by`). It is a type of person. Apart from the person attributes, he is characterized by a biography. He is affiliated to an organization (`Affiliation`).
- **Organization:** represents the organization to which the author is affiliated (`Affiliation`). It is identified by a name (primary key), which is considered to be unique.
- **Journal:** represents a journal. It is identified by a title (primary key), which is considered to be unique. Since a journal can publish different editions (`Publish`) of the journal, this element is represented with a weak entity:
 - **Journal edition:** represents a single journal edition, characterized by a volume and an optional number. In a journal edition, a journal article (`Published`

_in2) or an informal publication (Published _in1) are published.

- **Conference edition:** the conference edition where a conference and workshop paper are presented (Presented _in). It is identified by a book title (primary key) and an optional series. A conference edition is published by a publisher (Published _by _conference).
- **Editor:** represents the editor of a conference and workshop paper (Edited _by _paper), part in book or collection (Edited _by _part), reference work (Edited _by _reference) or informal publication (Edited _by _informal). An editor is a type of person.
- **Publisher:** represents the publisher of a book (Published _by _book), thesis (Published _by _thesis), conference edition (Published _by _conference), editorship (Published _by _editorship) or reference work (Published _by _reference). It is identified by a name (primary key), which is considered to be unique.
- **School:** represent a school where a thesis (Carried _out _in _thesis) or a book (Carried _out _in _book) have been carried out in. It is characterized by a name (primary key) which is considered to be unique
- **Person:** represents a generic person, who can be an editor, an author or both of them (overlapping). It is characterized by a ORCID (primary key) and a name.

3 | Neo4j

3.1. Creation of the database

3.1.1. Data set

The data has been taken from the dblp database provided by the organization as a XML file and a DLD file, which formally explains the XML's structure. Since the original XML contained a huge number of records, it has been divided into smaller files and reconstructed by taking some pieces in order to have the largest variety of data possible. The resulting subset contains roughly 113000 nodes.

The detailed description of the data set, provided by the dblp organization can be found at <https://dblp.org/xml/docu/dblpxml.pdf>.

3.1.2. Data importation in Neo4j

The XML file has been translated into a CSV file, an useful format especially for data importation in Neo4j. It has been done by using a dedicated python script built specifically to translate the dblp XML database (link: <https://github.com/ThomHurks/dblp-to-csv>). By using the DLD file, this script is able to split the data in different CSV files, one for each entity. This way, the data gets separated from the headers, making easier to import and create sets of files representing the relationships. The script also assigns a unique ID to each record, in order to recognize the elements in the files and have a feasible relation scheme between entities.

The usage of the python script in order to translate the files format makes the import of data in Neo4j a lot easier to perform, by only generating the following command:

```
neo4j-admin import --database=dblp --delimiter ";" --array-delimiter "|" --id-type INTEGER  
--nodes=Publication:Book="output_book_header.csv,output_book.csv"  
--nodes=Publication:Thesis="output_phdthesis_header.csv,output_phdthesis.csv"  
--nodes=Publication:Thesis="output_mastersthesis_header.csv,output_mastersthesis.csv"  
--nodes=Publication:Editorship="output_proceedings_header.csv,output_proceedings.csv"  
--nodes=Publication:Article="output_article_header.csv,output_article.csv"  
--nodes=Publication:Book_part="output_incollection_header.csv,output_incollection.csv"  
--nodes=Publication:Conference_paper="output_inproceedings_header.csv,output_inproceedings.csv"  
--nodes=Publication="output_www_header.csv,output_www.csv"  
--nodes=Publisher="output_publisher.csv"  
--relationships=published_by="output_publisher_published_by.csv"  
--nodes=Person:Editor="output_editor.csv"  
--relationships=edited_by="output_editor_edited_by.csv"  
--nodes=Journal="output_journal.csv"  
--relationships=published_in="output_journal_published_in.csv"  
--nodes=Person:Author="output_author.csv"  
--relationships=written_by="output_author_written_by.csv"  
--nodes=School="output_school.csv"  
--relationships=submitted_at="output_school_submitted_at.csv"
```

Figure 3.1: Importation command.

To import the data, it is necessary to put the `CSV` files, generated by the python script, in the `bin` in folder of the DBMS where the database will run. After the command has been run, a database called `dblp` is created. When it is opened, the database is already populated with the specified nodes and relationships.

3.1.3. First schema

Below is the representation of the schema loaded in Neo4j:

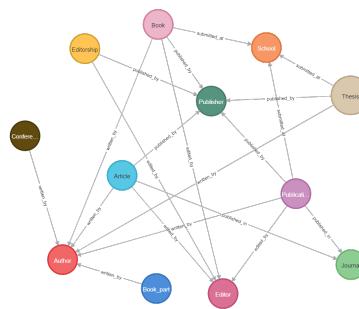


Figure 3.2: First schema.

The data set only contains a list of publications, the nodes that don't represent publication only contain a single attribute, since they had to be extracted from the publications' data. The nodes that represent publications contain all the data present in the data set. It is possible to identify the following nodes:

- **Article:** corresponds to the entity “journal article” and "Informal publication" in the ER diagram.
- **Author:** corresponds to the entity “author” in the ER diagram.
- **Book:** corresponds to the entity “book” in the ER diagram.
- **Book part:** corresponds to the entity “part in a book or collection” in the ER diagram.
- **Conference paper:** corresponds to the entity “conference and workshop paper” in the ER diagram.
- **Editor:** corresponds to the entity “editor” in the ER diagram.
- **Editorship:** corresponds to the entity “editorship” the ER diagram.
- **Journal:** corresponds to the entity “journal” in the ER diagram.
- **Publication:** corresponds to the entity “publication” in the ER diagram.
- **Publisher:** corresponds to the entity “publisher” in the ER diagram.
- **School:** corresponds to the entity “school” in the ER diagram.
- **Thesis:** corresponds to entity “thesis” in the ER diagram.

To represent inheritance in the graph, the interested nodes have the label of both the parent and the child. For example, an article is both **Article** and **Publication**.

Regarding relationships between nodes, they are shown in the following table:

Relationship	Start node	End node
written_by	Article Conference_paper Book Book_part Thesis Publication	Author
Edited_by	Article Editorship Book Publication	Editor
Published_in	Article Publication	Journal
Published_by	Article Editorship Book Thesis Publication	Publisher
Submitted_at	Thesis Book Publication	School

Table 3.1: Relationships

3.1.4. Differences with the ER diagram

Due to the limitations imposed by the chosen data set, there are some inevitable differences between the ER diagram and the graph implementation. The main difference is the absence of some entities:

1. Organization;
2. Journal edition: to compensate for the absence of this entity, the relationships that were connected to it are now instead routed directly to the Journal entity;
3. Conference edition: the same approach used to the Journal edition also applies

here.

4. Reference work;

5. Journal article and Informal publication: this two entities have been combined in the more general Article label;

The differences concerning the attributes have already been discussed int the section 3.1.3.

3.2. Queries

In this section we provide as brief description of the queries and their output by showing some screenshots of the result.

3.2.1. Creation and update queries

Query 1

This query merge the duplicates Authors and Editors (i.e. nodes with the same name).

```
1 match(a:Author), (e:Editor), (p:Publication)-[]→(e) where a.author = e.editor set a:Author:Editor
  create (p)-[r:edited_by]→(a)
2 match(a:Author), (e:Editor) WHERE a.author = e.editor detach delete e
```

Figure 3.3: Query 1.

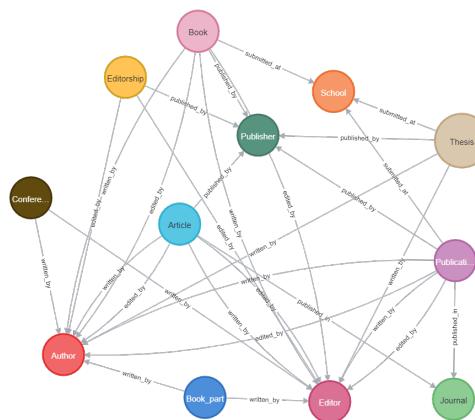


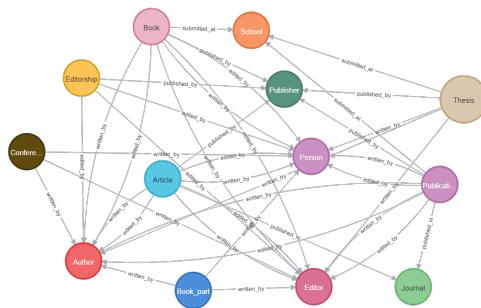
Figure 3.4: Output pf query 2.

Query 2

This query creates the generalization Person for Authors and Editors.

```
1 MATCH(a:Author) SET a:Author:Person Return a  
2 MATCH (n:Editor) set n:Editor:Person return n
```

Figure 3.5: Query 2.



3

Figure 3.6: Output pf query 2.

Query 3

This query search for two authors that have written two or more articles together, and creates a relationship **friend with** between them.

```
1 match (a1:Author)←[r1:written_by]-(p1:Publication)-[r2:written_by]→(a2:Author)←[r3:written_by]-  
  (p2:Publication)-[r4:written_by]→(a1)  
2 where p1.title ≠ p2.title and a1.author ≠ a2.author and id(a1) > id(a2) merge (a1)-  
  [r:friend_with]-(a2) return a1, a2
```

Figure 3.7: Query 3.

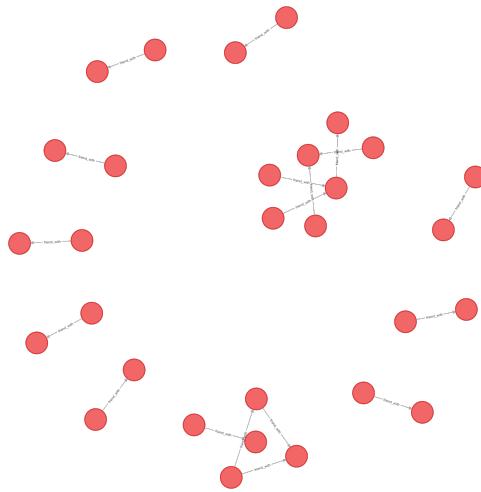


Figure 3.8: Output of query 3.

Query 4

This query creates the relationship `part of` between Book part and Book, the relationship `part of editorship` between Conference paper and Editorship.

```

1 match(c:Conference_paper), (e:Editorship)
2 where e.key = c.crossref
3 merge (c)-[:part_of_editorship]->[e]
4 match(bp:Book_part) match (b:Book)
5 where bp.crossref = b.key
6 merge (bp)-[:part_of]->(b)
7 return bp, b
  
```

Figure 3.9: Query 4.

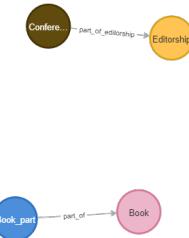


Figure 3.10: Output of query 4.

Query 5

This query creates some `Authors` and an `Article` written by them.

```

1 create(a1:Author {author: "Leonardo Cesani"})
2 create(a2:Author {author: "Matteo Paraboschi"})
3 create(a3:Author {author: "Fausto Lasca"})
4 create(a4:Author {author: "Alessio Buda"})
5 create(a5:Author {author: "Gabriele Munafò"})
6
7 create(a1)-[:friend_with]→(a2)
8 create(a1)-[:friend_with]→(a3)
9 create(a1)-[:friend_with]→(a4)
10 create(a1)-[:friend_with]→(a5)
11
12 create(a2)-[:friend_with]→(a3)
13 create(a2)-[:friend_with]→(a4)
14 create(a2)-[:friend_with]→(a5)
15
16 create(a3)-[:friend_with]→(a4)
17 create(a3)-[:friend_with]→(a5)
18
19 create(a4)-[:friend_with]→(a5)
20
21 create(:Art:Article {article: 25942,
22 authors: ["Leonardo Cesani", "Matteo Paraboschi", "Fausto Lasca", "Alessio Buda", "Gabriele Munafò"],
23 title: "Project Work 2022/2023, System and Methods for Big and Unstructured Data", year: 2022})
24 create(Art)-[:written_by]→(a1)
25 create(Art)-[:written_by]→(a2)
26 create(Art)-[:written_by]→(a3)
27 create(Art)-[:written_by]→(a4)
28 create(Art)-[:written_by]→(a5)
29 return a1, a2, a3, a4, a5, art

```

Figure 3.11: Query 5.



Figure 3.12: Output of query 5.

3.2.2. Request queries

Query 1

This query finds the shortest path in the graph between two given Authors.

```
1 match (a:Author {author: "Javier Gozámez"}),
2 match(b:Author {author : "Harvey Glickenstein"}),
3 p = shortestPath((a)-[*]-(b))
4 return p
```

Figure 3.13: Query 1.

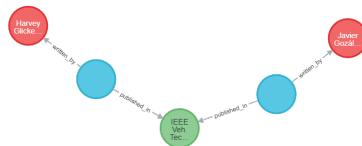


Figure 3.14: Output of query 1.

Query 2

This query finds the Author with the most number of publications published in a Journal.

```
1 match (p:Person) -[:written_by] - (c:Publication) - [r1:published_in] -> (j:Journal)
2 with p, j, count(r1) as in_journal, collect(c) as publication
3 RETURN p, j, publication, in_journal order by in_journal desc limit 1
```

Figure 3.15: Query 2.

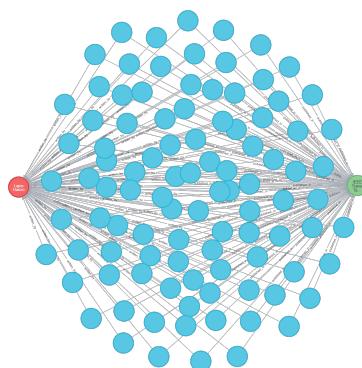


Figure 3.16: Output of query 2.

Query 3

This query extracts the Publications of two Authors who are colleagues and have the highest sum of Publications.

```

1 match (p1:Person)-[w1:written_by]-(pub1:Publication)
2 with p1, count(w1) as n_pub1, collect(pub1) as publication1
3 where n_pub1 > 30
4 match (p2:Person)-[w2:written_by]-(pub2:Publication)
5 with p1, n_pub1, publication1, p2, count(w2) as n_pub2, collect(pub2) as publication2
6 where id(p1) > id(p2) and n_pub2 > 30
7 match (p1)-[:friend_with]-(p2)
8 return p1, p2, publication1, publication2, n_pub1 + n_pub2 order by n_pub1 + n_pub2 desc limit 1
  
```

Figure 3.17: Query 3.

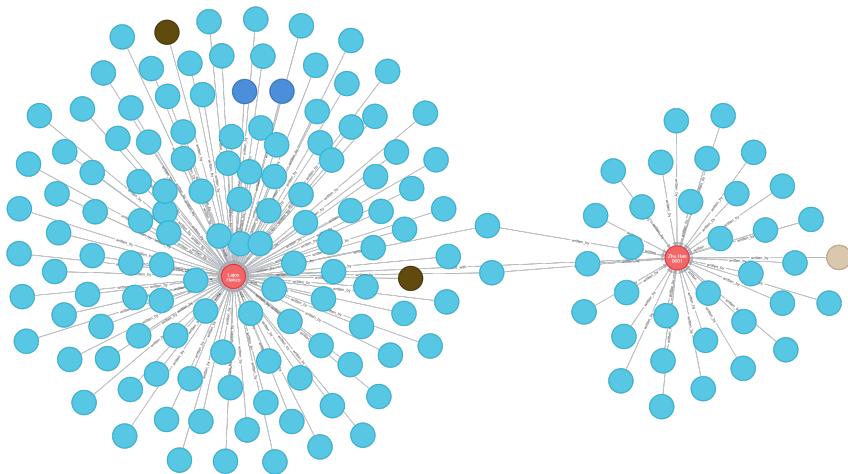


Figure 3.18: Output of query 3.

Query 4

This query finds the school with the most number of Publications authored by students whose name starts with the letter 'X'.

```

1 match (per:Person) -> [] -> (pub:Publication) -> [s:submitted_at] -> (school:School)
2 where per.author starts with 'X'
3 with school, count(s) as submissions, collect(pub) as publications, collect(per) as people
4 return school, people, publications, submissions order by submissions desc limit 1
  
```

Figure 3.19: Query 4.

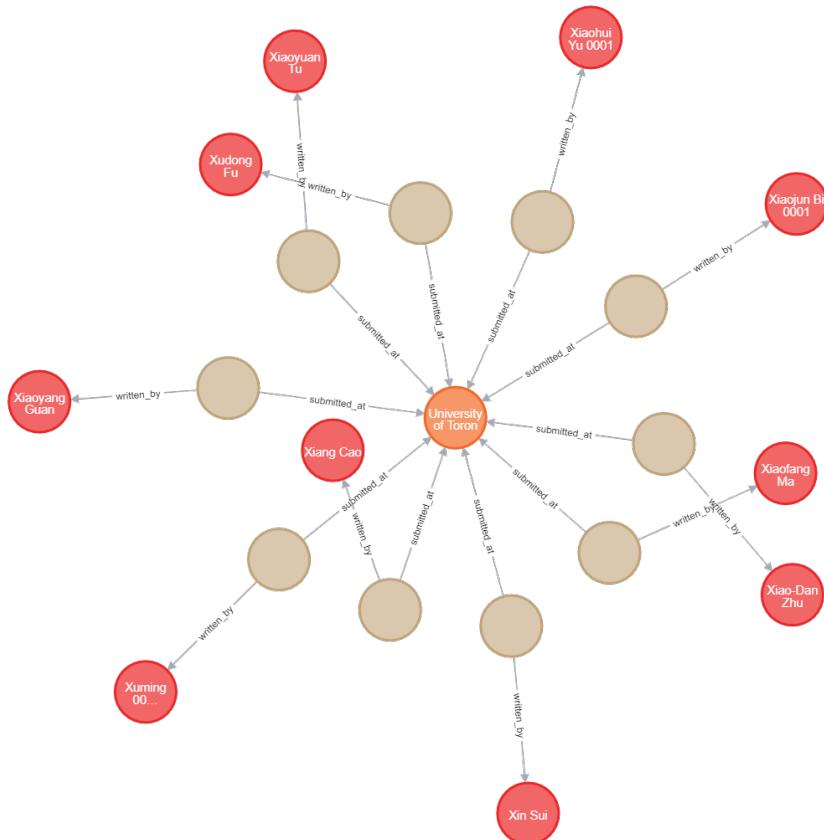


Figure 3.20: Output of query 4.

Query 5

This query extract all the 2-level colleagueship of the Author named "Lajos Hanzo".

```

1 match(p1:Person {author : "Lajos Hanzo"}) -[:friend_with*..2] -> (p2:Person)
2 where id(p1) <> id(p2)
3 return p1, p2
  
```

Figure 3.21: Query 5.

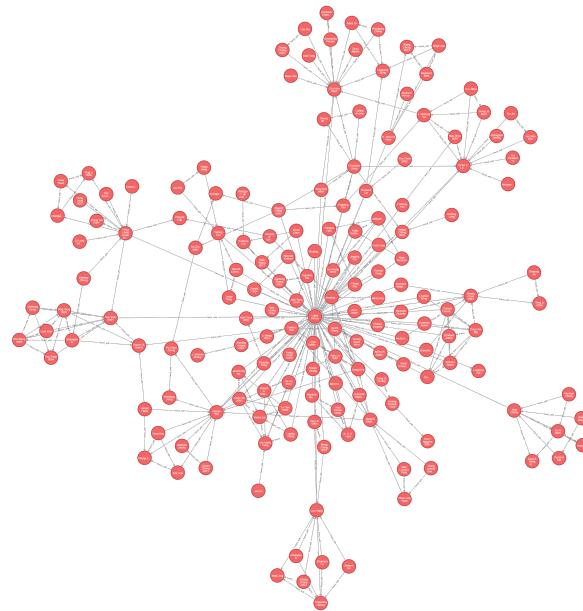


Figure 3.22: Output of query 5.

Query 6

This query finds the book, its chapters and the **Authors**, that has been written by the most number of people.

```
1 match (p:Person)←[r:written_by]-(bp:Book_part)-[r1:part_of]→(b:Book)
2 with b, count(distinct(p)) as authors, collect(bp) as book_part, collect(p) as people
3 return b, book_part, people, authors order by authors desc limit 1
```

Figure 3.23: Query 6.

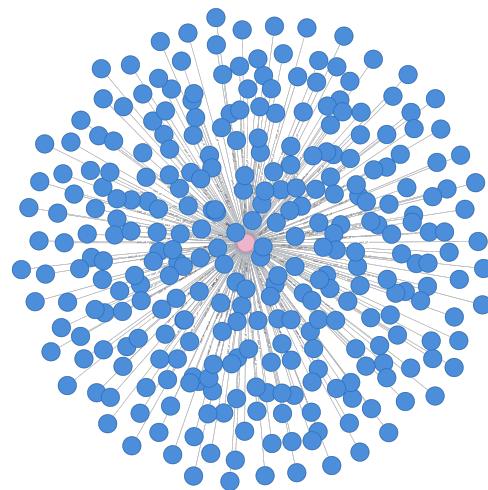


Figure 3.24: Output of query 6.

Query 7

This query finds the Authors, the Conference paper and the Editorship, have published a Conference paper in 2006.

```
1 match (p:Person)←[:written_by]-(c:Conference_paper)-[:part_of_editorship]→(e:Editorship)
2 where e.year = 2006
3 return p, c, e
```

Figure 3.25: Query 7.

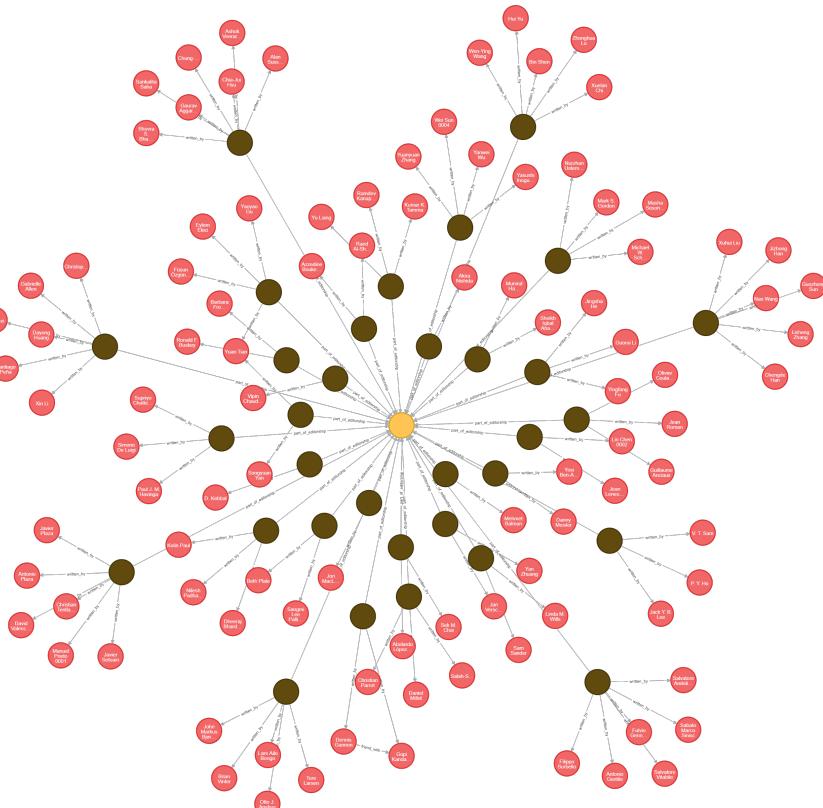


Figure 3.26: Output of query 7.

Query 8

This query search for the Author with the most Publications published in a Journal in a single year.

1 MATCH (a:Author)-[:written_by]-(p:Publication)-[:published_in]→(Journal)			
2 return a.author, p.year, count(p) as c order by c desc limit 1			
	a.author	p.year	c
	"Lajos Hanzo"	2021	13

Figure 3.27: Query 8.

Query 9

This query returns the journal articles published by Authors that submitted a Thesis at the Polytechnic University of Milan.

```
1 match (s:School {school:"Polytechnic University of Milan, Italy"})-[:submitted_at]-(t:Thesis)-[:written_by]→(au:Author)←[:written_by]-(ar:Article)-[:published_in]→(j:Journal)
2 return s, t, au, ar
```

Figure 3.28: Query 9.

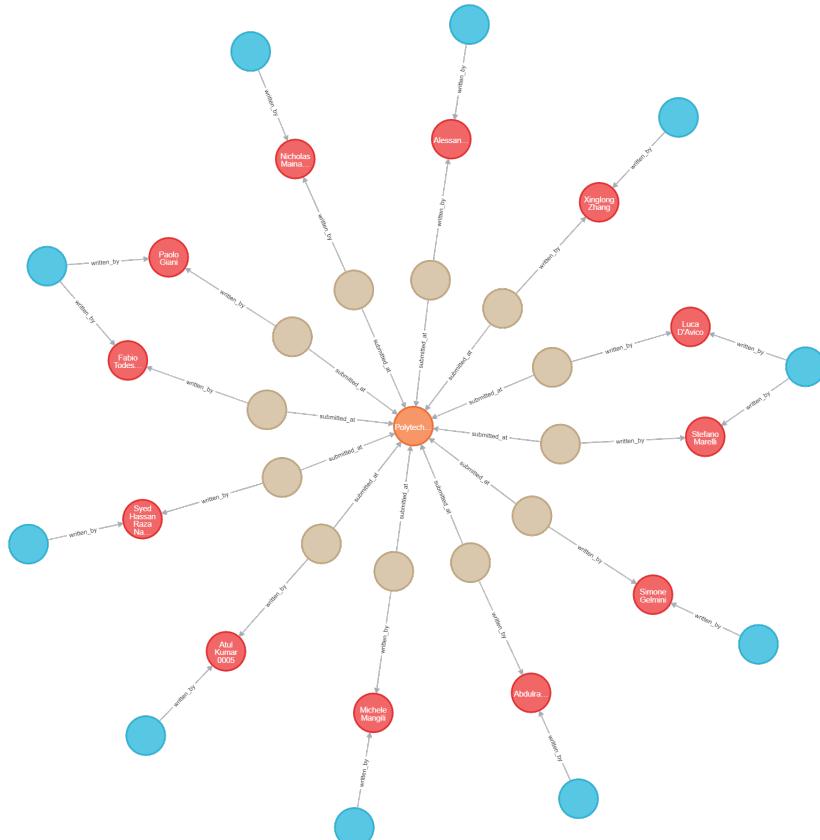


Figure 3.29: Output of query 9.

Query 10

This query returns a list of the Journals where **Articles**, written by **Authors** that submitted their thesis at the Polytechnic University of Milan, have been published. The list is ordered by the number of articles that has been published on each journal. **Journals** where

```
1 match (p:Publication)-[:published_in]→(j:Journal)
2 where exists( (p)-[:written_by]→(:Author)←[:written_by]-(:Thesis)-[:submitted_at]→(:School
  {school:"Polytechnic University of Milan, Italy"}) )
3 return j.journal, count(p) as n_publications
4 order by n_publications desc
```

	j.journal	n_publications
1	"IEEE Trans. Veh. Technol."	6
2	"IEEE Veh. Technol. Mag."	2
3	"it Inf. Technol."	1
4	"J. Comput. Secur."	1

Started streaming 4 records in less than 1 ms and completed after 226 ms.

Figure 3.30: Query 10.

4 | MongoDB

4.1. Document structure

Regarding the document structure, we followed the assignment's specification, including:

- **title**: the article's title;
- **authors**: a vector of embedded documents, each representing one of the authors of the article. Each author is described by the following attributes: name, affiliation, email and biography;
- **keywords**: a vector of strings representing the article's keywords;
- **journal**: an embedded document representing the journal edition in which the article has been published. For the journal the following fields are present: name of the journal, volume, number, the date of publication and the pages in which the article may be found;
- **abstract**: representing the article's abstract;
- **sections**: an array of documents representing the sections of the article. Each section has a title and one or more paragraphs, represented by an array of strings. It may also have one or more figures with the relative captions, represented as documents with two fields: url of the image and caption. Moreover, each section may also contain subsections. These are presented as an array of documents. Their structure is similar to that of a section with the only difference that a subsection cannot contain other subsections.
- **bibliography**: references to other documents (articles) cited in the present article.

4.1.1. Example of a document

```

▼ 0:
  _id: "63776ddd297f4328218ebc59"
  title: "On the Penrose process for rotating black holes"
  ▶ authors:
    ▼ 0:
      name: "Salinas Bauer"
      affiliation: "Geekol"
      email: "salinasbauer@geekol.com"
      ▶ bio:
        "Veniam ea velit in nulla...trud exercitation sunt."
    ▶ 1:
    ▶ 2:
    ▶ 3:
    ▶ 4:
    ▶ 5:
  ▶ keywords:
    0: "Information Technology/Informatics"
    1: "Research Networking/GGRID"
    2: "Process automation"
  ▶ journal:
    name: "Optim. Lett."
    volume: 11
    number: 58
    date: "1987-05-\t\t\t\t\t\t23"
    pages: "197 – 388"
  ▶ abstract:
    " Penrose described a pr... of  $1-1/\sqrt{2}$  =29%.\\n"
  ▶ sections:
    ▶ 0:
    ▶ 1:
    ▶ 2:
      ▶ title: "Chain enumeration of $k$...s of classical\\n types"
      ▶ paragraphs:
        ▶ 0:
          "Went here for brunch tod... or beer in the future."
        ▶ figure:
        ▶ subsections:
          ▶ 0:
            title: "Green-Tao theorem in function fields"
            ▶ paragraphs:
            ▶ figure:
      ▶ bibliography:
        0: "63776dddcbf3d894a5f8aff3"
        1: "63777073d939a5250d1c80ef"
        2: "63776ddef403a87eea28accf"

```

Figure 4.1: Example of a document.

Figure 4.1 shows an example document. It has a unique identifier `_id`, a title, an array of authors, an array of keywords, the details of the journal edition on which it has been published, an abstract, an array of sections and potentially subsections, a bibliography array containing `_ids` of articles cited in the document. For every author, journal, section and subsection relevant information is provided in the form of embedded documents with all the relevant field described Section 4.1.

4.1.2. Differences with the ER model

There are no relevant differences between the attributes of the entity `article` defined in the ER model and the fields of the article document as described in section 4.1.

Minor differences are:

- the `DOI` attribute of the **Publication** entity is now replaced by a unique identifier `_id`;
- the `title` of the journal in the ER model is now referred to as `name` of the journal;
- the `year` attribute of the **Publication** entity inherited by the **Article** entity becomes the `date` of publication of the journal edition;
- the following non relevant attributes of the **Publication** entity are not present in the document structure: `URL`, `timestamp`, `bib_source` and `bib_url`;
- the **Author** has an `email` associated to him, which was not present in the ER model.

4.2. Data upload and transformation

4.2.1. Creation of the dataset

To create the dataset to import in MongoDB we proceeded as follows.

First, we defined the structure of the document according to what was specified in the delivery of the project, as described in Section 4.1.

Then, we used the online tool <https://json-generator.com/> to create three JSON files of 2000 records, each with a slight variation of the defined structure: the types of documents only differ for the number of sections or the presence (or absence) of subsections and figures (for the structure of the three types of documents see Appendix A).

To assign values to each field, we used functions provided by the tool for the generation of random content. For example, we generated random names, company names and emails

to fill the data of the different authors. Then, we used functions that generated content from the *Lorem ipsum* text to fill more complex fields, where a longer portion of text was required, such as the body of the different paragraphs.

This, however, poses a problem: the content of the articles is not varied enough, with many articles similar to each other.

4.2.2. Data modification

To solve this issue we decided to include text coming from different sources, selected among those suggested in the project delivery.

In particular, to fill in the data relative to keywords we use a list scientific terms taken from the official website of the Chamber of Commerce of Genoa. As journal names, articles and paragraphs titles, we used those present in the dblp database, used in Chapter 3. To complete the abstracts we used a dataset from the Kaggle website, containing data from the ArXiv dataset: in this particular case we selected only the relevant data for our application. In the end, paragraphs have been taken from the Yelp review dump. The data have been inserted into separate CSV files.

Python script

To substitute the fields mentioned in Section 4.2.2, we wrote a simple Python script which opens the CSV files containing data to be substituted and writes them in the original JSON file. The used script may be found in Appendix B.

4.2.3. Data upload

The generated dataset has been imported in MongoDB by using the built-in feature provided by MongoDB Compass.

A new database was created and then, inside it, a new collection. The data has been inserted in the created collection by using the import feature of MongoDB Compass (*ADD DATA*), selecting JSON as the *input file type* and the three documents obtained using the Python script as *files* to be imported.

4.3. Queries

4.3.1. Data creation and update

Query 1: Insertion of an article

The following query adds an article to the `articles` collection. The inserted article contains all the fields defined in Section 4.1.

```
Atlas atlas-l7mawr-shard-0 [primary] journal_articles>db.articles.insertOne({
    "title": "SMBUD Project",
    "authors": [
        { "name": "Alessio Buda",
            "affiliation": "Politecnico di Milano",
            "email": "alessio.buda@mail.it",
            "bio": "Alessio's biography"
        },
        { "name": "Leonardo Cesani",
            "affiliation": "Politecnico di Milano",
            "email": "leonardo.cesani@mail.it",
            "bio": "Leonardo's biography"
        }
    ],
    "keywords": ["Big data", "University", "Project"],
    "journal": {
        "name": "Frontiers in Data Science",
        "volume": 1,
        "number": 12,
        "date": ISODate('2022-11-19T00:00:00.000Z'),
        "pages": "123-456",
    },
    "abstract": "Article's abstract",
    "sections": [
        { "title": "Section's title",
            "paragraphs": ["Text 1", "Text 2", "Text 3"]
        }
    ],
    "bibliography": [ObjectId('63776ddd15bc88e6627df47')]
})|
```

Figure 4.2: Query 1.

```
< { acknowledged: true,
  insertedId: ObjectId("63791cd1d3b5ae78ecabcd4a") }
> db.articles.find({_id: ObjectId("63791cd1d3b5ae78ecabcd4a")})
< { _id: ObjectId("63791cd1d3b5ae78ecabcd4a"),
  title: 'SMBUD Project',
  authors:
  [ { name: 'Alessio Buda',
      affiliation: 'Politecnico di Milano',
      email: 'alessio.buda@mail.it',
      bio: 'Alessio\\'s biography' },
    { name: 'Leonardo Cesani',
      affiliation: 'Politecnico di Milano',
      email: 'leonardo.cesani@mail.it',
      bio: 'Leonardo\\'s biography' } ],
  keywords: [ 'Big data', 'University', 'Project' ],
  journal:
  { name: 'Frontiers in Data Science',
    volume: 1,
    number: 12,
    date: 2022-11-19T00:00:00.000Z,
    pages: '123-456' },
  abstract: 'Article\\'s abstract',
  sections:
  [ { title: 'Section\\'s title',
      paragraphs: [ 'Text 1', 'Text 2', 'Text 3' ] } ],
  bibliography: [ ObjectId("63776ddd15bc88e6627df47") ] }
```

Figure 4.3: Query 1: result.

Query 2: Insertion of an article with missing fields

The following query inserts an article in the `articles` collection. The inserted article does not contain fundamental fields such as `authors` and `journal`. This shows how a document-based database can be flexible and does not require a rigid structure for the articles that are being added.

```
Atlas atlas-lx2lpr-shard-0 [primary] SMBUD_Database> db.Articles.insertOne({
    "title": "Test_article",
    "sections": [
        {
            "title": "Test_section",
            "paragraphs": [
                "Example of section text"
            ],
            "subsections": [
                {
                    "title": "Test_subsection",
                    "paragraphs": [
                        "Example of subsection text"
                    ]
                }
            ]
        }
    ]
})
```

Figure 4.4: Query 2.

```
< { _id: ObjectId("63794ef4499a81e5ffd91bdc"),
  title: 'Test_article',
  sections: [
    { title: 'Test_section',
      paragraphs: [ 'Example of section text' ],
      subsections: [
        { title: 'Test_subsection',
          paragraphs: [ 'Example of subsection text' ] } ] } ] }
```

Figure 4.5: Query 2: result.

Query 3: Update an abstract of an article

The following query modifies the value of the attribute `abstract` in an existing document. After selecting the article using its `_id`, the `$set` operator is used to modify the `abstract` field.

```
Atlas atlas-lx2lpr-shard-0 [primary] SMBUD_Database> db.Articles.updateOne(
    { "_id": ObjectId("63794ef4499a81e5ffd91bdc") },
    { $set: { "abstract": "Example of description of an article" } }
)
```

Figure 4.6: Query 3.

```
< { _id: ObjectId("63794ef4499a81e5ffd91bdc"),
  title: 'Test article',
  sections:
  [ { title: 'Test_section',
    paragraphs: [ 'Example of section text' ],
    subsections:
    [ { title: 'Test_subsection',
      paragraphs: [ 'Example of subsection text' ] } ] } ],
  authors:
  [ { name: 'Marco Brambilla',
    affiliation: 'Politecnico di Milano',
    email: 'marco.brambilla@example.com',
    bio: 'Teaches theory part of SMBUD' },
    { name: 'Andrea Tocchetti',
      affiliation: 'Politecnico di Milano',
      email: 'andrea.tocchetti@example.com',
      bio: 'Teaches excercise part of SMBUD' } ],
  abstract: 'Example of description of an article' }
```

Figure 4.7: Query 3: result.

Query 4: Add authors to an article

The following query adds the attribute `authors` to the article previously created (Query 2) by using the `$set` operator as seen in Query 3. This shows how the `update` operator can be used not only to update already existing information in an existing document, but also to add new information to it.

```
Atlas atlas-lx2lpr-shard-0 [primary] SMBUD_Database> db.Articles.updateOne(
  { "_id":ObjectId("63794ef4499a81e5ffd91bdc") },
  { $set:{ "authors": [
    {
      "name":"Marco Brambilla",
      "affiliation":"Politecnico di Milano",
      "email":"marco.brambilla@example.com",
      "bio":"Teaches theory part of SMBUD"
    },
    {
      "name":"Andrea Tocchetti",
      "affiliation":"Politecnico di Milano",
      "email":"andrea.tocchetti@example.com",
      "bio":"Teaches excercise part of SMBUD"
    }
  ]}}
)
```

Figure 4.8: Query 4.

```
< { _id: ObjectId("63794ef4499a81e5ffd91bdc"),
  title: 'Test_article',
  sections:
  [ { title: 'Test_section',
    paragraphs: [ 'Example of section text' ],
    subsections:
    [ { title: 'Test_subsection',
      paragraphs: [ 'Example of subsection text' ] } ] },
  authors:
  [ { name: 'Marco Brambilla',
    affiliation: 'Politecnico di Milano',
    email: 'marco.brambilla@example.com',
    bio: 'Teaches theory part of SMBUD' },
  { name: 'Andrea Tocchetti',
    affiliation: 'Politecnico di Milano',
    email: 'andrea.tocchetti@example.com',
    bio: 'Teaches excercise part of SMBUD' } ] }
```

Figure 4.9: Query 4: result.

Query 5: Add a section to an article

The following query adds a section to an article, for instance the one created in Query 1. After matching the article using its `_id`, the `sections` array is accessed using the dot notation (`sections.1`). Since there is only one section in the array (with index equal to 0), a new section is created. For the created sections the `title` and `paragraphs` are specified as fields of an embedded document.

Figure 4.10 shows the document before the update, the query is presented in Figure 4.11. Finally, the result of the query can be seen in Figure 4.12.

```
> db.articles.find({"_id": ObjectId("637a39089b27565984ad56cf")})
< { _id: ObjectId("637a39089b27565984ad56cf"),
  title: 'SMBUD Project',
  authors:
  [ { name: 'Alessio Buda',
    affiliation: 'Politecnico di Milano',
    email: 'alessio.buda@mail.it',
    bio: 'Alessio\'s biography' },
  { name: 'Leonardo Cesani',
    affiliation: 'Politecnico di Milano',
    email: 'leonardo.cesani@mail.it',
    bio: 'Leonardo\'s biography' } ],
  keywords: [ 'Big data', 'University', 'Project' ],
  journal:
  { name: 'Frontiers in Data Science',
    volume: 1,
    number: 12,
    date: 2022-11-19T00:00:00Z,
    pages: '123-456' },
  abstract: 'Article\'s abstract',
  sections:
  [ { title: 'Section\'s title',
    paragraphs: [ 'Text 1', 'Text 2', 'Text 3' ],
    subsections: [ { title: 'Subsection\'s title', paragraphs: [ 'Par text 1' ] } ] } ],
  bibliography: [ ObjectId("63776ddd15bc88e6627df47") ] }
```

Figure 4.10: Query 5: before.

```
Atlas atlas-l7mawr-shard-0 [primary] journal_articles>db.articles.updateOne({
    "_id": ObjectId("637a39089b27565984ad56cf")
}, {
    $set: {"sections.1": {"title": "New section's title", "paragraphs": ["Section text 1"]}}
})

```

Figure 4.11: Query 5.

```
< { _id: ObjectId("637a39089b27565984ad56cf"),
  title: 'SMBUD Project',
  authors: [
    { name: 'Alessio Buda',
      affiliation: 'Politecnico di Milano',
      email: 'alessio.buda@mail.it',
      bio: 'Alessio\'s biography' },
    { name: 'Leonardo Cesani',
      affiliation: 'Politecnico di Milano',
      email: 'leonardo.cesani@mail.it',
      bio: 'Leonardo\'s biography' } ],
  keywords: [ 'Big data', 'University', 'Project' ],
  journal: {
    name: 'Frontiers in Data Science',
    volume: 1,
    number: 12,
    date: 2022-11-19T00:00:00.000Z,
    pages: '123-456' },
  abstract: 'Article\'s abstract',
  sections: [
    { title: 'Section\'s title',
      paragraphs: [ 'Text 1', 'Text 2', 'Text 3' ],
      subsections: [ { title: 'Subsection\'s title', paragraphs: [ 'Par text 1' ] } ] },
    { title: 'New section\'s title',
      paragraphs: [ 'Section text 1' ] },
    bibliography: [ ObjectId("63776ddd15bc88e6627df47") ] }

```

Figure 4.12: Query 5: result.

Query 6: Add a subsection to an article

The following query adds a subsection to an article, for instance the one created in Query 1. After matching the article using its `_id`, the first element of the `sections` array is accessed using the dot notation (`sections.0`). In this element, represented by an embedded document, a new `subsections` field is created. The new subsection is added as a document in the array of subsections.

Figure 4.13 shows the document before the update, the query is presented in Figure 4.14. Finally, the result of the query can be seen in Figure 4.15.

```
> db.articles.find({"_id": ObjectId("637a39089b27565984ad56cf")})
< { _id: ObjectId("637a39089b27565984ad56cf"),
  title: 'SMBUD Project',
  authors:
  [ { name: 'Alessio Buda',
      affiliation: 'Politecnico di Milano',
      email: 'alessio.buda@mail.it',
      bio: 'Alessio's biography' },
    { name: 'Leonardo Cesani',
      affiliation: 'Politecnico di Milano',
      email: 'leonardo.cesani@mail.it',
      bio: 'Leonardo's biography' } ],
  keywords: [ 'Big data', 'University', 'Project' ],
  journal:
  { name: 'Frontiers in Data Science',
    volume: 1,
    number: 12,
    date: 2022-11-19T00:00:00.000Z,
    pages: '123-456' },
  abstract: 'Article's abstract',
  sections:
  [ { title: 'Section's title',
      paragraphs: [ 'Text 1', 'Text 2', 'Text 3' ] } ],
  bibliography: [ ObjectId("63776ddd15bc88e6627df47") ] }
```

Figure 4.13: Query 6: before.

```
Atlas atlas-l7mawr-shard-0 [primary] journal_articles> db.articles.updateOne(
  "_id": ObjectId("637a39089b27565984ad56cf")
),{
  $set: {"sections.0.subsections": [{"title": "Subsection's title", "paragraphs": ["Par text 1"]}]}
```

Figure 4.14: Query 6.

```
> db.articles.find({"_id": ObjectId("637a39089b27565984ad56cf")})
< { _id: ObjectId("637a39089b27565984ad56cf"),
  title: 'SMBUD Project',
  authors:
  [ { name: 'Alessio Buda',
      affiliation: 'Politecnico di Milano',
      email: 'alessio.buda@mail.it',
      bio: 'Alessio's biography' },
    { name: 'Leonardo Cesani',
      affiliation: 'Politecnico di Milano',
      email: 'leonardo.cesani@mail.it',
      bio: 'Leonardo's biography' } ],
  keywords: [ 'Big data', 'University', 'Project' ],
  journal:
  { name: 'Frontiers in Data Science',
    volume: 1,
    number: 12,
    date: 2022-11-19T00:00:00.000Z,
    pages: '123-456' },
  abstract: 'Article's abstract',
  sections:
  [ { title: 'Section's title',
      paragraphs: [ 'Text 1', 'Text 2', 'Text 3' ],
      subsections: [ { title: "Subsection's title", paragraphs: [ 'Par text 1' ] } ] } ],
  bibliography: [ ObjectId("63776ddd15bc88e6627df47") ] }
```

Figure 4.15: Query 6: result.

4.3.2. Data manipulation

Query 7: Articles mentioning "ramen"

The following query returns the articles that contain the word "ramen" in their paragraphs, filtering out articles with few authors or sections.

To be precise the query returns only the articles that have: 8 or more authors, 3 or more sections or at least both 6 authors and 2 sections.

The query uses `$exists` to filter the size of the authors and sections arrays, since `$size` only accepts a specific value, not a range of values.

To search for the word "ramen" in the articles, `$text` and `$search` are used. Before executing the query, a textual index on `sections.paragraphs` was created, allowing for the text search.

A projection is also made to make it easier to read the results. Figure 4.17, presenting the results, only shows 2 of the 81 articles returned by the query.

```
Atlas atlas-l7mawr-shard-0 [primary] journal_articles> db.articles.find({
    $or: [
        {"authors.7": {$exists: true}},
        {"sections.2": {$exists: true}},
        {"authors.6": {$exists: true}, "sections.1": {$exists: true}}
    ],
    $text: {$search: "ramen"}
}, {
    "title": 1,
    "authors.name": 1,
    "sections.paragraphs": 1
})
```

Figure 4.16: Query 7.

```
{
  _id: '637764dde0b7cb44a3185b6c7',
  title: 'Aspects of Higgs Production at the LHC',
  authors: [
    { name: 'Briggs Stokes' },
    { name: 'Rosario Barlow' },
    { name: 'Davenport Lamb' },
    { name: 'Palmer Gardner' },
    { name: 'Boyd Hunt' },
    { name: 'Craig Clay' }
  ],
  sections: [
    {
      paragraphs: [
        'My wife and I ordered a cheese steak and a thin crust square pizza. The cheese steak was hot, gooey and tasty. The pizza was crispy and very tasty. They have off s \'DELICIOUS!! Had the signature Mac n cheese. IT was good. But I didn\'t care for the truffle oil. I had the salad with the salmon accompanied with the house made dress: \'Nice decors and atmosphere. Cute little details about this place stands out from other ramen places. The ramen was good. I really like the broth that I got and the p',
        'Enterprise loves to advertise it\'s $9.99 a day weekend special, but neglects to mention that all locations in or near downtown Philadelphia do NOT participate in the \'What just happened? Did I just get thrown out?!?! It was 10:20AM and the place was not busy at all! If they don\'t want customers they should put a sign up!!! ] ',
        {
          paragraphs: [
            'this place is wonderful. food is delicious and they are very accommodating. Was here for a large party and told them I had food allergies and they made separate plates. \'This place is pretty awesome. Love the ambiance of being in this essentially wooden warehouse type thing. Friendly staff. Nice collection of interesting cocktails. T'
          ]
        }
      ]
    }
  ],
  _id: '637764de1f4551e2daea9e32',
  title: 'Color Magnetic Flux Tubes in Dense QCD. II: Effective World-Sheet Theory',
  authors: [
    { name: 'Bridges Goodwin' },
    { name: 'Wells Cleveland' },
    { name: 'Bridgette Figueroa' },
    { name: 'Cooke Becker' },
    { name: 'Hayden Shelton' },
    { name: 'Alfreda Rhodes' },
    { name: 'Jody Klein' }
  ],
  sections: [
    {
      paragraphs: [
        'I absolutely love, love, love this place. The menu changes seasonally and every month there is a new price fixe menu. The chef is extremely talented. We l',
        {
          paragraphs: [
            'Despite being an Asian food devotee, I\'ve never eaten Filipino food before. I was stoked when we decided to take a day trip out to the Northeast to sample Philly\'s \'Delicious goodness :). Corn fritters hit the spot, and the breading on the chicken is so yummy. Mashed potatoes were average but made for very good dipping for chick',
            'This is literally the worst ramen bar I have ever been to. The noodles are not completely cooked and are sticky. The sushi doesn\'t have enough sauce on it. The soy :',
            {
              paragraphs: [
                'I never write reviews but I had to this time. This is the second time I\'ve tried this place and I will never have it again. The first time, I tried their spaghetti. \'Good for a quick eat.\n\nIt\'s cheap and the exact same as the uptown version off Claiborn. I also heard that they are going to start being open until 2am. Food is'
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

Figure 4.17: Query 7: result.

Query 8: Conditions on keywords

The following query finds one document in the dataset that contains the keyword “Radar” and at least one of the keywords “Remote Control” or “Narrow Band Technologies”. The `$or` operator allows to filter based on conditions defined on multiple attributes and to perform a logical OR operation between them. For readability reasons, only the authors, the title and the keywords are displayed.

```
Atlas atlas-l7mawr-shard-0 [primary] journal_articles> db.articles.findOne(
    {
        "$or": [
            {"keywords": "Remote Control"},
            {"keywords": "Narrow Band Technologies"}
        ],
        "keywords": "Radar"
    },
    {
        "sections": 0,
        "journal": 0,
        "abstract": 0,
        "bibliography": 0
    }
)
```

Figure 4.18: Query 8.

```
{
  _id: '63776ddcc75ab6dad8fa5aa',
  title: 'On the evolution of superposition of squeezed displaced number states\n with the multiphoton Jaynes-Cummings model',
  authors: [
    { name: 'Randolph Glenn',
      affiliation: 'Exoteric',
      email: 'randolphglenn@exoteric.com',
      bio: 'Tempor dolor aute aliqua esse in ad esse laboris fugiat eiusmod ullamco incididunt enim excepteur.' },
    { name: 'Debbie Glover',
      affiliation: 'Unq',
      email: 'debbieglover@unq.com',
      bio: 'Culpa non nulla consequat sunt aute fugiat.' },
    { name: 'Wilda Benton',
      affiliation: 'Lexicondo',
      email: 'wildabenton@lexicondo.com',
      bio: 'Cillum cillum sunt deserunt pariatur officia eu.' },
    { name: 'Tricia Roberson',
      affiliation: 'Zoid',
      email: 'triciaroberson@zoid.com',
      bio: 'Est nisi consectetur adipisicing tempor occaecat nostrud est incididunt aliqua fugiat pariatur reprehenderit.' },
    { name: 'Ann Faulkner',
      affiliation: 'Concility',
      email: 'annfaulkner@concility.com',
      bio: 'Sunt sunt voluptate minim laboris sint qui quis.' },
    { name: 'Leigh Freeman',
      affiliation: 'Bezal',
      email: 'leighfreeman@bezal.com',
      bio: 'Nulla dolor anim veniam aliquip non anim adipisicing.' } ],
  keywords: [
    'Speech Processing/Technology',
    'Radar',
    'Narrow Band Technologies' ]
}
```

Figure 4.19: Query 8: result.

Query 9: Interesting articles in specific journals

The following query returns the articles published in a IEEE or IBM journal dealing with specific topics. The article topics are filtered using keywords: the only articles shown are the ones that contain both "Information Technology/Informatics" and "Printed circuits

and integrated circuits".

This query uses `$regex` to find the journal titles that contain either "IEEE" or "IBM". A projection is also made to make it easier to read the results.

```
Atlas atlas-l7mawr-shard-0 [primary] journal_articles> db.articles.find(
    keywords: {$all: ["Information Technology/Informatics", "Printed circuits and integrated circuits"] },
    $or: [
        {"journal.name": {$regex: "IEEE"}},
        {"journal.name": {$regex: "IBM"}}
    ],
    {
        "title": 1,
        "journal.name":1,
        "keywords":1
    }
)
```

Figure 4.20: Query 9.

```
< { _id: '63776dd466d412fcbb597fdb',
  title: 'Operation of a 1-Liter-Volume Gaseous Argon Scintillation Counter',
  keywords:
  [ 'Environmental and Biometrics Sensors/Actuators',
    'Information Technology/Informatics',
    'Printed circuits and integrated circuits' ],
  journal: { name: 'IBM J. Res. Dev.' } )
{ _id: '63776ddebd78270b31375a65',
  title: 'Comment on \'Note on the dog-and-rabbit chase problem in introductory\n kinematics\'',
  keywords:
  [ 'Printed circuits and integrated circuits',
    'Information Technology/Informatics',
    'Computer Technology/Graphics/Meta Computing' ],
  journal: { name: 'IEEE Trans. Veh. Technol.' } )
{ _id: '63776dde85d9a98e5725aecb',
  title: 'Constructing quantum games from a system of Bell\'s inequalities',
  keywords:
  [ 'Research Networking/GRID',
    'Information Technology/Informatics',
    'Printed circuits and integrated circuits' ],
  journal: { name: 'IEEE Veh. Technol. Mag.' } )
{ _id: '637770ab6b6626100195d0ae',
  title: 'Parallel electron-hole bilayer conductivity from electronic interface\n reconstruction',
  keywords:
  [ 'Human Language Technologies',
    'Information Technology/Informatics',
    'Printed circuits and integrated circuits' ],
  journal: { name: 'IEEE Trans. Veh. Technol.' } )
{ _id: '637770a0e083266ada6ca9ce',
  title: 'The Scattering Approach to the Casimir Force',
  keywords:
  [ 'Printed circuits and integrated circuits',
    'Information Technology/Informatics',
    'Data Protection/Storage/Cryptography/Security' ],
  journal: { name: 'IEEE Trans. Veh. Technol.' } }
```

Figure 4.21: Query 9: result.

Query 10: Conditions on fields of embedded documents

The following query finds 5 documents in the dataset for which at least one of the following conditions holds:

- are written by an author whose affiliation is “Valreda”;
- are published in “Comput. J.”;
- are written by an author whose affiliation is “Stralum”.

The titles of the documents are displayed.

This query uses the `limit()` method to limit the search only to 5 documents. The author, the name of the journal and the affiliation are in embedded documents: the author's name and its affiliation are in the Author document whereas the journal name is in the Journal document. Thus, to access these fields, we need to use the dot notation.

```
db.Big_Data_V2.find({
  "$or": [{"authors.affiliation": "Valreda"}, {"journal.name": "Comput. J."}, {"authors.affiliation": "Stralum"}],
},
{
  "title": 1,
}).limit(5)
```

Figure 4.22: Query 10.

```
{ _id: '63776dddc6edc5a',
  title: 'Charmonium in the vector channel at finite temperature from QCD sum\n rules' }
{ _id: '63776ddd2f7fbf02b6ecfdf3',
  title: 'HD 77361: A new case of super Li-rich K giant with anomalous low 12C/13C\n ratio' }
{ _id: '63776ddd2f51020922409414',
  title: 'A Bayesian Approach Accounting for Stochastic Fluctuations in Stellar\n Cluster Properties' }
{ _id: '63776dddc850bb526d54eef1',
  title: 'Molecular Line Observations of Infrared Dark Clouds II: Physical\n Conditions' }
{ _id: '63776ddd5c37f2c5afb5342c',
  title: 'Adding a New Dimension to DNA Melting Curves' }
```

Figure 4.23: Query 10: result.

Query 11: Average number of articles published in a journal

The following query returns the average number of articles published in a journal. The query uses an aggregation pipeline, the main operations are:

- `$group` and `$count` are used to return the number of articles published by each journal;
- Another `$group` is used, together with `$avg`, to compute the average on all the records of the list obtained with the previous step. To collect all the records in one, "null" is used as `_id`.

```
Atlas atlas-52pbmf-shard-0 [primary] Project> db.Big_Data_V2.aggregate([
  {
    $group: {
      _id: '$journal.name',
      count: {
        $count: {}
      }
    }
  },
  {
    $group: {
      _id: null,
      average: {
        $avg: '$count'
      }
    }
  },
  {
    $project: {
      _id: 0
    }
  }
])
```

Figure 4.24: Query 11.

```
< { average: 214.28571428571428 }>
```

Figure 4.25: Query 11: result.

Query 12: Conditions on journals (without using count())

The following query shows the number of documents in the dataset that were published in the journal “Comput. J.” and that are the ninth volume of a series. This query uses the aggregation pipeline, and is divided in 2 principal sections:

- The `$match` operator allows to filter the documents, finding only the required ones.
- The `$group` operator allows to aggregate the documents and to sum them using the `$sum` operator which sums every document and returns their total number.

```
db.Big_Data_V2.aggregate([
  {
    $match: {
      "journal.name": "Comput. J.",
      "journal.volume": {"$gt": 9}
    }
  },
  {
    $group: {
      _id: 0,
      count: {
        $sum: 1
      }
    }
  },
  {
    $project: {
      _id: 0,
      count: 1
    }
  }
]))
```

Figure 4.26: Query 12.

```
{ count: 182 }
```

Figure 4.27: Query 12: result.

Query 13: Most used keywords

This query returns the number of articles that contain each keyword, sorted from the most used to the least used. Since the keywords are stored in an array, the use of `$unwind` is necessary. The query uses an aggregation pipeline, with these operations:

- The previously mentioned `$unwind` is used to separate the elements of the keywords array. Every article is separated into multiple copies, one for each element of the array keywords;
- `$group` is used to group together all the articles that contain the same keyword from the obtained list. `$count` is used to count the number of articles grouped together.
- Finally, `$sort` is used to sort the elements in descending order.

```
Atlas atlas-52pkmf-shard-0 [primary] Project> db.Big_Data_V2.aggregate([
  {
    $unwind: {
      path: '$keywords'
    }
  },
  {
    $group: {
      _id: '$keywords',
      count: {
        $count: {}
      }
    }
  },
  {
    $sort: {
      count: -1
    }
  }
])|
```

Figure 4.28: Query 13.

```
< { _id: 'Quantum Informatics', count: 397 },
  { _id: 'Optical Networks and Systems', count: 387 },
  { _id: 'Magnetic and superconducting materials/devices',
    count: 383 },
  { _id: 'Embedded Systems and Real Time Systems', count: 383 },
  { _id: 'High Frequency Technology/Microwaves', count: 382 },
  { _id: 'Printed circuits and integrated circuits', count: 378 },
  { _id: '3D printing', count: 373 },
  { _id: 'Smart cards and access systems', count: 372 },
  { _id: 'Electronic circuits/components and equipment',
    count: 368 },
  { _id: 'Environmental and Biometrics Sensors/Actuators',
    count: 367 },
  { _id: 'Electronic engineering', count: 363 },
  { _id: 'Peripherals Technologies (Mass Data Storage/Displays)',
    count: 353 },
  { _id: 'Semiconductors', count: 326 },
  { _id: 'Computer Hardware', count: 211 },
  { _id: 'e-Government', count: 209 },
  { _id: 'Speech Processing/Technology', count: 208 },
  { _id: 'Radar', count: 206 },
  { _id: 'Operation Planning and Scheduling System', count: 205 },
  { _id: 'Serious Games', count: 203 },
  { _id: 'Broadband Technologies', count: 203 }>
```

Figure 4.29: Query 13: result.

Query 14: Titles of articles cited in the bibliography

The following query retrieves the titles of all the articles cited in the bibliography of a specific article.

The original article, in which the bibliography is present in the form of references to other documents, can be seen in Figure 4.30.

The query uses the aggregation pipeline. The most relevant operations are the following:

- The equivalent of an SQL JOIN is performed using the `$lookup` operator, joining articles' references present in the bibliography with the corresponding article based on their `_ids`
- An `$unwind` is performed returning a temporary set of articles in which, for each article present in the `bibliography` array, a new document is created. There are now multiple documents with the same values for each field of the original document,

but each containing only one article of the ones present in the `bibliography` array.

For readability reasons, only the title of the original document and the title of the article cited in the bibliography are shown. The query can be seen in Figure 4.31, while the result is shown in Figure 4.32.

```
< { _id: '63776ddd15bc88e6627df47',
  title: 'Chiral Topological Insulators, Superconductors and other competing\n orders in three dimensions',
  bibliography:
    [ '63777073b372ec5c13bc54a1',
      '63776de0074e5f9d38061bd',
      '63777073e31058db9972c204',
      '637770ab916065b1fe8a3cbc' ] }
```

Figure 4.30: Query 14: before.

```
Atlas atlas-l7mawr-shard-0 [primary] journal_articles> db.articles.aggregate([
  {
    $match: { "_id": ObjectId('63776ddde00236845a2fb904') }
  },
  {
    $lookup: {
      from: "articles",
      localField: "bibliography",
      foreignField: "_id",
      as: "bibliography"
    },
    $unwind: "$bibliography"
  },
  {
    $project: { "title": 1, "bibliography.title": 1 }
  }
])
```

Figure 4.31: Query 14.

```
< { _id: '63776ddd15bc88e6627df47',
  title: 'Chiral Topological Insulators, Superconductors and other competing\n orders in three dimensions',
  bibliography: { title: 'The stable AR-quiver of a quantum complete intersection' } }
{ _id: '63776ddd15bc88e6627df47',
  title: 'Chiral Topological Insulators, Superconductors and other competing\n orders in three dimensions',
  bibliography: { title: 'Equations and syzygies of the first secant variety to a smooth curve' } }
{ _id: '63776ddd15bc88e6627df47',
  title: 'Chiral Topological Insulators, Superconductors and other competing\n orders in three dimensions',
  bibliography: { title: 'PCA consistency in high dimension, low sample size context' } }
{ _id: '63776ddd15bc88e6627df47',
  title: 'Chiral Topological Insulators, Superconductors and other competing\n orders in three dimensions',
  bibliography: { title: 'On the structure of non-full-rank perfect codes' } }
```

Figure 4.32: Query 14: result.

Query 15: Cited articles in chronological order

The following query returns the titles of the articles, ordered from the most recent to the oldest, cited in the bibliography of two selected articles. The `bibliography` arrays of the two articles before the operation can be seen in Figure 4.33. To obtain the result, the aggregation pipeline offered by the `aggregate()` function is used. The most relevant operations are the following:

- The equivalent of an SQL `JOIN` is performed using the `$lookup` operator, joining articles' references present in the bibliography with the corresponding article based on their `_ids`;

- an `$unwind` is performed returning a temporary set of articles in which, for each article present in the `bibliography` array, a new document is created. There are now multiple documents with the same values for each field of the original document, but each containing only one article of the ones present in the `bibliography` array;
- the `$sort` operation orders the article in inverse chronological order.

For readability reasons, only the title of the original document, the title of the article cited in the bibliography and its publication date are shown. The query can be seen in Figure 4.34, while the result is shown in Figures 4.35 and 4.36.

```
< { _id: '63776ddd2f7fb02b6ecfdf3',
  title: 'HD 77361: A new case of super Li-rich K giant with anomalous low 12C/13C ratio',
  bibliography:
    [ '63777073b1aeb06de1b7492',
      '63776dde07a64b8cb303d4e9',
      '63776dde6a9de277c1ce0ae1' ] }
{ _id: '63776ddde00236845a2fb904',
  title: '$F$-pure homomorphisms, strong $F$-regularity, and $F$-injectivity',
  bibliography:
    [ '63776dde84264e473ba7c413',
      '63776dde697c3f791c68642d',
      '63776dde6efbe362b4909aba',
      '637770ac8fd8b72787bb6363',
      '63776dde5d187cd65b423b2' ] }
```

Figure 4.33: Query 15: before.

```
Atlas atlas-l7mawr-shard-0 [primary] journal_articles> db.articles.aggregate([
  {
    $match: { $or: [ { "_id": ObjectId("63776ddde00236845a2fb904") }, { "_id": ObjectId("63776ddd2f7fb02b6ecfdf3") } ] }
  },
  {
    $lookup: {
      from: "articles",
      localField: "bibliography",
      foreignField: "_id",
      as: "bibliography"
    }
  },
  {
    $unwind: "$bibliography"
  },
  {
    $sort: { "bibliography.journal.date": -1 }
  },
  {
    $project: { "title": 1, "bibliography.title": 1, "bibliography.journal.date": 1 }
  }
])
```

Figure 4.34: Query 15.

```
< { _id: '63776ddde00236845a2fb904',
  title: '$F$-pure homomorphisms, strong $F$-regularity, and $F$-injectivity',
  bibliography:
  { title: 'Computing the multifractal spectrum from time series: An algorithmic\n approach',
    journal: { date: 2020-04-11T22:00:00.000Z } }
{ _id: '63776ddde00236845a2fb904',
  title: '$F$-pure homomorphisms, strong $F$-regularity, and $F$-injectivity',
  bibliography:
  { title: 'Spin dimers under staggered and random field in\n Cu$_2$Fe$_2$Ge$_4$S$_{13}$',
    journal: { date: 2015-09-13T22:00:00.000Z } }
{ _id: '63776dddf7fbf02b6ecfdf3',
  title: 'HD 77361: A new case of super Li-rich K giant with anomalous low 12C/13C\n ratio',
  bibliography:
  { title: 'Phenomenology of ESR in heavy fermion systems: Fermi liquid and\n non-Fermi liquid regime',
    journal: { date: 2009-12-14T23:00:00.000Z } }
{ _id: '63776ddde00236845a2fb904',
  title: '$F$-pure homomorphisms, strong $F$-regularity, and $F$-injectivity',
  bibliography:
  { title: 'Quantum-optical state engineering up to the two-photon level',
    journal: { date: 2009-07-13T22:00:00.000Z } }
{ _id: '63776dddf7fbf02b6ecfdf3',
  title: 'HD 77361: A new case of super Li-rich K giant with anomalous low 12C/13C\n ratio',
  bibliography:
  { title: 'Bell\\'s Inequalities: Foundations and Quantum Communication',
    journal: { date: 2006-01-08T23:00:00.000Z } }
{ _id: '63776dddf7fbf02b6ecfdf3',
  title: 'HD 77361: A new case of super Li-rich K giant with anomalous low 12C/13C\n ratio',
  bibliography:
  { title: 'Terrorism: Mechanisms of Radicalization Processes, Control of Contagion\n and Counter-Terrorist Measures',
    journal: { date: 2004-08-14T22:00:00.000Z } }
```

Figure 4.35: Query 15: result (1st part).

```
{ _id: '63776ddde00236845a2fb904',
  title: '$F$-pure homomorphisms, strong $F$-regularity, and $F$-injectivity',
  bibliography:
  { title: 'Incorporating characteristics of human creativity into an evolutionary\n art algorithm',
    journal: { date: 1990-10-30T23:00:00.000Z } }
{ _id: '63776ddde00236845a2fb904',
  title: '$F$-pure homomorphisms, strong $F$-regularity, and $F$-injectivity',
  bibliography:
  { title: 'On the magnetic equation of state in (2+1)-flavor QCD',
    journal: { date: 1973-03-24T23:00:00.000Z } }
```

Figure 4.36: Query 15: result (2nd part).

Query 16: Sections containing figures

The following query returns the title of the sections containing at least an image of a specific article. To obtain this result, we used the `aggregate` operator. To select the article, we use the `$match` operator, then to create a separate document each containing only one section of the original article, we use the `$unwind` operator, specifying the array to perform the unwind on. After that, we select the documents containing a figure in their section by using a `$match` operator combined with a `$exists` operator on the attribute `figure` of the section, and the return the title of those by using a `$project` operator.

```
Atlas atlas-lx2lpr-shard-0 [primary] SMBUD_Database> db.Articles.aggregate([
    {$match: {_id:ObjectId("63776ddd297f4328218ebc59")}},
    {$unwind:{"path":"$sections"}},
    {$match:{"sections.figure":{$exists:true}}},
    {$project:{"sections.title":1}}
])
```

Figure 4.37: Query 16.

```
< { _id: '63776ddd297f4328218ebc59',
  sections: { title: 'Simulation of equatorial von Neumann measurements on GHZ states using\n nonlocal resources' } }
{ _id: '63776ddd297f4328218ebc59',
  sections: { title: 'Chain enumeration of $k$-divisible noncrossing partitions of classical\n types' } }
```

Figure 4.38: Query 16: result.

Query 17: Authors affiliated to a company

The following query returns the number of authors affiliated to a specific company called "Letpro". To obtain this result, we used the `aggregate` operator. The query needs to be executed on the whole data set, but there is no need to execute the next operations on articles which do not contain at least an author with the correct `affiliation`, so first of all, we use the `$match` operator. After that, we use the `$unwind` operator to create separate documents each containing only one author among the ones of the original articles. To select only the authors with the correct `affiliation`, we use a `$match` operator on the `affiliation` attribute of the `authors`. Finally, to count the distinct `authors`, we decided to exploit the `$group` operator, grouping by `author`, and then the `$count` operator to return the corresponding integer.

```
Atlas atlas-lx2lpr-shard-0 [primary] SMBUD_Database> db.Articles.aggregate([
    {$match: {"authors.affiliation":"Letpro"}},
    {$unwind:{"path":"$authors"}},
    {$match: {"authors.affiliation":"Letpro"}},
    {$group:{"_id":"$authors"}},
    {$count:"number of authors with affiliation to Letpro"}
])
```

Figure 4.39: Query 17.

```
] )
< { 'number of authors with affiliation to Letpro': 25 }
```

Figure 4.40: Query 17: result.

Query 18: Subsections with figures

The following query returns the titles of the subsections containing at least an image of a specific article. To obtain this result, we use the `aggregate` operator. First of all, we select the article by using a `$match` operator, then, to divide the document into smaller documents each containing only one subsection, we used a combination of two `$unwind` operations, the first one on the `sections` and the second one on the `subsections` of each section. This is possible because the document based databases use a pipeline. As done in the query ? (unwind 1), we select the documents containing a figure in their subsection by using a `$match` operator combined with a `$exists` operator on the attribute `figure` of the subsection, and the return the title of those by using a `$project` operator.

```
Atlas atlas-lx2lpr-shard-0 [primary] SMBUD_Database> db.Articles.aggregate([
    {$match:{_id:ObjectId("63777073646da6e0070c5f8a")}},
    {$unwind:{"path":"$sections"}},
    {$unwind:{"path":"$sections.subsections"}},
    {$match:{"sections.subsections.figure":{$exists:true}}},
    {$project:{"sections.subsections.title":1}}
])
```

Figure 4.41: Query 18.

```
< { _id: '63777073646da6e0070c5f8a',
  sections: { subsections: { title: 'Higher-level canonical subgroups for p-divisible groups' } } }
{ _id: '63777073646da6e0070c5f8a',
  sections: { subsections: { title: 'A one-dimensional Keller-Segel equation with a drift issued from the\n boundary' } } }
```

Figure 4.42: Query 18: result.

A | Appendix A

A.1. Article type 1

```
[
  '{{ repeat(2000)}},
{
  _id:'{{ objectId()}}',
  title: '{{ lorem(1, "sentence")}}',
  authors:[
    '{{ repeat(1, 7)}},
    {
      name: '{{ firstName()}} {{surname()}}',
      affiliation: '{{ company()}}',
      email : '{{email()}}',
      bio : '{{ lorem(1, "sentence")}}'
    }
  ],
  keywords: [
    '{{ repeat(3)}},
    '{{ lorem(1, "words")}}'
  ],
  journal: {
    name: '{{ city()}}',
    volume : '{{ integer(1, 12)}}',
    number: '{{ integer(1, 100)}}',
    date: '{{ date(new Date(1970, 0, 1), new Date(2022, 11, 11))}}'
  }
]
```

```

"YYYY-MM-dd")} }',
pages: '{{ integer(100, 200)} } - {{ integer(201, 400)} }'
} ,  
  

abstract: '{{ lorem(1, "sentence")}}',  
  

sections: [
//section 1: paragraphs only (no figures, no subsections)
{
    title: '{{ lorem(1, "sentence")}}',  
  

    paragraphs: [
        '{{ repeat(1,3)}}',
        '{{ lorem(1, "sentence")}}'
    ]
},  

//section 2: paragraphs, figures (no subsections)
{
    title: '{{ lorem(1, "sentence")}}',  
  

    paragraphs: [
        '{{ repeat(1,3)}}',
        '{{ lorem(1, "sentence")}}'
    ],  
  

    figure:
    {
        url: 'https://ccs-aerospaziale.polimi.it/wp-content/
uploads/2020/02/01_Polimi_centrato_COL_positivo.png',
        caption: 'Polimi logo'
    }
},  
  

//section 3: paragraphs, figures and subsections
{
    title: '{{ lorem(1, "sentence")}}',  


```

```
paragraphs: [
    '{ { repeat(1,3) } }',
    '{ { lorem(1, "sentence") } }'
],  
  
figure: [
    {
        url: 'https://ccs-aerospaziale.polimi.it/wp-content/
uploads/2020/02/01_Polimi_centrato_COL_positivo.png',
        caption: 'Polimi logo'
    }
],  
  
subsections: [
    '{ { repeat(1, 2) } }',
    {
        title: '{ { lorem(1, "sentence") } }',  
  
        paragraphs: [
            '{ { repeat(1,3) } }',
            '{ { lorem(1, "sentence") } }'
        ],
  
  
        figure: [
            {
                url: 'https://ccs-aerospaziale.polimi.it/wp-content/
                /uploads/2020/02/
                01_Polimi_centrato_COL_positivo.png',
                caption: 'Polimi logo'
            }
        ]
    }
],  
  
bibliography: [
```

```

'{{ repeat(3,5)}},
'{{ integer(100)}}
]
}
]
```

A.2. Article type 2

```

[
'{{ repeat(2000)}},
{
  _id: '{{ objectId()}}',
  title: '{{ lorem(1, "sentence")}}',
  authors:[
    '{{ repeat(1, 7)}},
    {
      name: '{{ firstName()}} {{surname()}}',
      affiliation: '{{ company()}}',
      email : '{{ email()}}',
      bio : '{{ lorem(1, "sentence")}}'
    }
  ],
  keywords: [
    '{{ repeat(3)}},
    '{{ lorem(1, "words")}}'
  ],
  journal: {
    name: '{{ city()}}',
    volume : '{{ integer(1, 12)}}',
    number: '{{ integer(1, 100)}}',
    date: '{{ date(new Date(1970, 0, 1), new Date(2022, 11, 11),
      "YYYY-MM-dd")}}',
    pages: '{{ integer(100, 200)}} - {{integer(201, 400)}}
```

```
 } ,  
 abstract: '{\{lorem(1, "sentence")\}}' ,  
  
 sections: [  
 //section 1: paragraphs , figures and subsections  
 {  
  
 title: '{\{lorem(1, "sentence")\}}' ,  
  
 paragraphs: [  
 '{\{repeat(1,3)\}}' ,  
 '{\{lorem(1, "sentence")\}}'  
 ] ,  
  
 figure: [  
 {  
 url: 'https://ccs-aerospaziale.polimi.it/wp-content/  
 uploads/2020/02/01_Polimi_centrato_COL_positivo.png' ,  
 caption: 'Polimi logo'  
 }  
 ] ,  
  
 subsections: [  
 '{\{repeat(1, 2)\}}' ,  
 {  
 title: '{\{lorem(1, "sentence")\}}' ,  
  
 paragraphs: [  
 '{\{repeat(1,3)\}}' ,  
 '{\{lorem(1, "sentence")\}}'  
 ] ,  
  
 figure:  
 {  
 url: 'https://ccs-aerospaziale.polimi.it/  
 wp-content/uploads/2020/02'
```

```

        '/01_Polimi_centrato_COL_positivo.png',
        caption: 'Polimi logo'
    }
}
],
},
//section 2: paragraphs, figures and subsections (without figures)
{
    title: '{{ lorem(1, "sentence")}}',
    paragraphs: [
        '{{ repeat(1,3)}}',
        '{{ lorem(1, "sentence")}}'
    ],
    figure: [
        {
            url: 'https://ccs-aerospaziale.polimi.it/wp-content/uploads/2020/02/01_Polimi_centrato_COL_positivo.png',
            caption: 'Polimi logo'
        }
    ],
    subsections: [
        '{{ repeat(1, 2)}}',
        {
            title: '{{ lorem(1, "sentence")}}',
            paragraphs: [
                '{{ repeat(1,3)}}',
                '{{ lorem(1, "sentence")}}'
            ]
        }
    ]
}
}
```

```

] ,
bibliography: [
  '{{ repeat(3,5)}},
  '{{ integer(100)}}
]
}
]
```

A.3. Article type 3

```

[
  '{{repeat(2000)}},
{
  _id: '{{objectId()}}',
  title: '{{lorem(1, "sentence")}}',
  authors: [
    '{{repeat(1, 7)}}',
    {
      name: '{{firstName()}} {{surname()}}',
      affiliation: '{{company()}}',
      email : '{{email()}}',
      bio : '{{lorem(1, "sentence")}}'
    }
  ],
  keywords: [
    '{{repeat(3)}}',
    '{{lorem(1, "words")}}'
  ],
  journal: {
    name: '{{city()}}',
    volume : '{{integer(1, 12)}}',
    number: '{{integer(1, 100)}}',
  }
]
```

```

date: '{{date(new Date(1970, 0, 1), new Date(2022, 11, 11),
"YYYY-MM-dd")}}',
pages: '{{integer(100, 200)} - {{integer(201, 400)}}}'
},

abstract: '{{lorem(1, "sentence")}}',

sections: [
//section 1: paragraphs only, no figures, no subsection
{
  title: '{{lorem(1, "sentence")}}',
  // testo opzionale
  paragraphs: [
    '{{repeat(1,3)}',
    '{{lorem(1, "sentence")}}'
  ]
},
//section 2: paragraphs only, no figures, no subsections
{
  title: '{{lorem(1, "sentence")}}',
  paragraphs: [
    '{{repeat(1,3)}',
    '{{lorem(1, "sentence")}}'
  ]
},
//section 3: paragraphs, figures, subsections
{
  title: '{{lorem(1, "sentence")}}',
  paragraphs: [
    '{{repeat(1,3)}',
    '{{lorem(1, "sentence")}}'
  ],
}
]

```

```
figure: [
  {
    url: 'https://ccs-aerospaziale.polimi.it/wp-content/
uploads/2020/02/01_Polimi_centrat o_COL_positivo.png',
    caption: 'Polimi logo'
  }
],

subsections: [
  '{{repeat(1, 2)}},
  {
    title: '{{lorem(1, "sentence")}}',
    paragraphs: [
      '{{repeat(1,3)}},
      '{{lorem(1, "sentence")}}'
    ]
  },
  figure:
  {
    url: 'https://ccs-aerospaziale.polimi.it/
wp-content/uploads/2020/02
/01_Polimi_centrat o_COL_positivo.png',
    caption: 'Polimi logo'
  }
]
}

bibliography: [
  '{{repeat(3,5)}},
  '{{integer(100)}}
]
```


B | Appendix B

B.1. Python code

```

import json
import csv
import random

def read_write_json():
    title = 1
    text = 1
    journals = 1
    abstracts = 1
    with open("article_type1.json", "r", encoding='utf-8') as
read_articles_json:
        json_data = json.load(read_articles_json)
        for i in range(2000): # cambiare con numero articles
            json_data[i]['keywords'] = get_keywords()
            json_data[i]['abstract'] = get_abstract(abstracts)
            abstracts = (abstracts + 1) % 67186
            json_data[i]['journal'][ 'name' ] = get_journal(journals)
            journals = (journals + 1) % 20254
            json_data[i][ 'title' ] = get_title(title)
            title = title + 1

            sections_length = len(json_data[i][ 'sections' ])
            for j in range(sections_length):
                json_data[i][ 'sections' ][j][ 'title' ] = get_title(title)
                title = title + 1
            paragraphs_length = len(json_data[i][ 'sections' ])

```

```

[j]['paragraphs'])
for k in range(paragraphs_length):
    json_data[i]['sections'][j]['paragraphs'][k] =
        get_text(text)
    text = (text + 1) % 34999
if j == 2:
    subsections_length = len(json_data[i]['sections']
[j]['subsections'])
    for a in range(subsections_length):
        json_data[i]['sections'][j]['subsections'][a]['title'] =
            get_title(title)
        title = (title + 1) % 67186
        subsection_paragraphs_len = len(json_data[i]['sections']
[j]['subsections'][a]['paragraphs']))
        for m in range(subsection_paragraphs_len):
            json_data[i]['sections'][j]['subsections'][a]
['paragraphs'][m] = get_text(text)
            text = (text + 1) % 34999

bib_length = len(json_data[i]['bibliography'])
for k in range(bib_length):
    json_data[i]['bibliography'][k] = get_id()

with open('output_articles_type1.json', 'w', encoding='utf-8') as write_articles_json:
    json.dump(json_data, write_articles_json)
    write_articles_json.close()
    print(str(i) + ".done.")
read_articles_json.close()
write_articles_json.close()

with open("article_type2.json", "r", encoding='utf-8') as read_articles_json:
    json_data = json.load(read_articles_json)
    for i in range(2000):

```

```

json_data[i]['keywords'] = get_keywords()
json_data[i]['abstract'] = get_abstract(abstracts)
abstracts = (abstracts + 1) % 67186
json_data[i]['journal']['name'] = get_journal(journals)
journals = (journals + 1) % 20254
json_data[i]['title'] = get_title(title)
title = (title + 1) % 67186

sections_length = len(json_data[i]['sections'])
for j in range(sections_length):
    json_data[i]['sections'][j]['title'] =
        get_title(title)
    title = (title + 1) % 67186
    paragraphs_length = len(json_data[i]['sections']
        [j]['paragraphs'])
    for k in range(paragraphs_length):
        json_data[i]['sections'][j]['paragraphs'][k] =
            get_text(text)
        text = (text + 1) % 34999
    subsections_length = len(json_data[i]['sections']
        [j]['subsections'])
    for a in range(subsections_length):
        new_title = get_title(title)
        json_data[i]['sections'][j]['subsections'][a]
            ['title'] = new_title
        title = (title + 1) % 67186
        subsection_paragraphs_len = len(json_data[i]
            ['sections'][j]['subsections'][a]['paragraphs'])
        for m in range(subsection_paragraphs_len):
            json_data[i]['sections'][j]['subsections'
                [a]['paragraphs'][m]] = get_text(text)
            text = (text + 1) % 34999

bib_length = len(json_data[i]['bibliography'])
for k in range(bib_length):
    json_data[i]['bibliography'][k] = get_id()

```

```

with open('output_articles_type2.json', 'w',
encoding='utf-8') as write_articles_json:
    json.dump(json_data, write_articles_json)
    write_articles_json.close()
    print(str(i) + " done.")
read_articles_json.close()
write_articles_json.close()

with open("article_type3.json", "r", encoding='utf-8') as
read_articles_json:
    json_data = json.load(read_articles_json)
    for i in range(2000):
        json_data[i]['keywords'] = get_keywords()
        json_data[i]['abstract'] = get_abstract(abstracts)
        abstracts = (abstracts + 1) % 67186
        json_data[i]['journal']['name'] = get_journal(journals)
        journals = (journals + 1) % 20254
        json_data[i]['title'] = get_title(title)
        title = (title + 1) % 67186

        sections_length = len(json_data[i]['sections'])
        for j in range(sections_length):
            json_data[i]['sections'][j]['title'] = get_title(title)
            title = (title + 1) % 67186
            paragraphs_length = len(json_data[i]['sections']
[j]['paragraphs'])
            for k in range(paragraphs_length):
                json_data[i]['sections'][j]['paragraphs'][k] =
get_text(text)
                text = (text + 1) % 34999
        if j == 2:
            subsections_length = len(json_data[i]['sections'][j]
['subsections'])
            for a in range(subsections_length):
                json_data[i]['sections'][j]['subsections'][a]
['title'] = get_title(title)
                title = (title + 1) % 67186

```

```

subsection_paragraphs_len = len(json_data[i]
['sections'][j]['subsections'][a]['paragraphs'])
for m in range(subsection_paragraphs_len):
    json_data[i]['sections'][j]['subsections'][a]
    ['paragraphs'][m] = get_text(text)
    text = (text + 1) % 34999

bib_length = len(json_data[i]['bibliography'])
for k in range(bib_length):
    json_data[i]['bibliography'][k] = get_id()

with open('output_articles_type3.json', 'w',
encoding='utf-8') as write_articles_json:
    json.dump(json_data, write_articles_json)
    write_articles_json.close()
    print(str(i) + " done.")
read_articles_json.close()
write_articles_json.close()

def get_journal(index):
    with open('journals.csv', encoding='Latin-1') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        rows = list(csv_reader)
        ret = rows[index][0]
        csv_file.close()
    return ret

def get_abstract(index):
    with open('abstracts.csv', encoding='utf-8') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        rows = list(csv_reader)
        ret = rows[index][0]
        csv_file.close()
    return ret

```

```

def get_text(index):
    with open('reviews.csv', encoding='utf-8') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        rows = list(csv_reader)
        ret = rows[index][0]
        csv_file.close()
    return ret

def get_keywords():
    text_file = open("keywords.txt", "r")
    lines = text_file.readlines()
    text_file.close()
    rnd_1 = random.randint(1, 98)
    rnd_2 = random.randint(1, 98)
    while rnd_1 == rnd_2:
        rnd_2 = random.randint(1, 98)
    rnd_3 = random.randint(1, 98)
    while rnd_3 == rnd_1 or rnd_3 == rnd_2:
        rnd_3 = random.randint(1, 98)

    out_1 = lines[rnd_1][:len(lines[rnd_1]) - 1]
    out_2 = lines[rnd_2][:len(lines[rnd_2]) - 1]
    out_3 = lines[rnd_3][:len(lines[rnd_3]) - 1]

    text_file.close()

    return [out_1, out_2, out_3]

def get_title(index):
    with open('titles.csv', encoding='utf-8') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        rows = list(csv_reader)
        ret = rows[index][0]
        csv_file.close()

```

```
    return ret

def get_id():
    article_type = random.randint(1,3);
    with open("article_type" + str(article_type) + ".json", "r",
encoding='utf-8') as read_articles_json:
        json_data = json.load(read_articles_json)
        idx = random.randint(0, len(json_data)-1)
        ret_idx = json_data[idx]['_id']
        read_articles_json.close()
    return ret_idx

def main():
    read_write_json()

if __name__ == '__main__':
    main()
```


List of Figures

2.1	ER diagram.	3
3.1	Importation command.	8
3.2	First schema.	8
3.3	Query 1.	11
3.4	Output pf query 2.	11
3.5	Query 2.	12
3.6	Output pf query 2.	12
3.7	Query 3.	12
3.8	Output of query 3.	13
3.9	Query 4.	13
3.10	Output of query 4.	13
3.11	Query 5.	14
3.12	Output of query 5.	14
3.13	Query 1.	15
3.14	Output of query 1.	15
3.15	Query 2.	15
3.16	Output of query 2.	15
3.17	Query 3.	16
3.18	Output of query 3.	16
3.19	Query 4.	17
3.20	Output of query 4.	17
3.21	Query 5.	17
3.22	Output of query 5.	18
3.23	Query 6.	18
3.24	Output of query 6.	18
3.25	Query 7.	19
3.26	Output of query 7.	19
3.27	Query 8.	20
3.28	Query 9.	20

3.29 Output of query 9	20
3.30 Query 10.	21
4.1 Example of a document.	24
4.2 Query 1.	27
4.3 Query 1: result.	27
4.4 Query 2.	28
4.5 Query 2: result.	28
4.6 Query 3.	28
4.7 Query 3: result.	29
4.8 Query 4.	29
4.9 Query 4: result.	30
4.10 Query 5: before.	30
4.11 Query 5.	31
4.12 Query 5: result.	31
4.13 Query 6: before.	32
4.14 Query 6.	32
4.15 Query 6: result.	32
4.16 Query 7.	33
4.17 Query 7: result.	33
4.18 Query 8.	34
4.19 Query 8: result.	34
4.20 Query 9.	35
4.21 Query 9: result.	35
4.22 Query 10.	36
4.23 Query 10: result.	36
4.24 Query 11.	37
4.25 Query 11: result.	37
4.26 Query 12.	38
4.27 Query 12: result.	38
4.28 Query 13.	39
4.29 Query 13: result.	39
4.30 Query 14: before.	40
4.31 Query 14.	40
4.32 Query 14: result.	40
4.33 Query 15: before.	41
4.34 Query 15.	41

4.35 Query 15: result (1st part).	42
4.36 Query 15: result (2nd part).	42
4.37 Query 16.	43
4.38 Query 16: result.	43
4.39 Query 17.	43
4.40 Query 17: result.	43
4.41 Query 18.	44
4.42 Query 18: result.	44

List of Tables

3.1 Relationships	10
-----------------------------	----

