POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Roberto Macaccaro 10615972**
Elena Montemurro 10627677
Marta Radaelli 10657046
Luca Rondini 10628339
Francesco Scandale 10616610

Group Number: **24**

Academic Year: 2022-2023

Contents

1 | First assignment

1.1. Assignment specifications

1.1.1. Assignment description

The purpose of this assignment is to effectively manage the data of a bibliographic database while keeping track of all published scientific publications, their characteristic, citations, authors, editors, and publishers. As these relationships are the key point, we developed a graph database which allows to manage and visualize the connection and efficiently retrieve any significant information.

1.1.2. Hypotheses

The assumptions taken into consideration are the following:

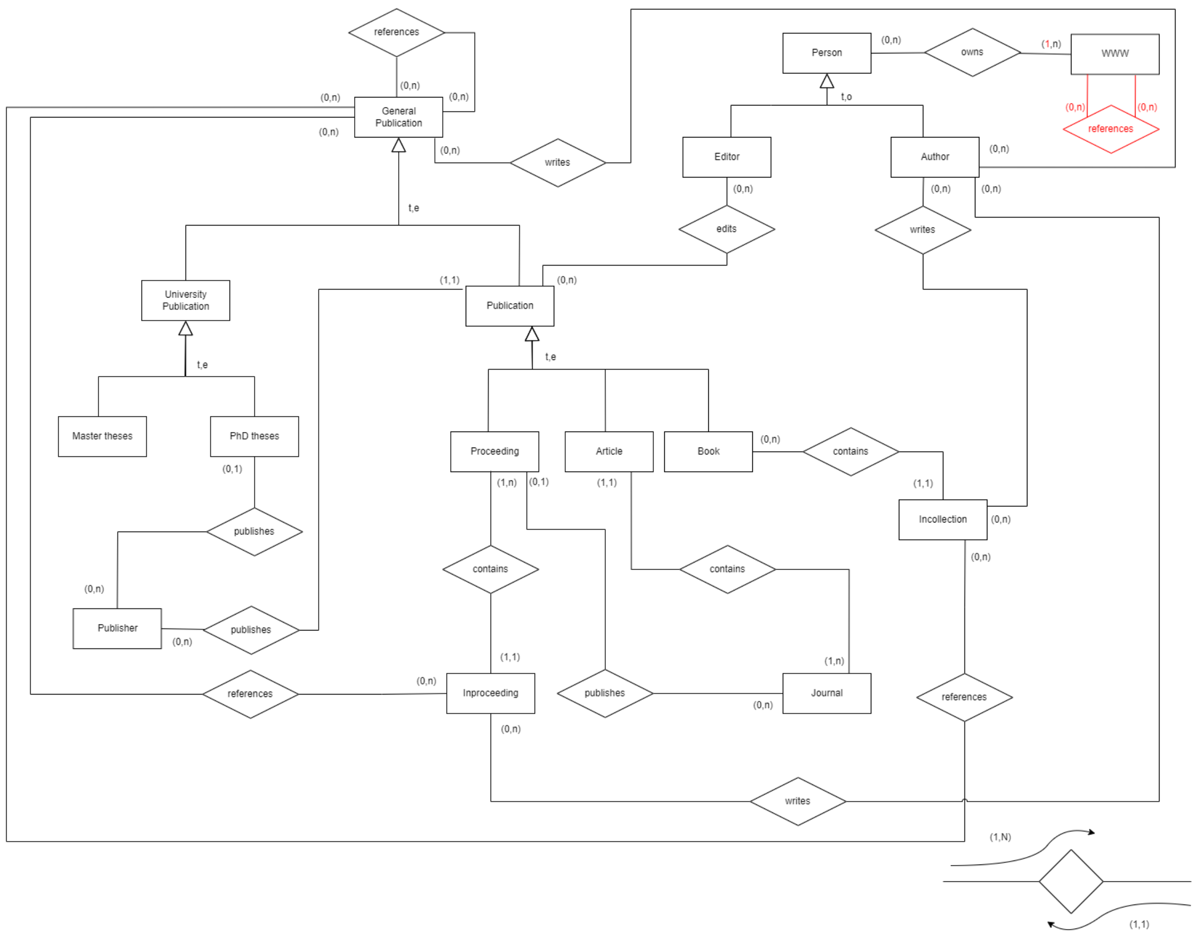
- PEOPLE
  - A *person* can be both an *author* and an *editor*;
  - A *person* can have multiple renditions of his/her name;
  - A *person* can change name, but records for the previous names are kept;
  - People can have personal *web pages*;
  - In order to distinguish *homonym people*, an incremental four digits number is appended to the name. The number is part of the *url*, but not printed with the name.
- GENERAL PUBLICATIONS
  - All *general publications, incollections* and *inproceedings* can reference each other;
  - *Publications* can have multiple *authors* and *editors*;
  - *University publications* do not have *editors*;

  - An *inproceeding* and an *incollection* do not have a dedicated editor;
  - A *book* can have several *incollections* authored by different people;
  - A *book* with authored *incollections* cannot have its own *authors*;
  - A *proceeding* can have several *inproceedings* authored by different people;
  - A *proceeding* with authored *inproceedings* cannot have its own *authors*;
  - A *proceeding* can have *editors*, but they are often the conference organizers;
  - A *proceeding* can be published in a *journal* or as a dedicated *volume*;
  - An *article* is published in a *journal.*

- PUBLISHER
  - *Proceedings, books, articles* and *PhD theses* can all have a *publisher*;
  - *Master theses* do not have *publishers.*

1.1.3. Conceptual model

Here is the Entity-Relationship model of our database:



To avoid creating a chaotic schema, the attributes were not added to the general ER model just presented, but only to smaller scale partial diagrams. The Entity-Relationship model contains different entities connected to each other through various relationships. Note that dblp publication records were inspired by the original BibTeX syntax:

- *PERSON*: the central element of dblp, an author and/or editor of the below publications;
- *WWW*: the web page of an author. Note that there are very few records in the dblp database for this entity;
- *PUBLISHER*: the entity responsible of publishing PhDs, articles, books and proceedings;
- *ARTICLE*: an editorial or peer-reviewed piece;

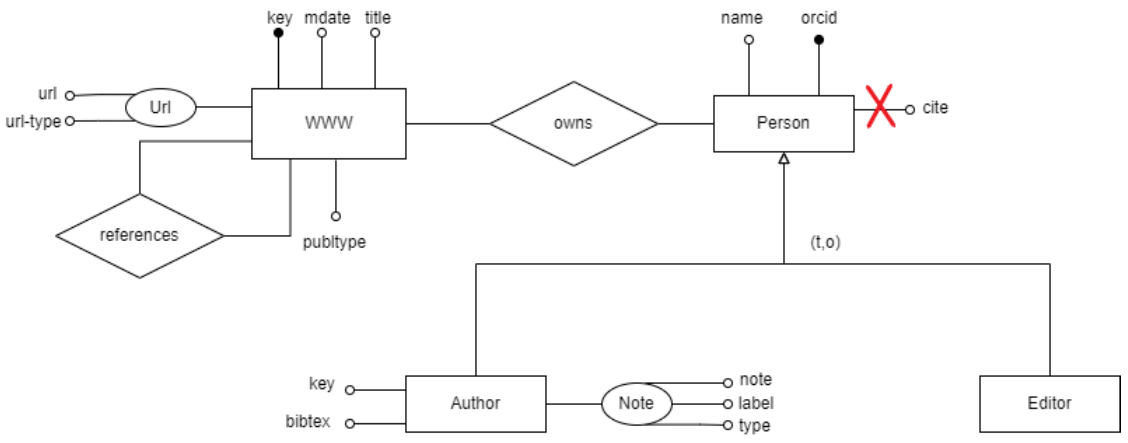
- *BOOKS*: an authored monograph or an edited collection of works.
- *INCOLLECTIONS*: a titled section or chapter in a book;
- *PROCEEDINGS*: the proceedings volume of a conference or workshop;
- *INPROCEEDINGS*: a paper in a conference or workshop published as part of a proceeding;
- *PhD THESES*: a PhD thesis;
- *MASTER THESES*: a Master's thesis. Note that there are very few records in the dblp database;
- *JOURNAL*: entity were articles or peer-reviewed proceedings can be published.

The entities *general publications, publications* and *university publications* of the hierarchy were created to better divide relationships and aggregate common attributes. This was done after performing a careful check and analysis of the database's xml file in order to better understand its complex structure and the meaning of the attributes.

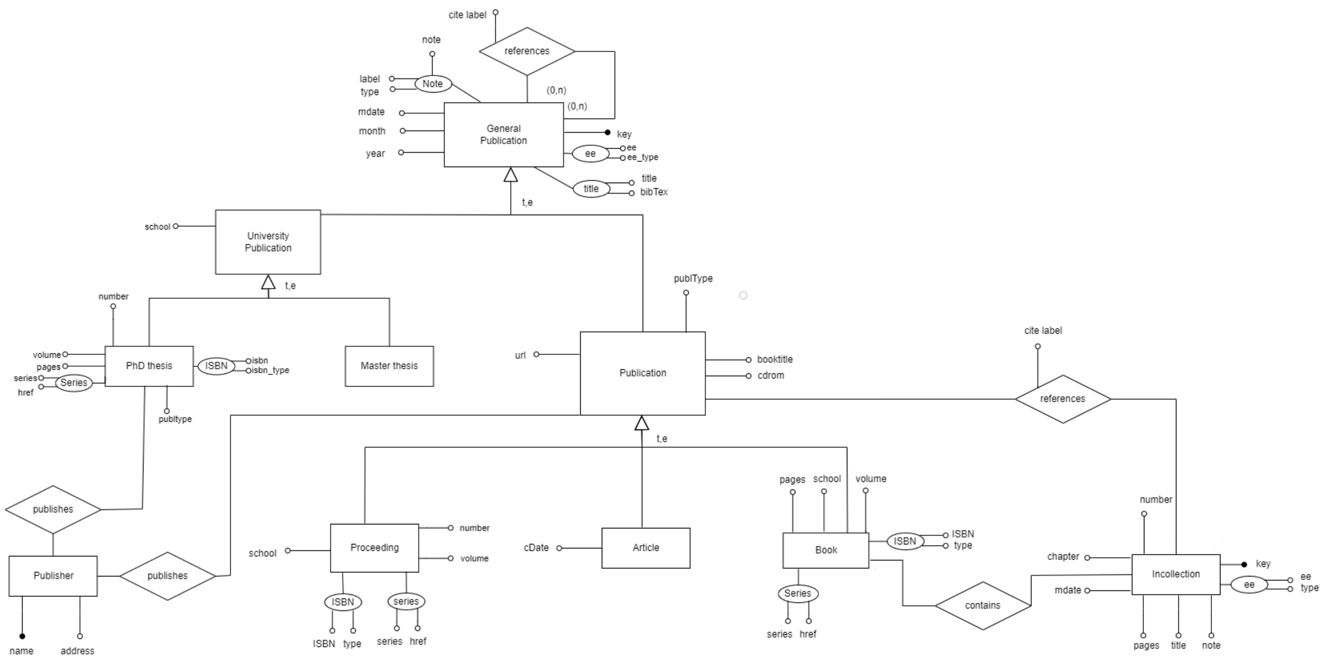
Among the entities previously described, some relations hold. In particular:

- *WRITES*: simply relates an author to its own work;
- *EDITS*: relates and editor to everything it has edited;
- *OWNS*: simply relates a person to all its websites;
- *PUBLISHES*: relates a publishing house to its own publications;
- *REFERENCES*: specifies what was cited in a publication;
- *CONTAINS*: relates an article to its journal, an inproceeding to its proceeding and an incollection to its book.

On the following pages there are the partial schemas complete of all the attributes with their descriptions. Note that, if an attribute has already been described for another entity and has no appreciable differences, its definition is omitted to avoid redundancy.



- PERSON
  - *OrcId*: the author's persistent identifier;
  - ~~*Crossref*: when we get additional info on a author that already has a www reference, the key stays the same, but the url changes and a new reference is added. We register that the two references are equivalent by using this attribute;~~
- AUTHOR
  - *Note* and *note - type/label*: are used to give further useful information about the author, for example its affiliation to an organization. Type is usually controlled vocabulary, while label is free-text;
  - *Bibtex*: stores the author name in Name {Surname} format.
  - *key*: a DBLP numeric identifier;
- WWW
  - *Key*: the ID of the author's record;
  - *url*: contains the location of the home page;
  - *Title*: describes the kind of webpage;
  - *PublType*: is used when the author is not a physical person, but an association of some sort;
  - *Mdate*: is the date of the last modification of the record.



- PUBLISHER
  - *Address*: usually just the city of the headquarters, but can also contain the full address;
- GENERAL PUBLICATION
  - *year*: year of publication or creation;
  - *month*: month of publication or creation;
  - *ee*: provides the DOI (digital object identifier) used to permanently identify the publication and link to it on the web;
  - *Title - bibtex*: contains the title in BibTex format.
  - *Cite - label*: provides extra information on the citation.(in the references relationship)
- PhD THESIS
  - *School*: the school where the thesis was written;
  - *Series* and *Series - href*: the series of publications the work was published in. The optional attribute href contains the local URL of the main page of the series;

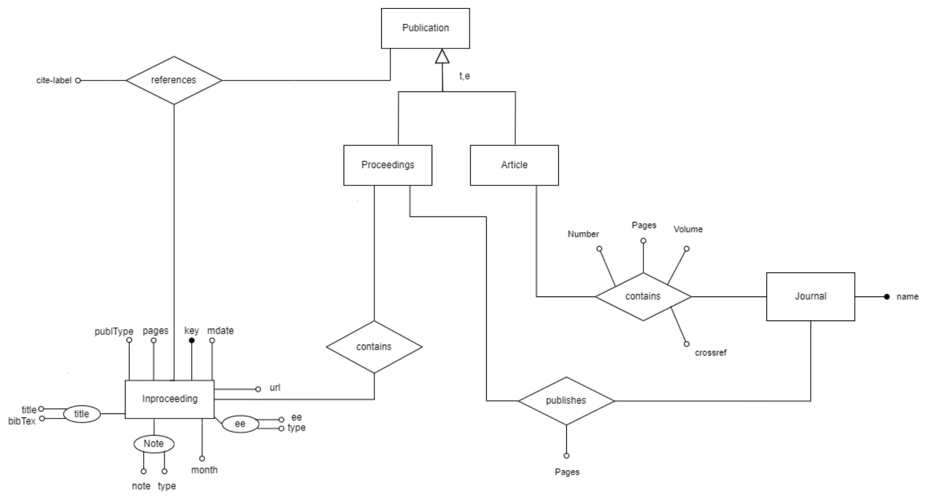
  - *Number*: the issue number;
  - *Volume*: the number of the volume the thesis appears in;
  - *Pages*: can represent the location of the publication if it consists of only one page, its total number of pages (usually in electronic publications) or the *from-to* range of pages it occupies.

- PUBLICATION
  - *Booktitle*: usually contains a shortened version of the title;
  - *Publtype*: provides additional information, usually indicates that the work was withdrawn by the author(s);
  - *Url*: contains the location of the publication web page.

- PROCEEDING
  - ~~*Key*: identifier of the publication, specifies if it was published as an independent volume or in a journal;~~
  - *School*: an institution associated to the author of the publication;
  - *Volume*: if series appears along with the volume attribute, volume is interpreted as the numbering of the series. If volume appears on its own, it's the number of the volume the proceeding appears in.

- ARTICLE
  - *cDate*: creation date of the article.

- INCOLLECTION
  - ~~*Volume*: the number of the volume the incollection appears in.~~
  - *Chapter*: the number of the chapter the incollection appears in.



- JOURNAL
  - *Number* and *Volume* (connected to article): specify the issue of the journal.

1.2. Graph database

1.2.1. Sample dataset and data preparation

The official dblp database was selected from a pool of suitable sample datasets. The original `XML` file was converted to `CSV` using the corresponding `dblp.dtd` file and a parser. In order to import the outputs into Neo4J, the files were put into the *import* folder of the Neo4J-Community directory. To upload data in `CSV` format the `LOAD CSV` command can be used, but due to the large amount of files, a quicker command line approach was exploited.

1. Load the nodes and the relationships created using the parser:

```
neo4j-admin import --database smbud --delimiter=";" --ignore-extra-
    columns --array-delimiter="|" --nodes=Author="import/
    output_author.csv" --nodes=Book="import/output_book_header.csv,
    import/output_book.csv" --nodes=Incollection="import/
    output_incollection_header.csv,import/output_incollection.csv"
    --nodes=Article="import/output_article_header.csv,import/
    output_article.csv" --nodes=Inproceeding="import/
    output_inproceedings_header.csv,import/output_inproceedings.csv"
     --nodes=Proceeding="import/output_proceedings_header.csv,import
    /output_proceedings.csv" --nodes=PhdThesis="import/
    output_phdthesis_header.csv,import/output_phdthesis.csv" --nodes
    =MasterThesis="import/output_mastersthesis_header.csv,import/
    output_mastersthesis.csv" --nodes=WWW="import/output_www_header.
```

```
csv,import/output_www.csv" --nodes=Editor="import/output_editor.
csv" --nodes=Journal="import/output_journal.csv" --nodes=
Publisher="import/output_publisher.csv" --relationships=EditedBy
="import/output_editor_edited_by.csv" --relationships=
PublishedIn="import/output_journal_published_in.csv" --
relationships=PublishedBy="import/output_publisher_published_by.
csv" --relationships=AuthoredBy="import/
output_author_authored_by.csv"
```

A couple of relationships were impossible to get via the parser used, so they needed to be created manually.

1. Create the CONTAINS relationship between books and incollections:

```
MATCH (a:Incollection),(b:Book)
WHERE a.crossref=b.key
CREATE (b)-[r:Contains]->(a)
RETURN type(r)
```

2. A similar query can be also done for the relationship between proceedings and inproceedings. The number of relationships to be created was huge, therefore a periodic iterate command from the APOC library was used:

```
CALL apoc.periodic.iterate(
  "MATCH (p:Proceeding), (i:Inproceeding) WHERE i.crossref=p.key
   RETURN p,i",
  "MERGE (p)-[:MadeOf]->(i)",
  {batchSize:10000, parallel:true})
```

Some attributes were removed as they had been used to create relationships and became redundant in the graph database. For example, the name of the author appeared in both the *Publication* and the *Author* nodes, as generated in the CSV by the parser, instead of being only in the Author nodes.

- Example command to remove the attribute author from books:

```
MATCH (b:Book) REMOVE b.author
```

Now the database is ready. The commands and queries in section 2.3 were developed to provide an example of usage of the graph database implementation.

1.2.2. Graph Diagram

To lighten a bit the import process, some less important or extremely rare attributes were dropped and will not appear in the following descriptions.

1. **Nodes**

   - Article: label identifying all articles in the database (editorial or peer-reviewed). It's connected to Author, Editor, Publisher, Journal. Attributes:
     - article: progressive number for all articles;
     - key: path in the dblp file system to find the article;
     - mdate: date of the last modification of the record;
     - publtype: kind of publication;
     - title: title of the article;
     - volume: specifies the journal issue the article can be found in;
     - year: when the article was published.
   - Author: label identifying all authors of at least one publication. It's connected to Article, Book, Incollection, Inproceeding, MasterThesis, PhdThesis, Proceeding, WWW. Attributes:
     - author: name of the author;
     - author-orcid: the author's persistent identifier.
   - Book: label identifying all books. It's connected to Author, Editor, Incollection, Publisher. Attributes:
     - book: progressive number for all books;
     - booktitle: title of the book (optional field);
     - ee: electronic edition of the book;
     - isbn: isbn code of the book;
     - key: path in the dblp file system to find the article;
     - mdate: date of the last modification of the record;
     - title: title of the book (required field);
     - url: link to the page in the dblp website;
     - year: when the book was published.
   - Editor: label identifying all editors who edited at least one publication. It's connected to Article, Book, Inproceeding, Proceeding, WWW. Attributes:

  - editor: name of the editor;
  - editor-orcid: the editor's persistent identifier.

- Incollection: label identifying all incollections present in the database. It's connected to Author, Book. Attributes:
  - booktitle: title of its book (optional field);
  - ee: electronic edition of the incollection;
  - incollection: progressive number for all incollections;
  - key: path in the dblp file system to find the incollection;
  - mdate: date of the last modification of the record;
  - publtype: kind of publication;
  - title: title of the incollection (required field);
  - url: link to the page in the dblp website;
  - year: when the incollection was published.
- Inproceeding: label identifying all publications inside a proceeding. It's connected to Author, Editor, Proceeding. Attributes:
  - booktitle: title of its proceeding;
  - ee: electronic edition of the inproceeding;
  - inproceeding: progressive number for all inproceedings;
  - key: path in the dblp file system to find this inproceeding;
  - mdate: date of the last modification of the record;
  - pages: where the inproceeding can be found within its proceeding;
  - title: title of the inproceeding;
  - url: relative link to the page in the dblp website;
  - year: when the inproceeding was published.
- Journal: label identifying all journals where some publication is published. It's connected to Article and Proceeding. Attributes:
  - journal: name of the journal.

- MasterThesis: label identifying all the master thesis published. It is connected to Author. Attributes:
  - ee: electronic edition of the master thesis;
  - key: path in the dblp file system to find the master thesis;
  - mastersthesis: progressive number for all master theses;
  - mdate: date of the last modification of the record;
  - note: additional notes;
  - school: an institution associated to the author of the publication;
  - title: title of the work;
  - year: year of publication of the thesis.
- PhdThesis: label identifying all the PhD thesis published. It is connected to Author and Publisher. Attributes:
  - ee: electronic edition of the PhD thesis;
  - isbn: isbn code of the thesis;
  - key: path in the dblp file system to find the PHD thesis;
  - mdate: date of the last modification of the record;
  - month: month of publication;
  - note: additional notes;
  - pages: number of pages of the work;
  - phdthesis: progressive number for all phd theses;
  - publtype: type of the publication informal, encyclopedia, ...;
  - school: an institution associated to the author of the publication;
  - series: name of the series the work belongs to;
  - series href: path in the dblp file system to find the series;
  - title: title of the work;
  - volume: number of volume if there are more than one;
  - year: year of publication.

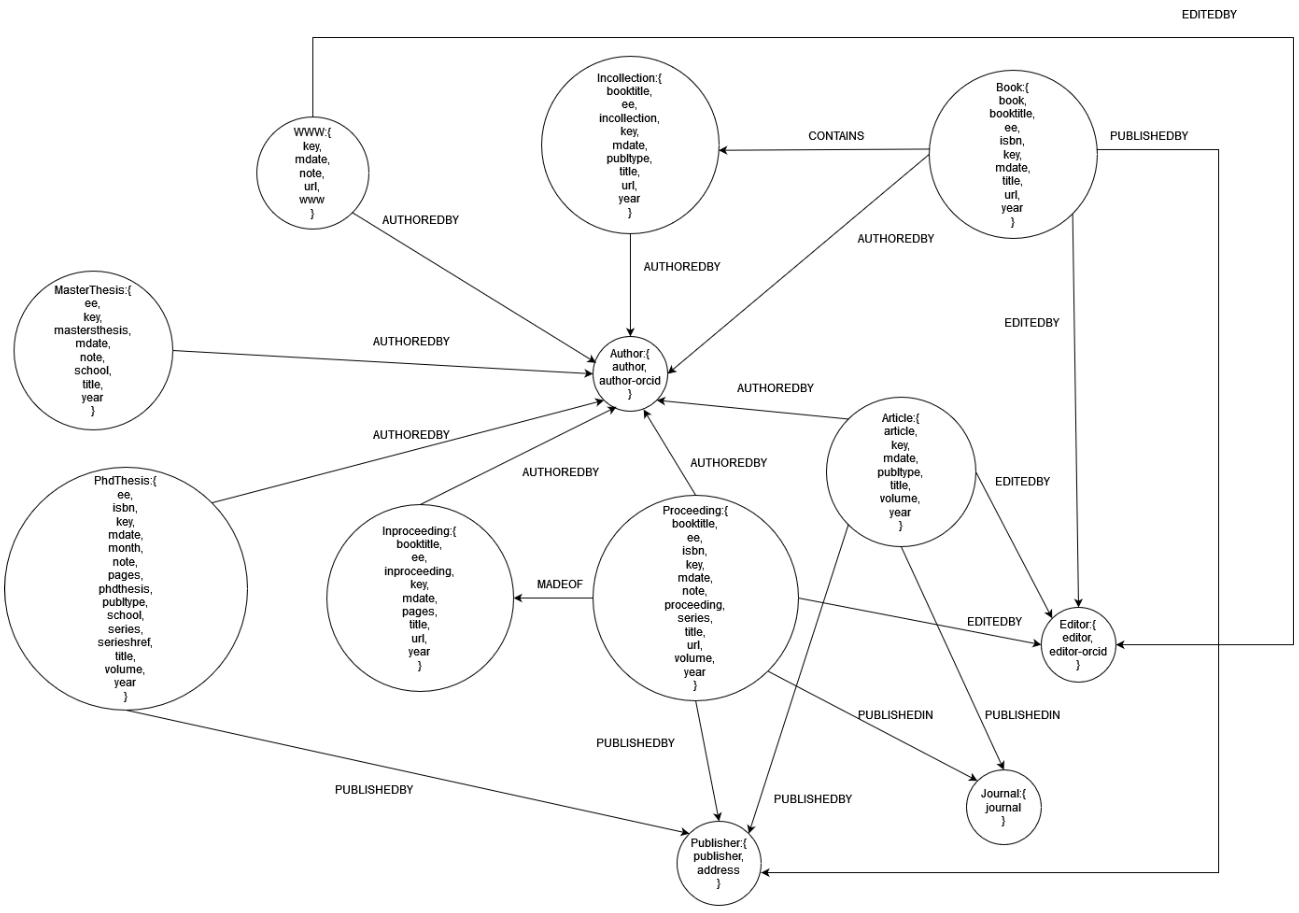
- Proceeding: label identifying all the proceedings published. It is connected to Author, Editor, Journal and Publisher. Attributes:
  - ee: electronic edition of the proceeding;
  - isbn: isbn of the proceeding;
  - key: path in the dblp file system to find the proceeding;
  - mdate: date of the last modification of the record;
  - note: additional notes;
  - proceeding: progressive number for all proceedings;
  - publtype: type of the publication informal, encyclopedia, ...;
  - school: an institution associated to the author of the publication;
  - series: name of the series the work belongs to;
  - title: title of the publication;
  - url: link to the page of the series, if there is one;
  - volume: if series appears along with the volume attribute, volume is interpreted as the numbering of the series. If volume appears on its own, it's the number of the volume the proceeding appears in;
  - year: year of publication.
- Publisher: label identifying all the publishers. It is connected to PhdThesis, Book, Incollection, Proceeding and Article. Attributes:
  - publisher: name of the publisher;
  - address: address of the publisher.
- WWW: label identifying all the pages referred to an author. It is connected to Author and Editor. Attributes:
  - key: path to find the record in the file system of the database;
  - mdate: date of the last modification of the record;
  - note: additional notes;
  - publtype: is used when the author is not a physical person, but an association of some sort;

  - title: describe the kind of webpage;
  - url: contains the list of link to the pages of the authors;
  - www: progressive number for all the www elements.

2. **Links**

   - AuthoredBy: connects a General Pubblication (that can be an Article, a Book, a Proceeding, a MasterThesis or a PhdThesis), an Incollection or an Inproceeding with the relative author(s). It can also connect a WWW (Web page) with the relative owner(s). It is in both cases a many-to-many relationship;
   - Contains: connects a Book with all the Incollections that are part of that book. It is a one-to-many relationship.
   - EditedBy: connects a Pubblication (that can be an Article, a Book or a Proceeding) with the relative editor(s). It is a many-to-many relationship;
   - MadeOf: connects an Inproceeding with all the contained Proceedings. It is a one-to-many relationship.
   - PublishedBy: connects a General Pubblication with the relative Publisher. All the Pubblications are always linked to an editor except theses: in particular, PhdThesis *may* be linked to a Publisher, while a MasterThesis never has a Publisher. It is a one-to-many relationship;
   - PublishedIn: connects a Journal with all the Articles that are published inside it. It is a one-to-many relationship.

3. Diagram


EDITEDBY
WWW:{ key, mdate, note, url, www }
Incollection:{ booktitle, ee, incollection, key, mdate, publtype, title, url, year }
Book:{ book, booktitle, ee, isbn, key, mdate, title, url, year }
CONTAINS
PUBLISHEDBY
AUTHOREDBY
AUTHOREDBY
AUTHOREDBY
MasterThesis:{ ee, key, mastersthesis, mdate, note, school, title, year }
AUTHOREDBY
Author:{ author, author-orcid }
EDITEDBY
AUTHOREDBY
Article:{ article, key, mdate, publtype, title, volume, year }
AUTHOREDBY
AUTHOREDBY
AUTHOREDBY
EDITEDBY
PhdThesis:{ ee, isbn, key, mdate, month, note, pages, phdthesis, publtype, school, series, serieshref, title, volume, year }
Inproceeding:{ booktitle, ee, inproceeding, key, mdate, pages, title, url, year }
MADEOF
Proceeding:{ booktitle, ee, isbn, key, mdate, note, proceeding, series, title, url, volume, year }
EDITEDBY
Editor:{ editor, editor-orcid }
PUBLISHEDIN
PUBLISHEDIN
PUBLISHEDBY
PUBLISHEDBY
PUBLISHEDBY
Journal:{ journal }
Publisher:{ publisher, address }

1.2.3. Commands

- **Insert a new book in the system:**

```
CREATE (b:Book {book: "43948239428", booktitle: "SMBUD Project",
    isbn: ["0"], key: "reference/smbud", mdate:"24/10/2022", title:"
    SMBUD Project", url:"db/reference/smbud", series: "SMBUD Books",
     `series-href`: "db/series/smbud", year:2022})
RETURN b
```

- **Create the relationship between the new book and its author:**

```
MATCH (b:Book {key: "reference/smbud"}), (a:Author {author: "
    Stefano Ceri"})
CREATE (b)-[r:AuthoredBy]->(a)
RETURN r
```

- **Modify the ISBN of the new book:**

```
MATCH (b:Book {key: "reference/smbud"})
SET b.isbn = "9780080503936"
RETURN r
```

- **Create new authors:**

```
CREATE (a1:Author {author: "Luca Rondini"}), (a2:Author {author: "
    Francesco Scandale"}), (a3:Author {author: "Marta Radaelli"}), (
    a4:Author {author: "Elena Montemurro"}), (a5:Author {author: "
    Roberto Macaccaro"})
RETURN a1, a2, a3, a4, a5
```

- **Connect the new authors to their book:**

```
MATCH (a1:Author {author: "Luca Rondini"}), (a2:Author {author: "
    Francesco Scandale"}), (a3:Author {author: "Marta Radaelli"}), (
    a4:Author {author: "Elena Montemurro"}), (a5:Author {author: "
    Roberto Macaccaro"}), (b:Book {key: "reference/smbud"})
CREATE (b)-[r1:AuthoredBy]->(a1), (b)-[r2:AuthoredBy]->(a2), (b)-[
    r3:AuthoredBy]->(a3), (b)-[r4:AuthoredBy]->(a4), (b)-[r5:
    AuthoredBy]->(a5)
RETURN type(r1)
```

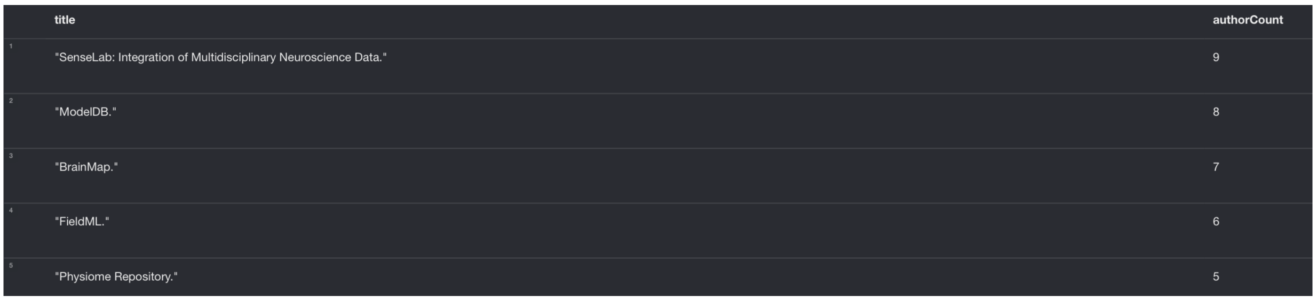
- **Delete the newly created book:**

```
MATCH (b:Book {key: "reference/smbud"})
DETACH DELETE b
```

1.2.4. Queries

- Count the number of authors that worked on incollections for the book "Encyclopedia of Computational Neuroscience":

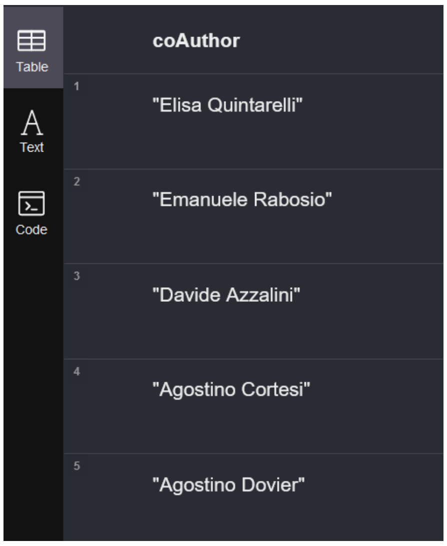
```
MATCH (b:Book{key: "reference/cn/2014" })-[:Contains]->(i:
    Incollection)-[:AuthoredBy]->(a:Author)
WITH i.title as title, count(*) AS authorCount
ORDER BY authorCount DESC
RETURN title,authorCount LIMIT 20
```



	title	authorCount
1	"SenseLab: Integration of Multidisciplinary Neuroscience Data."	9
2	"ModelDB."	8
3	"BrainMap."	7
4	"FieldML."	6
5	"Physiome Repository."	5

- List all distinct co-authors of Letizia Tanca:

```
MATCH (a:Author {author:"Letizia Tanca"})<-[:AuthoredBy]-()-[:
    AuthoredBy]->(coAuthor:Author)
RETURN DISTINCT coAuthor.author
```

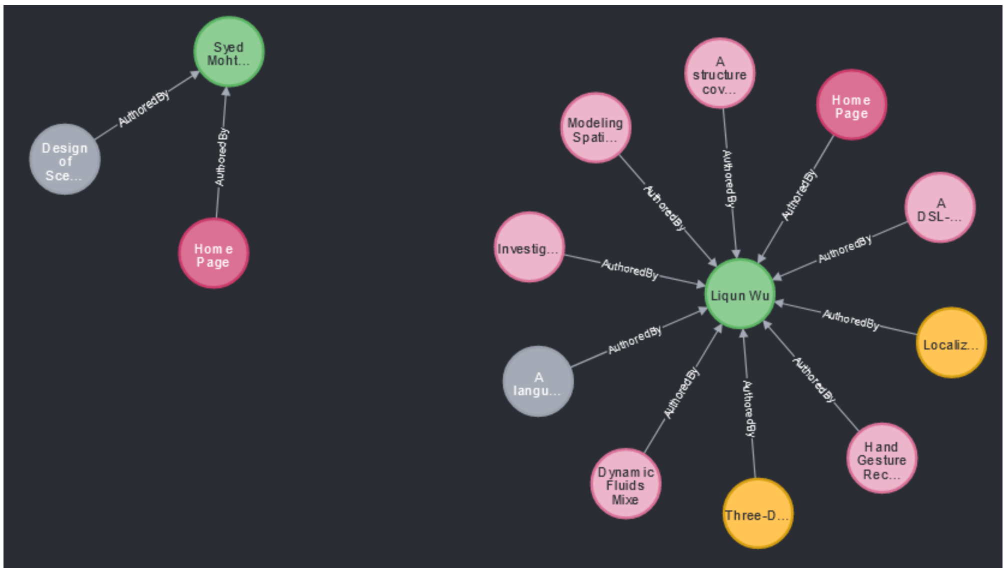


	coAuthor
1	"Elisa Quintarelli"
2	"Emanuele Rabosio"
3	"Davide Azzalini"
4	"Agostino Cortesi"
5	"Agostino Dovier"

- **Find all works by authors that also wrote a phdThesis while at University of Oldenburg, Germany**:

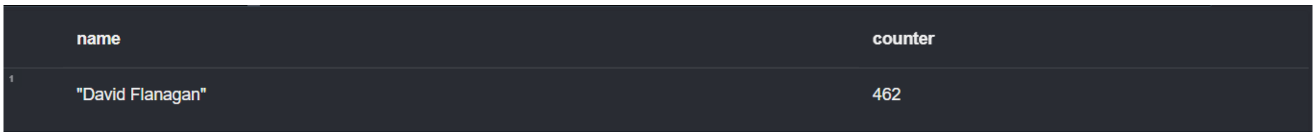
```
MATCH (p:PhdThesis)-[:AuthoredBy]->(a:Author)<-[:AuthoredBy]-(works)
WHERE "University of Oldenburg, Germany" IN p.school
RETURN p,a,works LIMIT 10
```

Authors in green, Articles in yellow, Inproceedings in light pink, PhdThesis in grey, WWW in dark pink.



- **Find an author with multiple works published for the same publisher, with at least one of them being a book**:

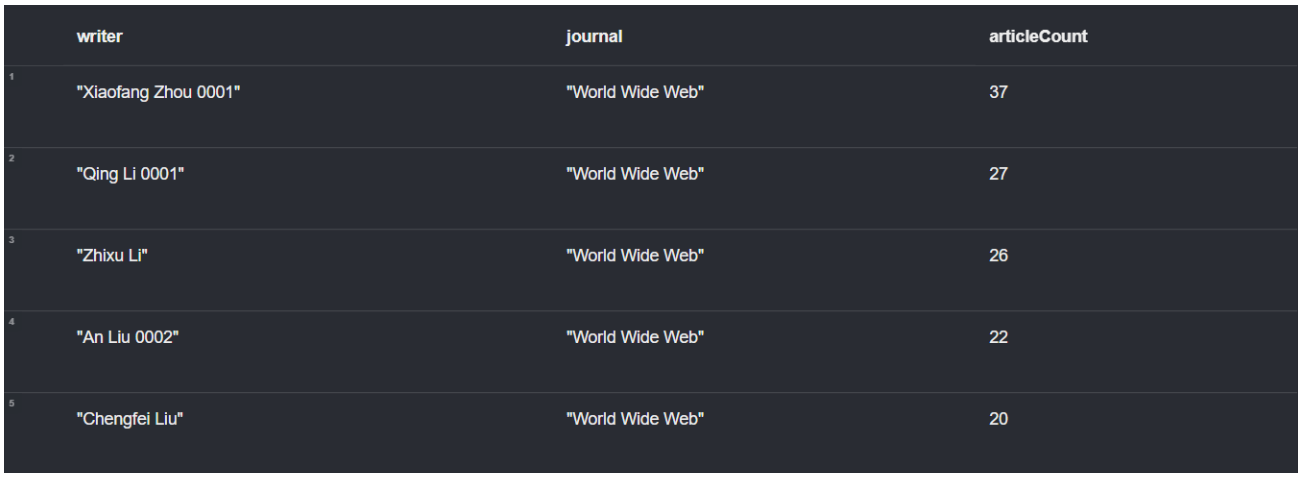
```
MATCH (p:Publisher)<-[:PublishedBy]-(b:Book)-[:AuthoredBy]->(a:Author)<-[:AuthoredBy]-(n)-[:PublishedBy]->(p)
WHERE n.title<>b.title
WITH COUNT(a) AS counter, a.author AS name
ORDER BY counter DESC
LIMIT 1
RETURN name, counter
```



	name	counter
1	"David Flanagan"	462

- **Find the top 10 writers for the journal "World Wide Web":**

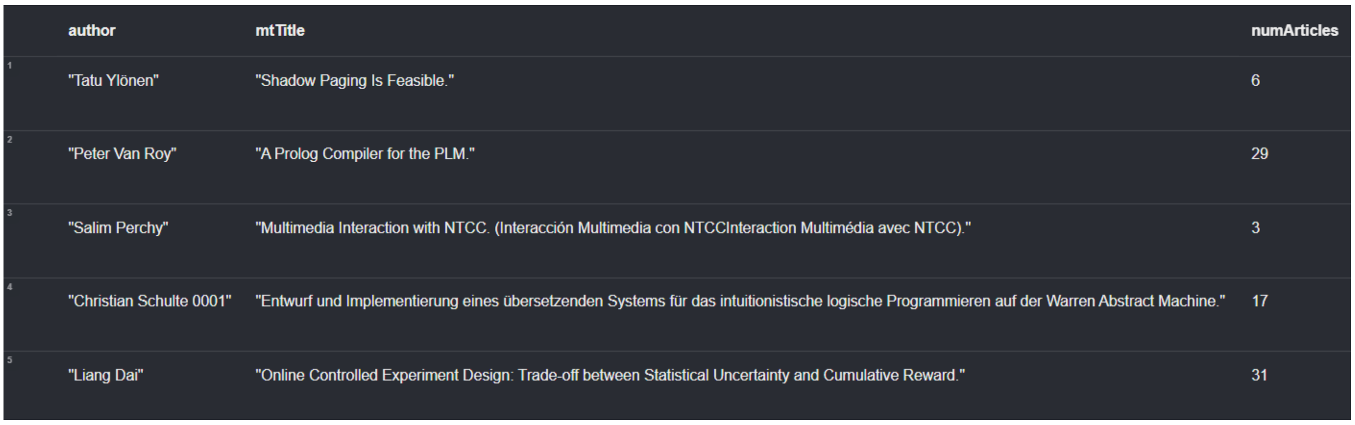
```
MATCH (a:Author)<-[:AuthoredBy]-(article:Article)-[:PublishedIn]->(
    j:Journal {journal:"World Wide Web"})
WITH a,j, count(article) AS articleCount
ORDER BY articleCount DESC
LIMIT 10
RETURN a.author as writer,j.journal as journal, articleCount
```



	writer	journal	articleCount
1	"Xiaofang Zhou 0001"	"World Wide Web"	37
2	"Qing Li 0001"	"World Wide Web"	27
3	"Zhixu Li"	"World Wide Web"	26
4	"An Liu 0002"	"World Wide Web"	22
5	"Chengfei Liu"	"World Wide Web"	20

- **Find Master theses authors who also wrote articles, including the count of articles:**

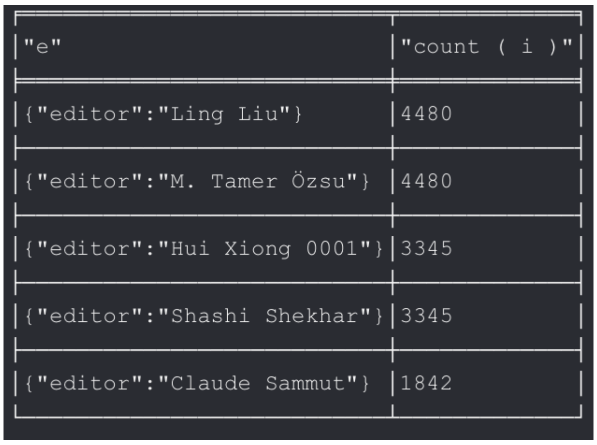
```
MATCH (m:MasterThesis)-[:AuthoredBy]->(a:Author)<-[:AuthoredBy]-(b:
    Article)
WITH a.author AS author, m.title AS mtTitle, COUNT(b) AS
    numArticles
RETURN author, mtTitle, numArticles
```



	author	mtTitle	numArticles
1	"Tatu Ylönen"	"Shadow Paging Is Feasible."	6
2	"Peter Van Roy"	"A Prolog Compiler for the PLM."	29
3	"Salim Perchy"	"Multimedia Interaction with NTCC. (Interacción Multimedia con NTCCInteraction Multimédia avec NTCC)."	3
4	"Christian Schulte 0001"	"Entwurf und Implementierung eines übersetzenden Systems für das intuitionistische logische Programmieren auf der Warren Abstract Machine."	17
5	"Liang Dai"	"Online Controlled Experiment Design: Trade-off between Statistical Uncertainty and Cumulative Reward."	31

- **Find the top 25 editors by number of incollections in books they edited:**

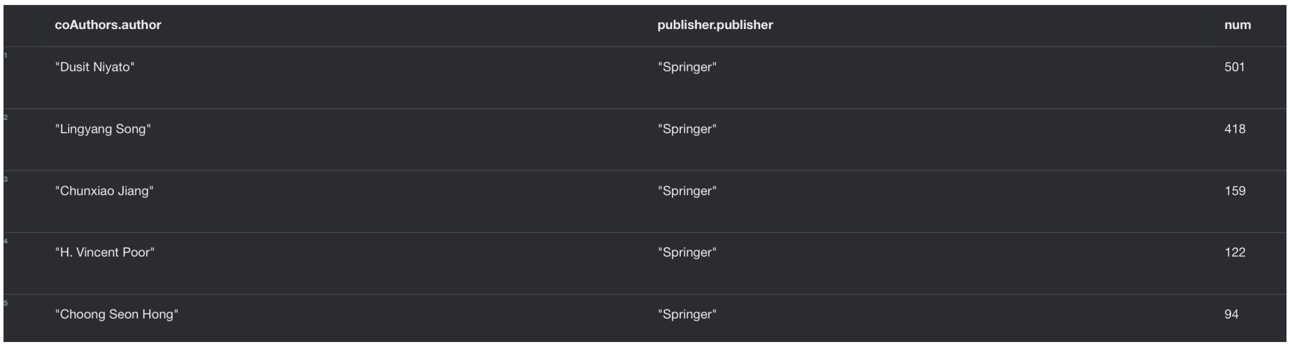
```
MATCH (i:Incollection)<-[r:Contains]-(b:Book)-[r2:EditedBy]->(e:Editor)
RETURN e, count(i) ORDER BY count(i) DESC LIMIT 25
```



```
╒═══════════════════════════════╤═══════════╕
│"e"                            │"count(i)" │
╞═══════════════════════════════╪═══════════╡
│{"editor":"Ling Liu"}          │4480       │
├───────────────────────────────┼───────────┤
│{"editor":"M. Tamer Özsu"}     │4480       │
├───────────────────────────────┼───────────┤
│{"editor":"Hui Xiong 0001"}    │3345       │
├───────────────────────────────┼───────────┤
│{"editor":"Shashi Shekhar"}    │3345       │
├───────────────────────────────┼───────────┤
│{"editor":"Claude Sammut"}     │1842       │
└───────────────────────────────┴───────────┘
```

- **Count the number of books published, by publisher, written by authors that collaborated with Zhu Han 0001:**

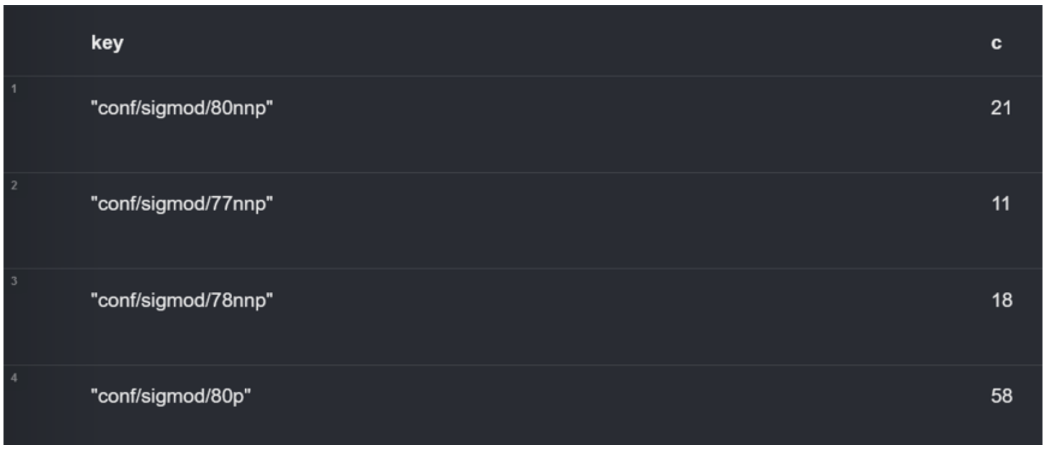
```
MATCH (a:Author {author:"Zhu Han 0001"})<-[:AuthoredBy]-()-[:AuthoredBy]->(coAuthors:Author), (coAuthors)<-[:AuthoredBy]-(b:Book)-[:PublishedBy]->(publisher)
WHERE a <> coAuthors
WITH coAuthors,publisher,count(*) AS num
ORDER BY num DESC
RETURN coAuthors.author,publisher.publisher,num LIMIT 5
```



coAuthors.author	publisher.publisher	num
"Dusit Niyato"	"Springer"	501
"Lingyang Song"	"Springer"	418
"Chunxiao Jiang"	"Springer"	159
"H. Vincent Poor"	"Springer"	122
"Choong Seon Hong"	"Springer"	94

- **Find number of inproceedings releated to proceedings which have been published on a journal**:

```
MATCH (i:Inproceeding)<-[:MadeOf]-(p:Proceeding)-[:PublishedIn]->(j
    :Journal)
WITH p, count(i) as c
RETURN p.key as key,c
```

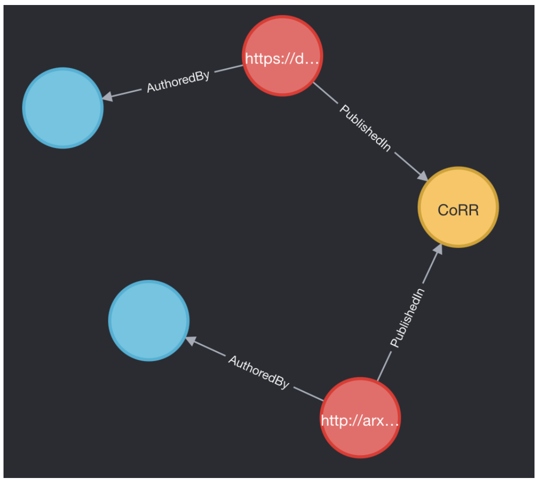


	key	c
1	"conf/sigmod/80nnp"	21
2	"conf/sigmod/77nnp"	11
3	"conf/sigmod/78nnp"	18
4	"conf/sigmod/80p"	58

- **Shortest path between two authors - Fast Bidirectional Breadth-first Search Algorithm**:

```
MATCH (a1:Author {author: "Angelo Morzenti"}), (a2:Author {author:
    "Marco Brambilla 0001"}), p=sh ortestPath((a1)-[*]-(a2))
WITH length(p) AS len, p
RETURN len,p
```

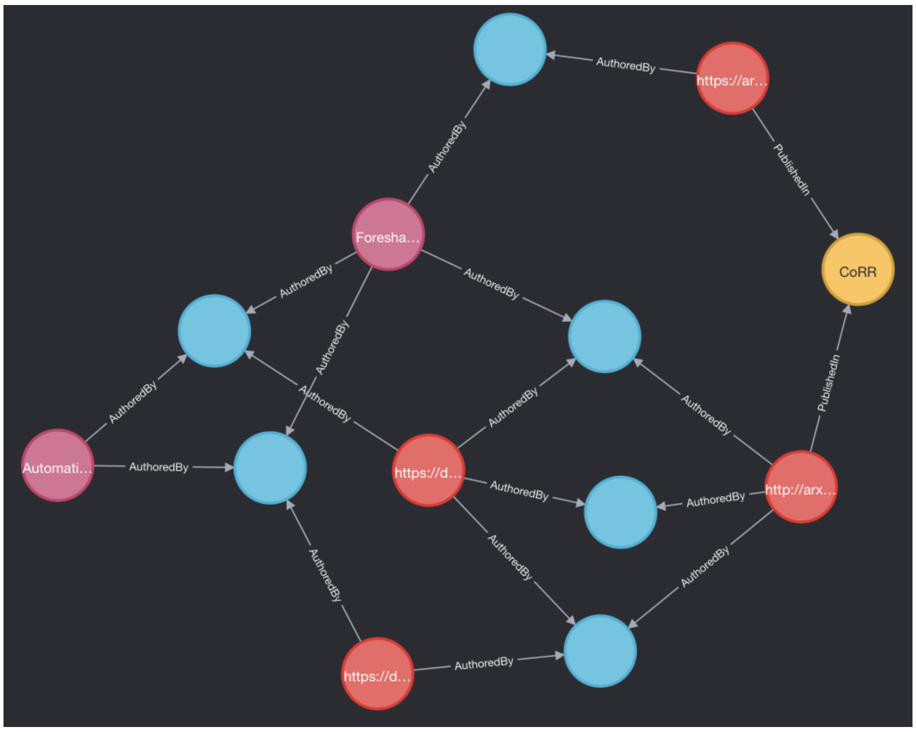
Author (in blue) on top is "Marco Brambilla 0001" and author at the bottom is "Angelo Morzenti". Authors in blue, Articles in red, Journals in yellow.



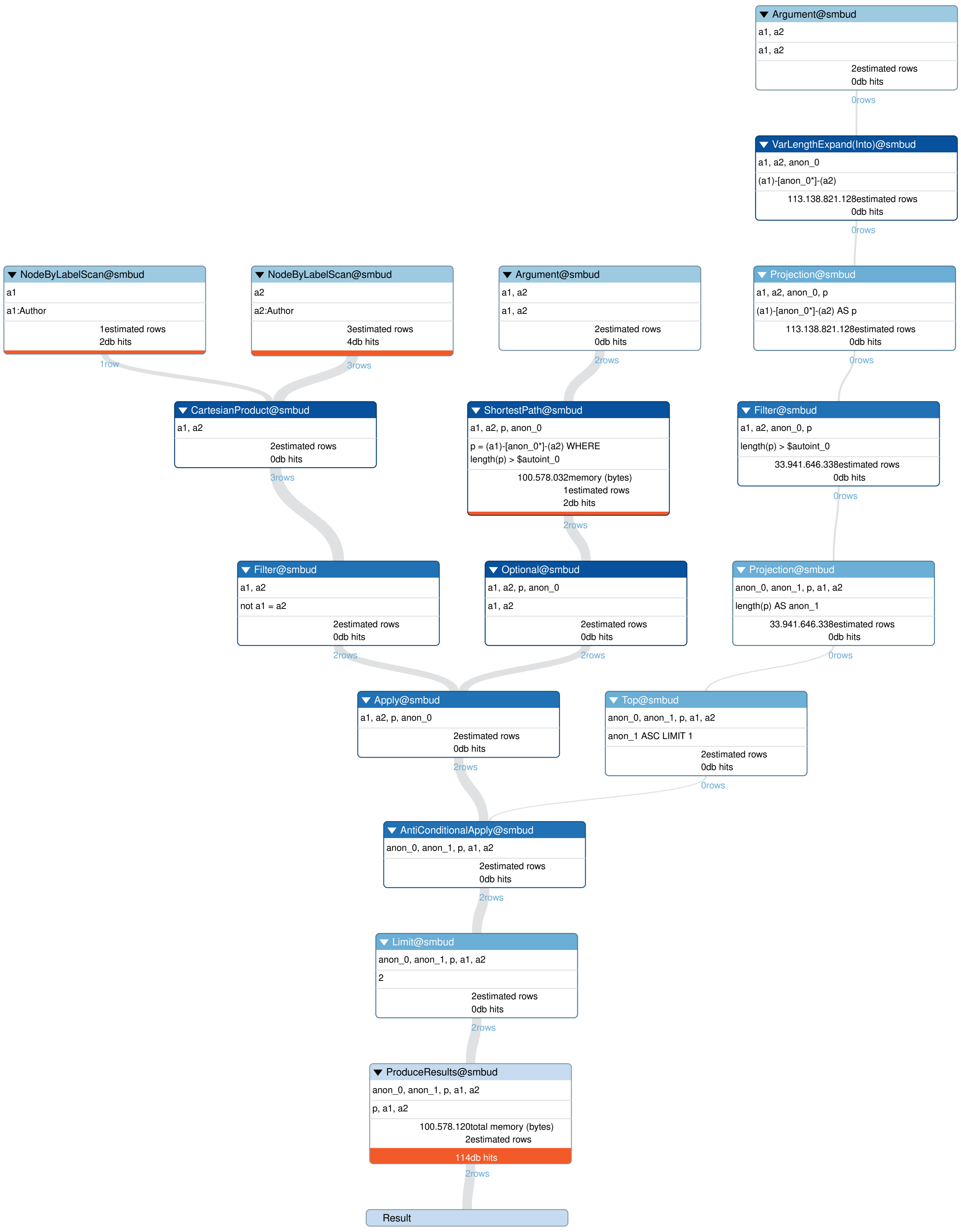
- **Shortest path between two authors - Slower Exhaustive Depth-first Search Algorithm**:

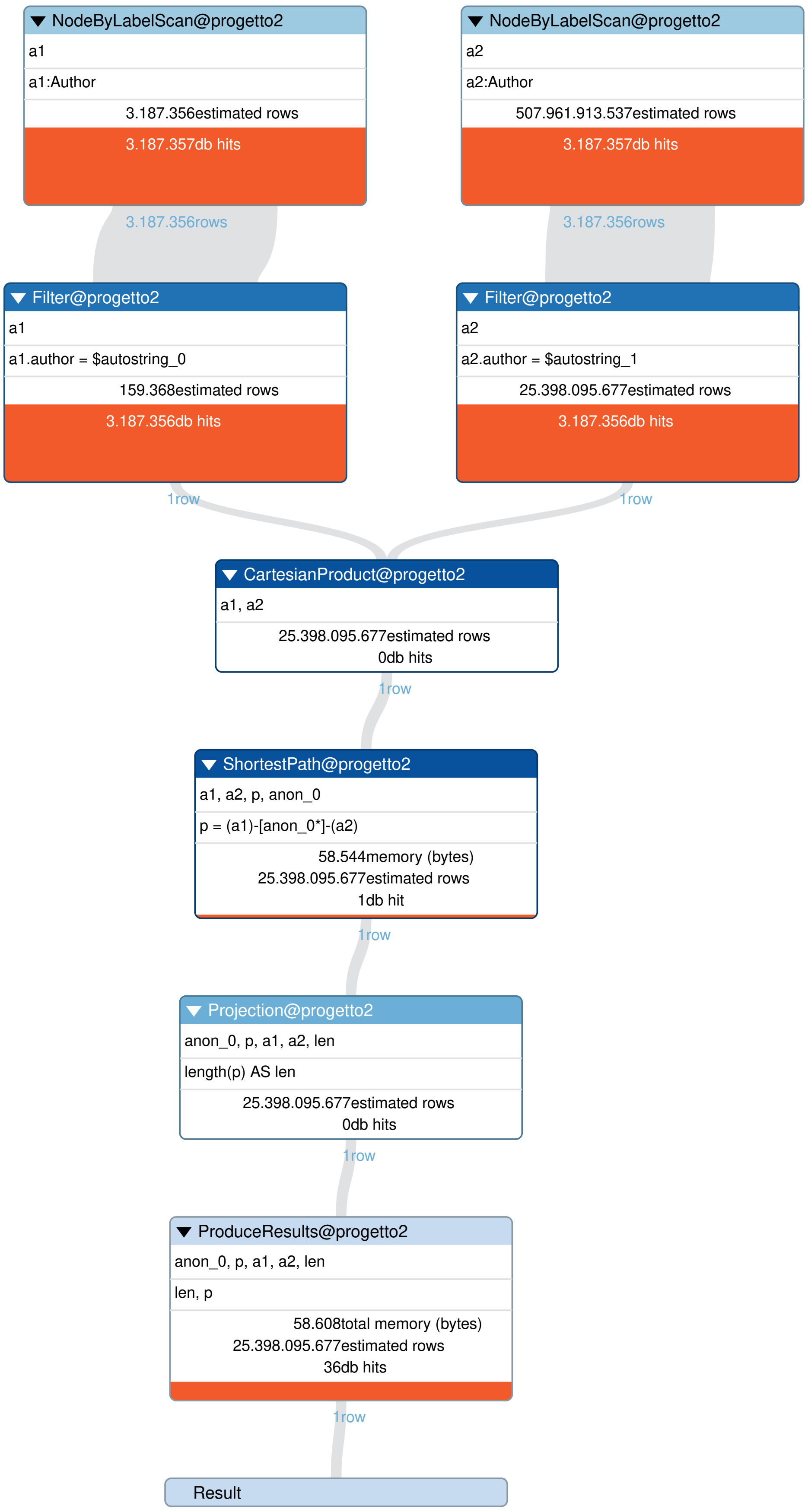
```
MATCH (a1:Author), (a2:Author), p=shortestPath((a1)-[*]-(a2))
WHERE length(p) > 5 AND a1<>a2
RETURN p,a1,a2 LIMIT 2
```

Result with the paths between "Paul Kocher" and "Daniel Genkin", "Paul Kocher" and "Daniel Gruss".https://www.overleaf.com/project/635ba58bd0c32f35ee476454 Authors in blue, Articles in red, Journals in yellow, Inproceedings in pink.



Schemas showing complexity and performance of these last two queries are on the following pages.


Argument@smbud
a1, a2
a1, a2
2estimated rows
0db hits
0rows
VarLengthExpand(Into)@smbud
a1, a2, anon_0
(a1)-[anon_0*]-(a2)
113.138.821.128estimated rows
0db hits
0rows
NodeByLabelScan@smbud
a1
a1:Author
1estimated rows
2db hits
1row
NodeByLabelScan@smbud
a2
a2:Author
3estimated rows
4db hits
3rows
Argument@smbud
a1, a2
a1, a2
2estimated rows
0db hits
2rows
Projection@smbud
a1, a2, anon_0, p
(a1)-[anon_0*]-(a2) AS p
113.138.821.128estimated rows
0db hits
0rows
CartesianProduct@smbud
a1, a2
2estimated rows
0db hits
3rows
ShortestPath@smbud
a1, a2, p, anon_0
p = (a1)-[anon_0*]-(a2) WHERE length(p) > $autoint_0
100.578.032memory (bytes)
1estimated rows
2db hits
2rows
Filter@smbud
a1, a2, anon_0, p
length(p) > $autoint_0
33.941.646.338estimated rows
0db hits
0rows
Filter@smbud
a1, a2
not a1 = a2
2estimated rows
0db hits
2rows
Optional@smbud
a1, a2, p, anon_0
a1, a2
2estimated rows
0db hits
2rows
Projection@smbud
anon_0, anon_1, p, a1, a2
length(p) AS anon_1
33.941.646.338estimated rows
0db hits
0rows
Apply@smbud
a1, a2, p, anon_0
2estimated rows
0db hits
2rows
Top@smbud
anon_0, anon_1, p, a1, a2
anon_1 ASC LIMIT 1
2estimated rows
0db hits
0rows
AntiConditionalApply@smbud
anon_0, anon_1, p, a1, a2
2estimated rows
0db hits
2rows
Limit@smbud
anon_0, anon_1, p, a1, a2
2
2estimated rows
0db hits
2rows
ProduceResults@smbud
anon_0, anon_1, p, a1, a2
p, a1, a2
100.578.120total memory (bytes)
2estimated rows
114db hits
2rows
Result


NodeByLabelScan@progetto2
a1
a1:Author
3.187.356estimated rows
3.187.357db hits
3.187.356rows
NodeByLabelScan@progetto2
a2
a2:Author
507.961.913.537estimated rows
3.187.357db hits
3.187.356rows
Filter@progetto2
a1
a1.author = $autostring_0
159.368estimated rows
3.187.356db hits
1row
Filter@progetto2
a2
a2.author = $autostring_1
25.398.095.677estimated rows
3.187.356db hits
1row
CartesianProduct@progetto2
a1, a2
25.398.095.677estimated rows
0db hits
1row
ShortestPath@progetto2
a1, a2, p, anon_0
p = (a1)-[anon_0*]-(a2)
58.544memory (bytes)
25.398.095.677estimated rows
1db hit
1row
Projection@progetto2
anon_0, p, a1, a2, len
length(p) AS len
25.398.095.677estimated rows
0db hits
1row
ProduceResults@progetto2
anon_0, p, a1, a2, len
len, p
58.608total memory (bytes)
25.398.095.677estimated rows
36db hits
1row
Result

2 | Second assignment

2.1. Assignment description

The purpose of this assignment is to build an information system that is able to manage a big set of scientific papers with all their information and references. The system must be scalable and flexible enough to handle a big amount of data and `MongoDB` is the perfect solution as it's a schema-less, document-oriented database. The schema-less approach allows to store different information for each paper, with the biggest, but acceptable, drawback being the duplication of some of the data, such as the titles of journals.

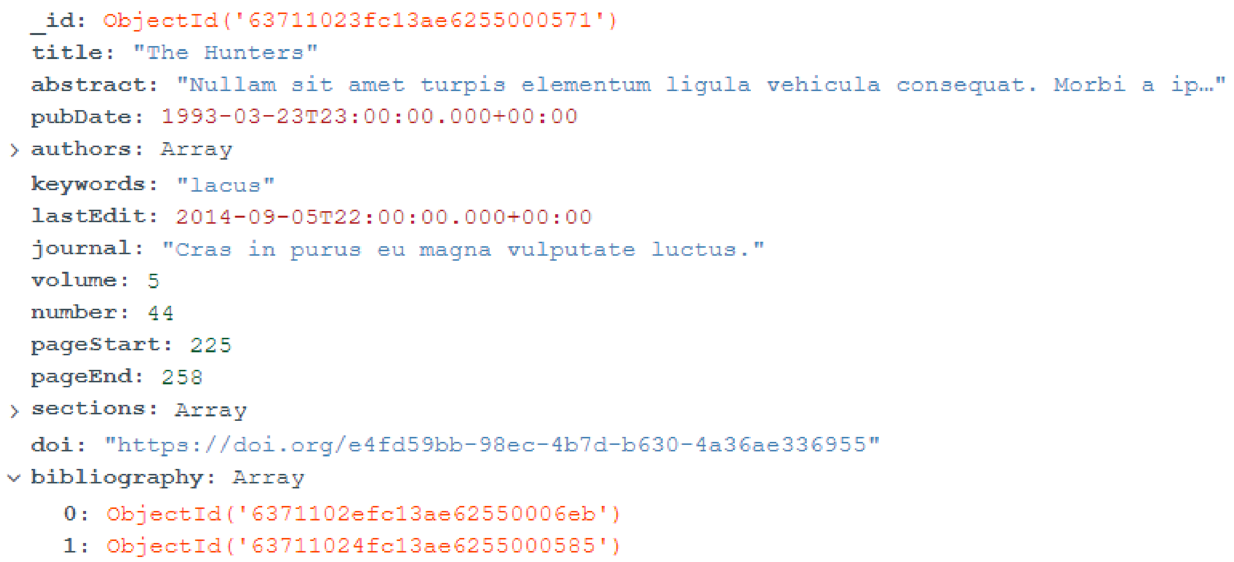
2.2. Data structure

The structure of the documents, representing scientific papers, is composed by the following attributes and sub-documents:

- *ID*: the unique identifier of the document. It's automatically generated/imported by MongoDB when the document is created/imported;
- *Title*: the title of the paper;
- *Abstract*: a short summary of the content of the paper;
- *Publication Date*: the date in which the paper was published;
- *Authors*: an array containing information about the authors of the paper. Each element of the array represents a different author and is a sub-document composed of:
  - *Name*: the first name of the author;
  - *Surname*: the last name of the author;
  - *ORCID*: the ORCID of the author;
  - *Birthdate*: the birth date of the author;

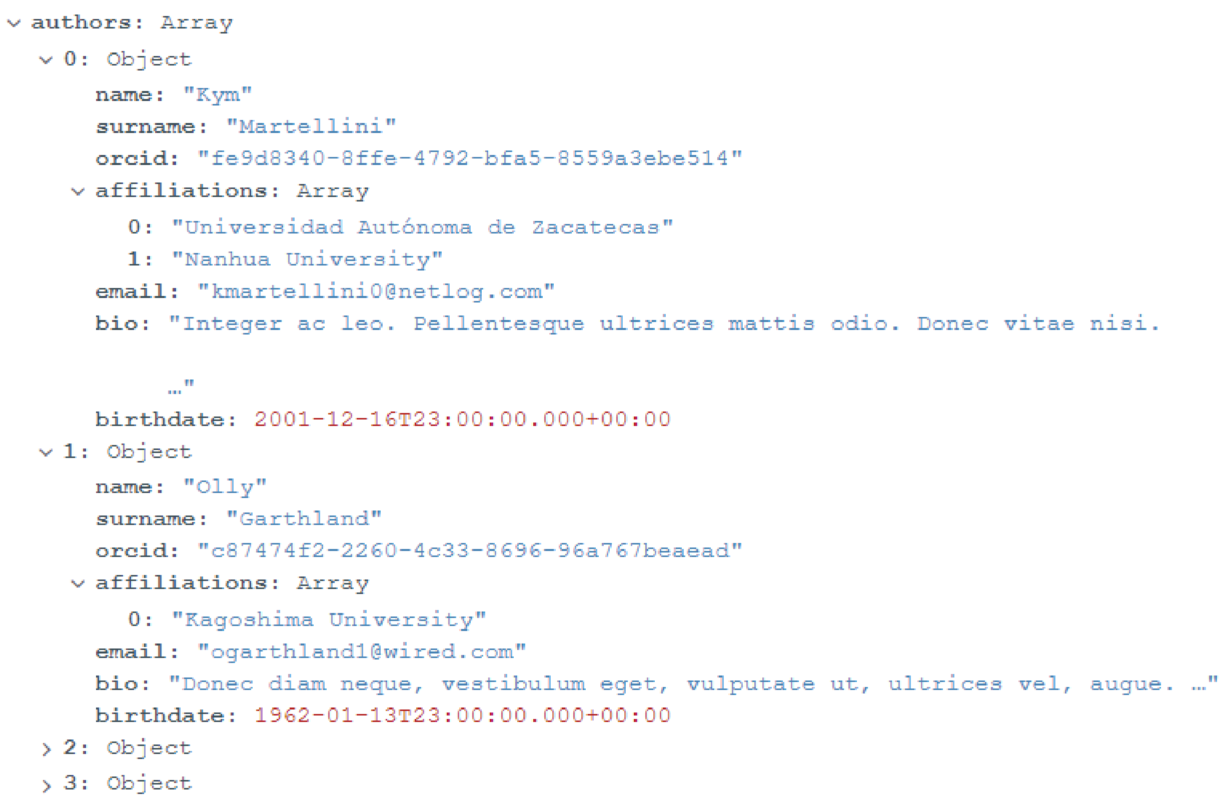
    - *Email*: the email address of the author;
    - *Bio*: a short biography of the author;
    - *Affiliations*: an array of strings, where each string is the name of a university or other institutions the author is affiliated with;
- *Keywords*: a string containing keywords related to the paper;
- *Last Edit*: the date of the last edit of the document made on the database;
- *Journal*: the name of the journal in which the paper was published;
- *Volume*: the number of the volume of the journal that contains the paper;
- *Number*: the number of the journal that contains the paper;
- *Start Page*: the number of the page in which the paper begins;
- *End Page*: the number of the page in which the paper ends;
- *Sections*: an array of sub-documents, where each sub-document represents a section of the paper. Each sub-document has a "type" attribute that indicates the type of section that it represents. The others attributes depend on the subsection type as follows:
    - *Title*: a section that represents the title of a paragraph. Attributes:
        - *Text*: the title itself;
    - *Text*: a section that represents a paragraph. Attributes:
        - *Text*: the text of the paragraph;
    - *Figure*: a section that represents a figure inside the paper. Attributes:
        - *URL*: the URL that points to the image file;
        - *Caption*: the text of the caption associated with the figure;
    - *Subsection*: a section that contains other sections. Attributes:
        - *Subsection*: an array of sections. In a recursive manner, each section has the same structure that is being described here;
- *DOI*: the Digital Object Identifier of the paper;
- *Bibliography*: an array of references to all the others works mentioned in the paper. Each reference is the ID of the document, not the document itself.

Here is a preview of a sample document, taken from the database created:



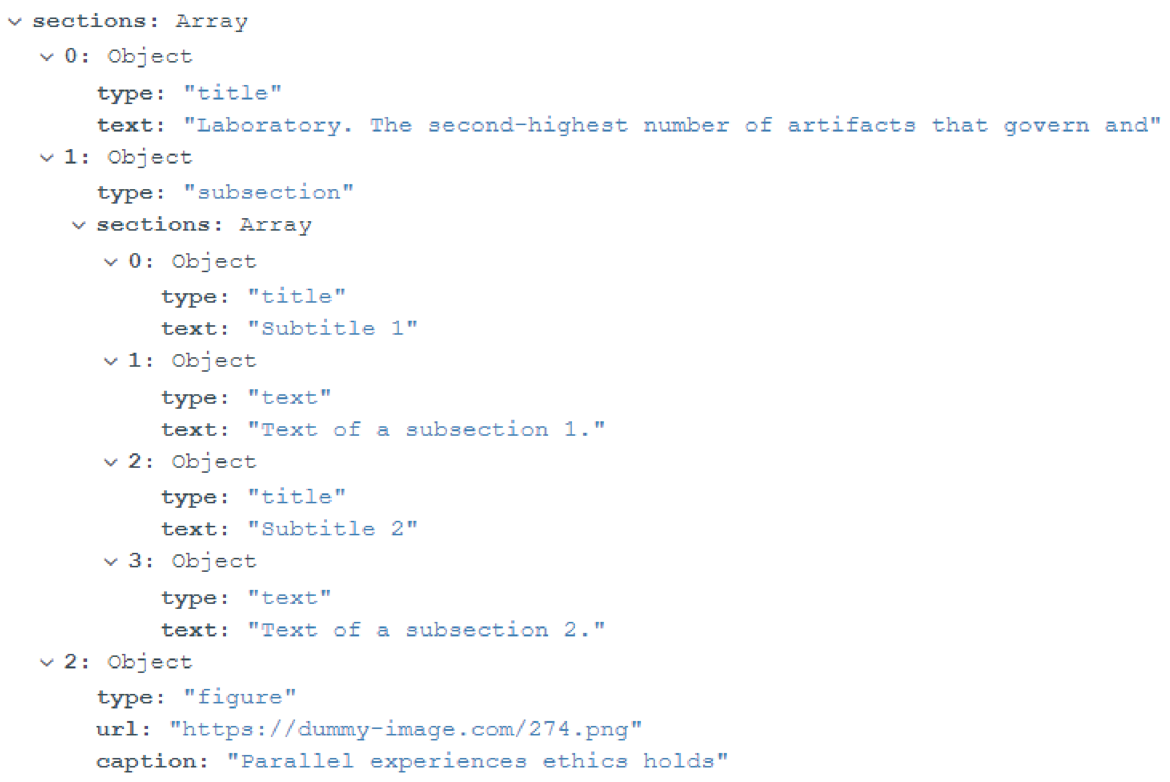
```
  _id: ObjectId('63711023fc13ae6255000571')
  title: "The Hunters"
  abstract: "Nullam sit amet turpis elementum ligula vehicula consequat. Morbi a ip…"
  pubDate: 1993-03-23T23:00:00.000+00:00
› authors: Array
  keywords: "lacus"
  lastEdit: 2014-09-05T22:00:00.000+00:00
  journal: "Cras in purus eu magna vulputate luctus."
  volume: 5
  number: 44
  pageStart: 225
  pageEnd: 258
› sections: Array
  doi: "https://doi.org/e4fd59bb-98ec-4b7d-b630-4a36ae336955"
⌄ bibliography: Array
    0: ObjectId('6371102efc13ae62550006eb')
    1: ObjectId('63711024fc13ae6255000585')
```

By expanding the *author* and *sections* arrays also the sub-documents are visible:



```
⌄ authors: Array
  ⌄ 0: Object
      name: "Kym"
      surname: "Martellini"
      orcid: "fe9d8340-8ffe-4792-bfa5-8559a3ebe514"
    ⌄ affiliations: Array
        0: "Universidad Autónoma de Zacatecas"
        1: "Nanhua University"
      email: "kmartellini0@netlog.com"
      bio: "Integer ac leo. Pellentesque ultrices mattis odio. Donec vitae nisi.

           …"
      birthdate: 2001-12-16T23:00:00.000+00:00
  ⌄ 1: Object
      name: "Olly"
      surname: "Garthland"
      orcid: "c87474f2-2260-4c33-8696-96a767beaead"
    ⌄ affiliations: Array
        0: "Kagoshima University"
      email: "ogarthland1@wired.com"
      bio: "Donec diam neque, vestibulum eget, vulputate ut, ultrices vel, augue. …"
      birthdate: 1962-01-13T23:00:00.000+00:00
  › 2: Object
  › 3: Object
```



```
⌄ sections: Array
  ⌄ 0: Object
      type: "title"
      text: "Laboratory. The second-highest number of artifacts that govern and"
  ⌄ 1: Object
      type: "subsection"
    ⌄ sections: Array
      ⌄ 0: Object
          type: "title"
          text: "Subtitle 1"
      ⌄ 1: Object
          type: "text"
          text: "Text of a subsection 1."
      ⌄ 2: Object
          type: "title"
          text: "Subtitle 2"
      ⌄ 3: Object
          type: "text"
          text: "Text of a subsection 2."
  ⌄ 2: Object
      type: "figure"
      url: "https://dummy-image.com/274.png"
      caption: "Parallel experiences ethics holds"
```

2.3. Data generation

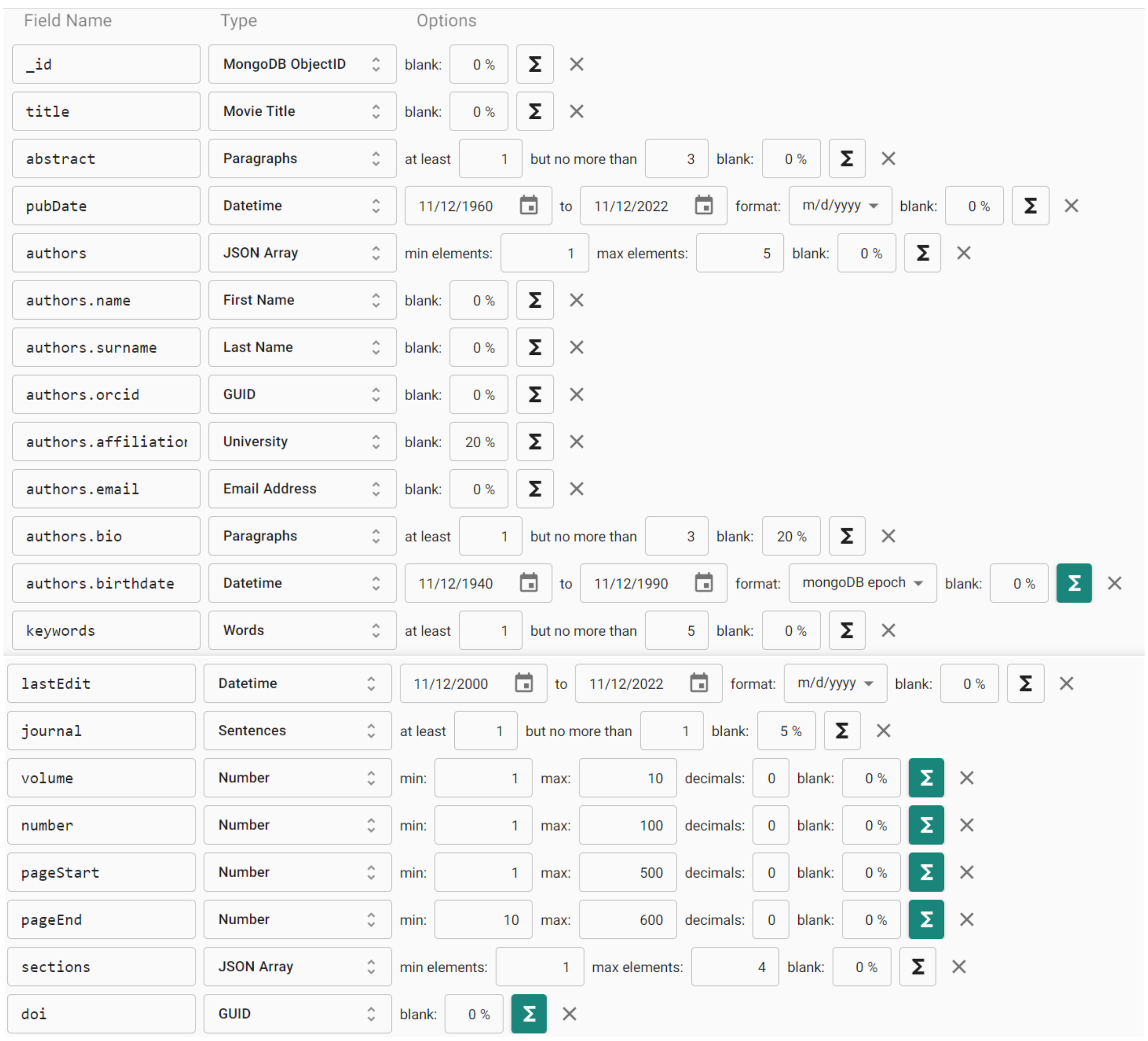
The documents used to populate the database were randomly generated using a two stage approach that allowed to create thousands of different documents that reflected the structure defined above.
In particular, the Mockaroo online data generator tool was used to generate the JSON dataset with the documents and all their attributes and sub-documents excluding the *sections* array and the *bibliography* array. These two remaining parts were added in the second stage using a custom script (see section 2.3.2).

2.3.1. First stage - Mockaroo data generation

The Mockaroo generator was set to generate the various attributes by using the most appropriate field type according to the content that each field is supposed to have in a real database. In particular, the following assumptions have been applied in order to better simulate a realistic dataset:

- Some attributes (like the author bio or the list of affiliations) were not generated for all the authors (this is achieved by setting a *blank* percentage different from 0 in Mockaroo) to simulate the real-life situation in which some information is missing;

- A *formula* was applied to the generation of the *Volume*, *Number*, *Page Start* and *Page End* attributes to ensure that, if the *Journal* attribute is not present (see the previous point), then also those values are not generated;
- A *formula* was applied to the generation of the *Page End* attribute to ensure that its value is always bigger than or equal to the *Page Start* number (if this is not true following the random generation, the function substitutes the value with an appropriate one);
- A *formula* was applied to the author birth date to ensure that if, according to the generated publication date, an author is younger than 18 years old at the time of publication of the paper, then the birth date is modified so that the author is at least 18 years old.



Field Name	Type	Options
_id	MongoDB ObjectID	blank: 0 %
title	Movie Title	blank: 0 %
abstract	Paragraphs	at least 1 but no more than 3 blank: 0 %
pubDate	Datetime	11/12/1960 to 11/12/2022 format: m/d/yyyy blank: 0 %
authors	JSON Array	min elements: 1 max elements: 5 blank: 0 %
authors.name	First Name	blank: 0 %
authors.surname	Last Name	blank: 0 %
authors.orcid	GUID	blank: 0 %
authors.affiliatior	University	blank: 20 %
authors.email	Email Address	blank: 0 %
authors.bio	Paragraphs	at least 1 but no more than 3 blank: 20 %
authors.birthdate	Datetime	11/12/1940 to 11/12/1990 format: mongoDB epoch blank: 0 %
keywords	Words	at least 1 but no more than 5 blank: 0 %
lastEdit	Datetime	11/12/2000 to 11/12/2022 format: m/d/yyyy blank: 0 %
journal	Sentences	at least 1 but no more than 1 blank: 5 %
volume	Number	min: 1 max: 10 decimals: 0 blank: 0 %
number	Number	min: 1 max: 100 decimals: 0 blank: 0 %
pageStart	Number	min: 1 max: 500 decimals: 0 blank: 0 %
pageEnd	Number	min: 10 max: 600 decimals: 0 blank: 0 %
sections	JSON Array	min elements: 1 max elements: 4 blank: 0 %
doi	GUID	blank: 0 %

2.3.2. Second stage - Data preprocessing

Since all the necessary data couldn't be generated using Mockaroo, two ad-hoc scripts helped completing it.
The first script was used to generate the attribute "sections" of the documents: to make this process easier, the possible combinations of sections and subsections were all statically set, assigning to each document at least title and text (there could also be images, URLs, different number of subsections, ...). Then, using a random sentence generator, all the textual fields were filled, while the URLs were created with a randomly generated number concatenated to the link "https://dummy-image.com/". Moreover, the bibliography was assigned by randomly extracting the number of citations and extracting the cited objects from the IDs of the articles already in the database.
The second script was then used to convert all date attributes from strings to actual JSON dates because they were originally recognized by MongoDB as strings and no quicker solution was found in Mockaroo.

2.3.3. Importing the data in MongoDB

Through the MongoDB application, the database "smbud" and its collection "articles" were created. The dataset can be easily imported by clicking on the corresponding button, setting the type of the input file as "JSON" and selecting the corresponding file.

2.4. Commands

- **Insert a document with the date in the wrong format (on purpose to delete it later)**:

```
db.articles.insertOne({
  "title": "SMBUD handbook",
  "abstract": "let's write a great abstract for this article",
  "pubDate": Date("12/5/2020"),
  "authors": [{"name": "Marta", "surname": "Radaelli", "orcid": "
   73532eb7-1234-76f9-aa07-f3af45h9ff87", "affiliations": ["
   Politecnico di Milano", "Universita' Statale di Milano"],
  "email": "marta.radaelli99@polimi.it", "bio": "lived in Erba all
   here life, looking at the lake", "birthdate": Date("10/6/1999")
   }, {"name": "Luca", "surname": "Rondini", "orcid": "71kj2eb7-
   cvbf-76f9-aa07-f3af45123457", "affiliations": ["Politecnico di
   Milano"],
  "email": "luca.rond@libero.it", "bio": "travels everyday from
   Bergamo, he's very tired", "birthdate": Date("11/8/1999")}],
  "keywords": "great random text",
  "lastEdit": Date("12/5/2021"),
  "journal": "Scientific Journal",
  "volume": 2,
  "number": 13,
  "pageStart": 56,
  "pageEnd": 59,
  "sections": [{"type": "title", "text": "Section 1 of the great
   article"}, {"type": "text", "text": "we are writing the best
   article that for some reason takes 3 pages."}],
  "doi": "https://doi.org/9a13rf00-7632-4551-afe6-4d2asdfg82cd",
})
```

- **Delete the document just created**:

```
db.articles.deleteOne({"title": "SMBUD handbook"})
```

- **Insert a correct document**:

```
db.articles.insertOne({
  "title": "SMBUD handbook",
  "abstract": "let's write a great abstract for this article",
  "pubDate":  ISODate('2020-05-12T23:00:00Z'),
  "authors": [{"name": "Marta", "surname": "Radaelli", "orcid": "
   73532eb7-1234-76f9-aa07-f3af45h9ff87", "affiliations": ["
   Politecnico di Milano", "Universita' Statale di Milano"],
  "email": "marta.radaelli99@polimi.it", "bio": "lived in Erba all
   here life, looking at the lake", "birthdate": ISODate('
   1999-06-10T23:00:00Z')}, {"name": "Luca", "surname": "Rondini",
   "orcid": "71kj2eb7-cvbf-76f9-aa07-f3af45123457", "affiliations":
    ["Politecnico di Milano"],
  "email": "luca.rond@libero.it", "bio": "travels everyday from
   Bergamo, he's very tired", "birthdate": ISODate('1999-08-11T23
   :00:00Z')}],
  "keywords": "great random text",
  "lastEdit": ISODate('2021-05-12T23:00:00Z'),
  "journal": "Scientific Journal",
  "volume": 2,
  "number": 13,
  "pageStart": 56,
  "pageEnd": 59,
  "sections": [{"type": "title", "text": "Section 1 of the great
   article"}, {"type": "text", "text": "we are writing the best
   article that for some reason takes 3 pages."}],
  "doi": "https://doi.org/9a13rf00-7632-4551-afe6-4d2asdfg82cd",
})
```

- **Add an author to the previous document**:

```
db.articles.updateOne(
  {"_id": ObjectId("63736862775c2d4bf1f7f5ca")},
  {"$push": {"authors": {"name": "Francesco", "surname": "Scandale"
    , "orcid": "73532eb7-alsk-76f9-a987-f3af1w23ff87", "affiliations
    ": ["Universita' Cattolica"],
  "email": "francesco.scandale@fastweb.it", "bio": "the only
    bergamasco eating spicy things", "birthdate": ISODate('
    2000-03-08T23:00:00Z')}}}
)
```

  The id is the one of the document inserted by the previous query

- **Insert more than one document**:

```
db.articles.insertMany([{
  "title": "SMBUD new insert one",
  "abstract": "new abstract written by us",
  "pubDate":  ISODate('2020-08-12T23:00:00Z'),
  "authors": [{"name": "Roberto", "surname": "Macaccaro", "orcid":
    "73532eb7-vfcd-76f9-jgnf-f3af45hoptrh", "affiliations": ["
    University of Insubria"],
  "email": "robert.macaccaro@vodafone.it", "bio": "lives in the
    best and most famous town of the world: Carbonate", "birthdate":
     ISODate('1998-07-14T23:00:00Z')}],
  "keywords": "new abstract document",
  "lastEdit": ISODate('2021-08-28T23:00:00Z'),
  "journal": "Insubria Journal",
  "volume": 4,
  "number": 2,
  "pageStart": 53,
  "pageEnd": 60,
  "sections": [{"type": "title", "text": "Subsections are fun"}, {"
    type": "text", "text": "writing queries takes a long time"}, {"
    type": "figure", "url": "https://dummy-image.com/345.png", "
    caption": "Roberto's selfie"}],
  "doi": "https://doi.org/9a13rf00-cbfv-4551-asdf-4d2345fg82cd",
},{
  "title": "SMBUD last document",
  "abstract": "this is an abstract.",
  "pubDate":  ISODate('2019-04-12T23:00:00Z'),
  "authors": [{"name": "Elena", "surname": "Montemurro", "orcid": "
    73532eb7-vfcd-76f9-ansh-f3af45hoptrh", "affiliations": ["
    Politecnico di Torino"],
  "email": "elena.montemurro@live.it", "bio": "trying to get a
```

```
    degree at the wrong university", "birthdate": ISODate('
    2000-02-16T23:00:00Z')}],
  "keywords": "try keywords",
  "lastEdit": ISODate('2021-08-28T23:00:00Z'),
  "journal": "Fantastic Journal",
  "volume": 2,
  "number": 13,
  "pageStart": 34,
  "pageEnd": 37,
  "sections": [{"type": "title", "text": "Subsections are fun"}, {"
    type": "text", "text": "writing queries takes a long time"}, {"
    type": "figure", "url": "https://dummy-image.com/345.png", "
    caption": "Elena's selfie"}],
  "doi": "https://doi.org/9a13rf00-1234-5678-asdf-4d2345fg82cd",
}])
```

- **Update the first document inserted with the last two as bibliography**:

```
db.articles.updateOne(
  {"title": "SMBUD handbook"},
  {"$set": {"bibliography": [ObjectId("63736eb3775c2d4bf1f7f5cc"),
    ObjectId("63736eb3775c2d4bf1f7f5cd")]}}
)
```

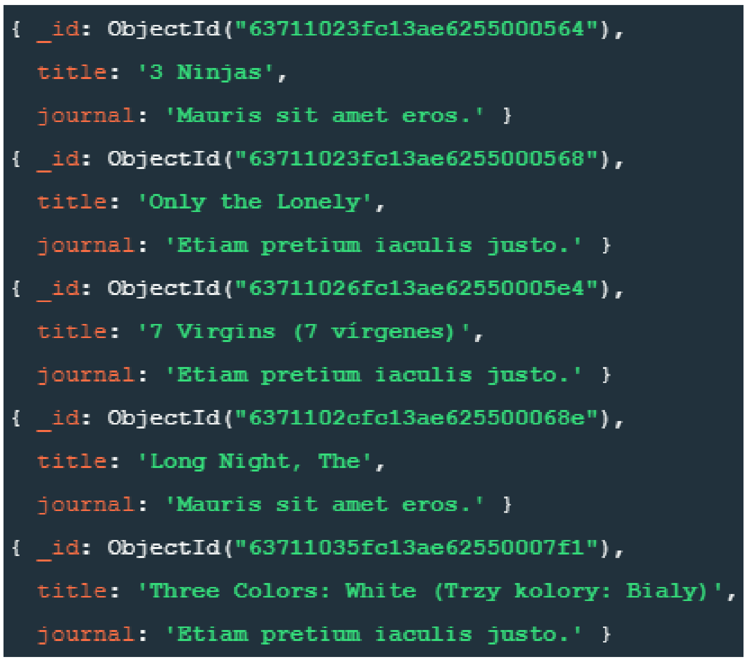
The bibliography contains the ids of the documents inserted by the previous query

2.5. Queries

While working on writing meaningful queries, a few of them were modified after finding a more efficient version thanks to the *.explain("executionStats")* command. Perfomance results are listed after some of the queries.

- **Find 5 documents published in one of two journals: *'Mauris sit amet eros'* or *'Etiam pretium iaculis justo'*:**

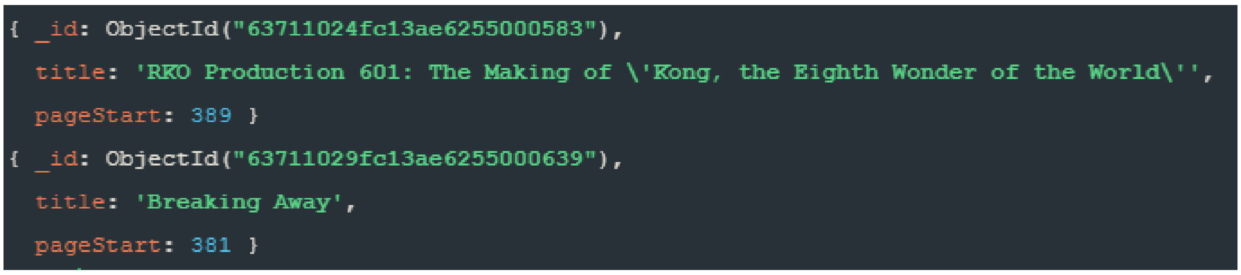
```
db.articles.find({"$or":[{"journal":"Mauris sit amet eros."},{"
    journal":"Etiam pretium iaculis justo."}]},{"title":1,"journal"
    :1}).limit(5)
```



```
{ _id: ObjectId("63711023fc13ae6255000564"),
  title: '3 Ninjas',
  journal: 'Mauris sit amet eros.' }
{ _id: ObjectId("63711023fc13ae6255000568"),
  title: 'Only the Lonely',
  journal: 'Etiam pretium iaculis justo.' }
{ _id: ObjectId("63711026fc13ae62550005e4"),
  title: '7 Virgins (7 vírgenes)',
  journal: 'Etiam pretium iaculis justo.' }
{ _id: ObjectId("6371102cfc13ae625500068e"),
  title: 'Long Night, The',
  journal: 'Mauris sit amet eros.' }
{ _id: ObjectId("63711035fc13ae62550007f1"),
  title: 'Three Colors: White (Trzy kolory: Bialy)',
  journal: 'Etiam pretium iaculis justo.' }
```

- **Find all the documents published in journal *'Sed sagittis.'* that start after page 380, returning both the title and the initial page number:**

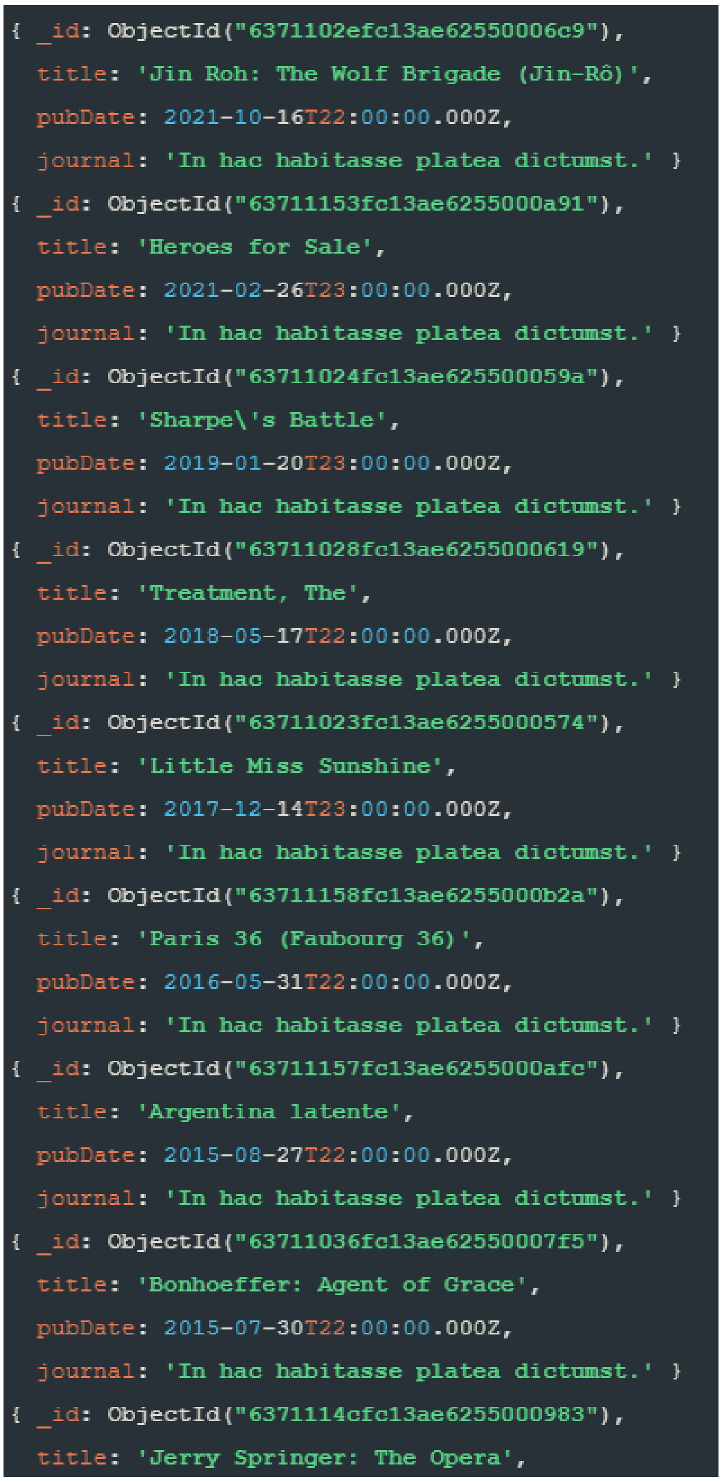
```
db.articles.find({"$and": [{"journal": "Sed sagittis."},{"pageStart
    ": {"$gt": 380}}]},{"title": 1, "pageStart": 1})
```



```
{ _id: ObjectId("63711024fc13ae6255000583"),
  title: 'RKO Production 601: The Making of \'Kong, the Eighth Wonder of the World\'',
  pageStart: 389 }
{ _id: ObjectId("63711029fc13ae6255000639"),
  title: 'Breaking Away',
  pageStart: 381 }
```

- Find all documents published in journal *'In hac habitasse platea dictumst.'* after a certain date. Sort in descending order:

```
db.articles.find({"journal": "In hac habitasse platea dictumst.",
  "pubDate": {"$gt": ISODate('1980-10-29T23:00:00.000Z')}},{ "title
   ":1, "journal":1, "pubDate":1}).sort({"pubDate":-1})
```



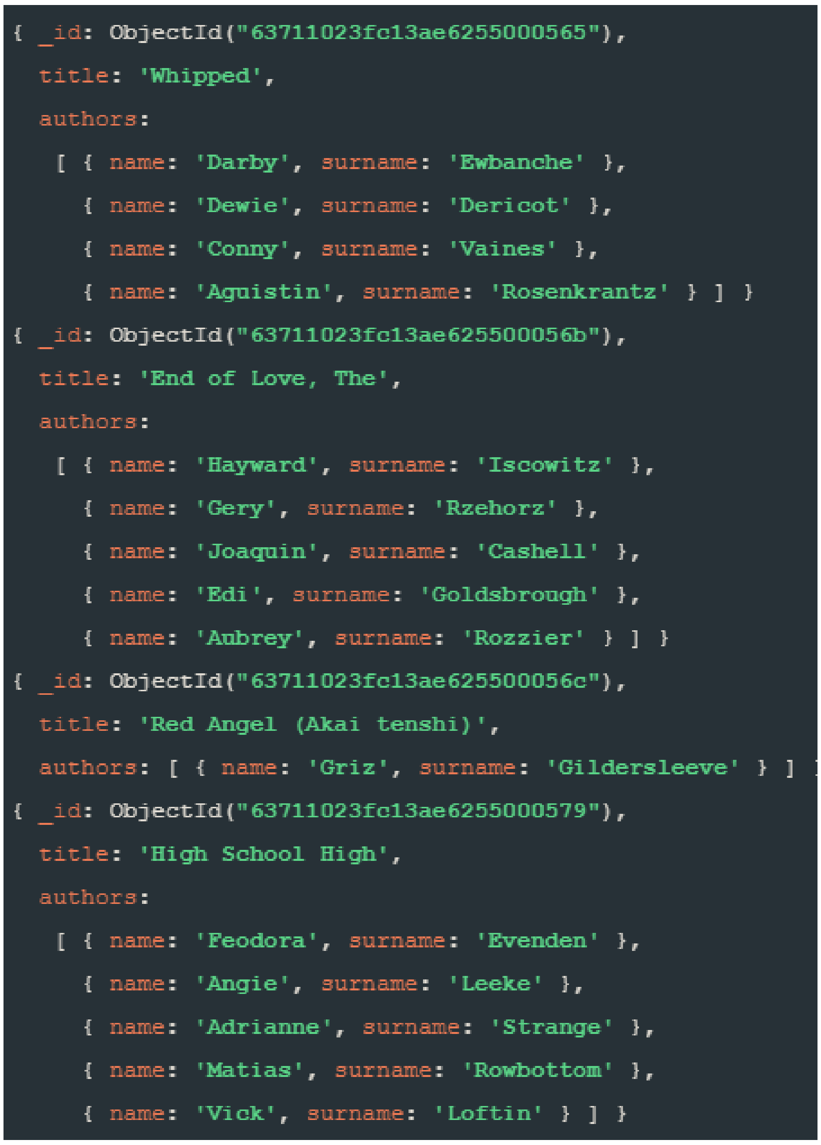
```
{ _id: ObjectId("6371102efc13ae62550006c9"),
  title: 'Jin Roh: The Wolf Brigade (Jin-Rô)',
  pubDate: 2021-10-16T22:00:00.000Z,
  journal: 'In hac habitasse platea dictumst.' }
{ _id: ObjectId("63711153fc13ae6255000a91"),
  title: 'Heroes for Sale',
  pubDate: 2021-02-26T23:00:00.000Z,
  journal: 'In hac habitasse platea dictumst.' }
{ _id: ObjectId("63711024fc13ae625500059a"),
  title: 'Sharpe\'s Battle',
  pubDate: 2019-01-20T23:00:00.000Z,
  journal: 'In hac habitasse platea dictumst.' }
{ _id: ObjectId("63711028fc13ae6255000619"),
  title: 'Treatment, The',
  pubDate: 2018-05-17T22:00:00.000Z,
  journal: 'In hac habitasse platea dictumst.' }
{ _id: ObjectId("63711023fc13ae6255000574"),
  title: 'Little Miss Sunshine',
  pubDate: 2017-12-14T23:00:00.000Z,
  journal: 'In hac habitasse platea dictumst.' }
{ _id: ObjectId("63711158fc13ae6255000b2a"),
  title: 'Paris 36 (Faubourg 36)',
  pubDate: 2016-05-31T22:00:00.000Z,
  journal: 'In hac habitasse platea dictumst.' }
{ _id: ObjectId("63711157fc13ae6255000afc"),
  title: 'Argentina latente',
  pubDate: 2015-08-27T22:00:00.000Z,
  journal: 'In hac habitasse platea dictumst.' }
{ _id: ObjectId("63711036fc13ae62550007f5"),
  title: 'Bonhoeffer: Agent of Grace',
  pubDate: 2015-07-30T22:00:00.000Z,
  journal: 'In hac habitasse platea dictumst.' }
{ _id: ObjectId("6371114cfc13ae6255000983"),
  title: 'Jerry Springer: The Opera',
```

Performances:

```
executionSuccess: true,
nReturned: 31,
executionTimeMillis: 6,
totalKeysExamined: 0,
totalDocsExamined: 1501
```

- **Find 4 documents published after 1980 with their last update before 2010**:

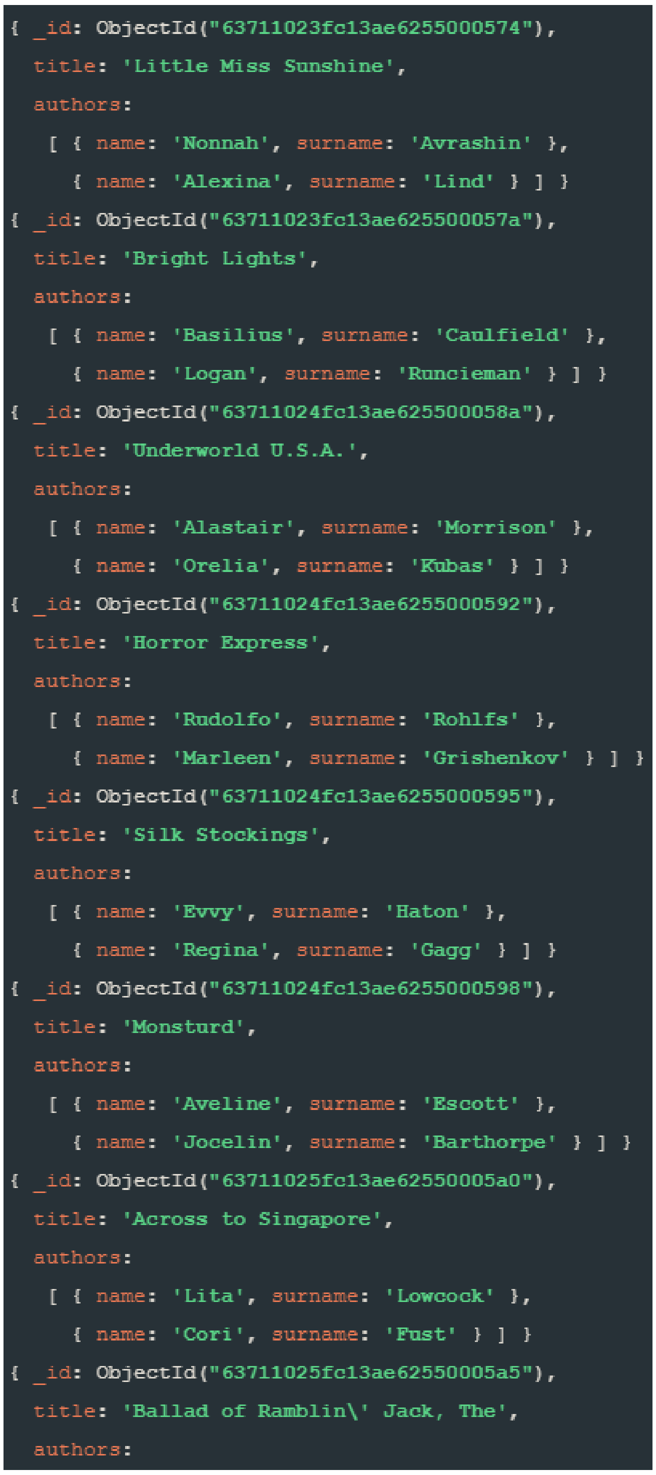
```
db.articles.find({"$and":[{"pubDate":{"$gt":ISODate('1980-01-01T00
    :00:00Z')}},{"lastEdit":{"$lt":ISODate('2010-01-01T00:00:00Z')
    }}]},{"title":1,"authors.name":1,"authors.surname":1}).limit(4)
```



```
{ _id: ObjectId("63711023fc13ae6255000565"),
  title: 'Whipped',
  authors:
   [ { name: 'Darby', surname: 'Ewbanche' },
     { name: 'Dewie', surname: 'Dericot' },
     { name: 'Conny', surname: 'Vaines' },
     { name: 'Aguistin', surname: 'Rosenkrantz' } ] }
{ _id: ObjectId("63711023fc13ae625500056b"),
  title: 'End of Love, The',
  authors:
   [ { name: 'Hayward', surname: 'Iscowitz' },
     { name: 'Gery', surname: 'Rzehorz' },
     { name: 'Joaquin', surname: 'Cashell' },
     { name: 'Edi', surname: 'Goldsbrough' },
     { name: 'Aubrey', surname: 'Rozzier' } ] }
{ _id: ObjectId("63711023fc13ae625500056c"),
  title: 'Red Angel (Akai tenshi)',
  authors: [ { name: 'Griz', surname: 'Gildersleeve' } ]
{ _id: ObjectId("63711023fc13ae6255000579"),
  title: 'High School High',
  authors:
   [ { name: 'Feodora', surname: 'Evenden' },
     { name: 'Angie', surname: 'Leeke' },
     { name: 'Adrianne', surname: 'Strange' },
     { name: 'Matias', surname: 'Rowbottom' },
     { name: 'Vick', surname: 'Loftin' } ] }
```

- **Find all documents with a bibliography and 2 authors:**

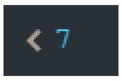
```
db.articles.find({"$and":[{"bibliography":{"$exists":true}},{"
    authors":{"$size":2}}]},{"title":1, "authors.name":1,"authors.
    surname":1})
```



```
{ _id: ObjectId("63711023fc13ae6255000574"),
  title: 'Little Miss Sunshine',
  authors:
   [ { name: 'Nonnah', surname: 'Avrashin' },
     { name: 'Alexina', surname: 'Lind' } ] }
{ _id: ObjectId("63711023fc13ae625500057a"),
  title: 'Bright Lights',
  authors:
   [ { name: 'Basilius', surname: 'Caulfield' },
     { name: 'Logan', surname: 'Runcieman' } ] }
{ _id: ObjectId("63711024fc13ae625500058a"),
  title: 'Underworld U.S.A.',
  authors:
   [ { name: 'Alastair', surname: 'Morrison' },
     { name: 'Orelia', surname: 'Kubas' } ] }
{ _id: ObjectId("63711024fc13ae6255000592"),
  title: 'Horror Express',
  authors:
   [ { name: 'Rudolfo', surname: 'Rohlfs' },
     { name: 'Marleen', surname: 'Grishenkov' } ] }
{ _id: ObjectId("63711024fc13ae6255000595"),
  title: 'Silk Stockings',
  authors:
   [ { name: 'Evvy', surname: 'Haton' },
     { name: 'Regina', surname: 'Gagg' } ] }
{ _id: ObjectId("63711024fc13ae6255000598"),
  title: 'Monsturd',
  authors:
   [ { name: 'Aveline', surname: 'Escott' },
     { name: 'Jocelin', surname: 'Barthorpe' } ] }
{ _id: ObjectId("63711025fc13ae62550005a0"),
  title: 'Across to Singapore',
  authors:
   [ { name: 'Lita', surname: 'Lowcock' },
     { name: 'Cori', surname: 'Fust' } ] }
{ _id: ObjectId("63711025fc13ae62550005a5"),
  title: 'Ballad of Ramblin\' Jack, The',
  authors:
```

- **Count how many documents are in a volume of number greater than 2 of a journal**:

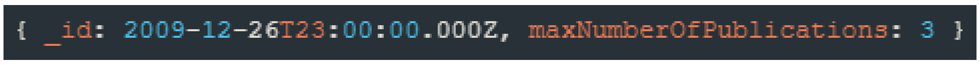
```
db.articles.count({"volume":{"$gt":2},"journal":"Mauris sit amet
    eros."})
```



```
< 7
```

- **Find the day with the biggest number of documents published**:

```
db.articles.aggregate([{"$group":{"_id":"$pubDate","
    maxNumberOfPublications":{"$sum":+1}}},{"$sort":{"
    maxNumberOfPublications":-1}},{"$limit":1}])
```



```
{ _id: 2009-12-26T23:00:00.000Z, maxNumberOfPublications: 3 }
```

Performances:

```
executionSuccess: true,
nReturned: 1455,
executionTimeMillis: 7,
totalKeysExamined: 0,
totalDocsExamined: 1501
```

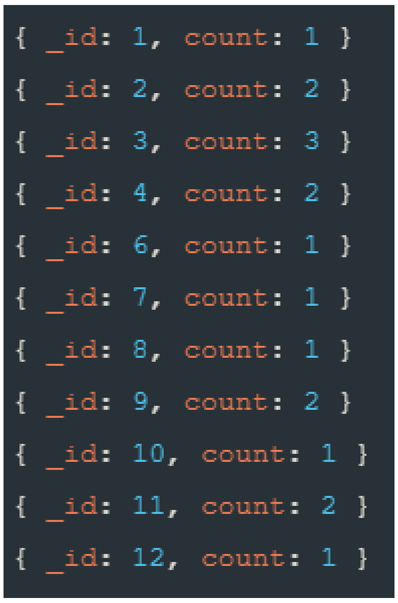
The maximum number of articles published in a day can also be found using two group stages, but we can see that it's less efficient than the previous one:

```
db.articles.aggregate([{"$group":{"_id":"$pubDate","count":{"$sum"
    :+1}}},{"$group":{"_id":true, "maxNumOfPublications":{"$max":"
    $count"}}}])
```

```
executionSuccess: true,
nReturned: 1,
executionTimeMillis: 17,
totalKeysExamined: 0,
totalDocsExamined: 1501
```

- **Count the number of documents published in each month of 2020 and order by month**:

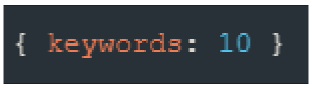
```
db.articles.aggregate([{"$match":{"$and":[{"pubDate":{"$gte":
    ISODate('2020-01-01T00:00:00Z')}},{"pubDate":{"$lt":ISODate('
    2021-01-01T00:00:00Z')}}]}},{"$group":{"_id":{"$month":"$pubDate
    "},"count":{"$sum":1}}},{"$sort":{"_id":1}}])
```



```
{ _id: 1, count: 1 }
{ _id: 2, count: 2 }
{ _id: 3, count: 3 }
{ _id: 4, count: 2 }
{ _id: 6, count: 1 }
{ _id: 7, count: 1 }
{ _id: 8, count: 1 }
{ _id: 9, count: 2 }
{ _id: 10, count: 1 }
{ _id: 11, count: 2 }
{ _id: 12, count: 1 }
```

- **Count how many documents have only the keywords** ***"quam"*** **or** ***"ut"***:

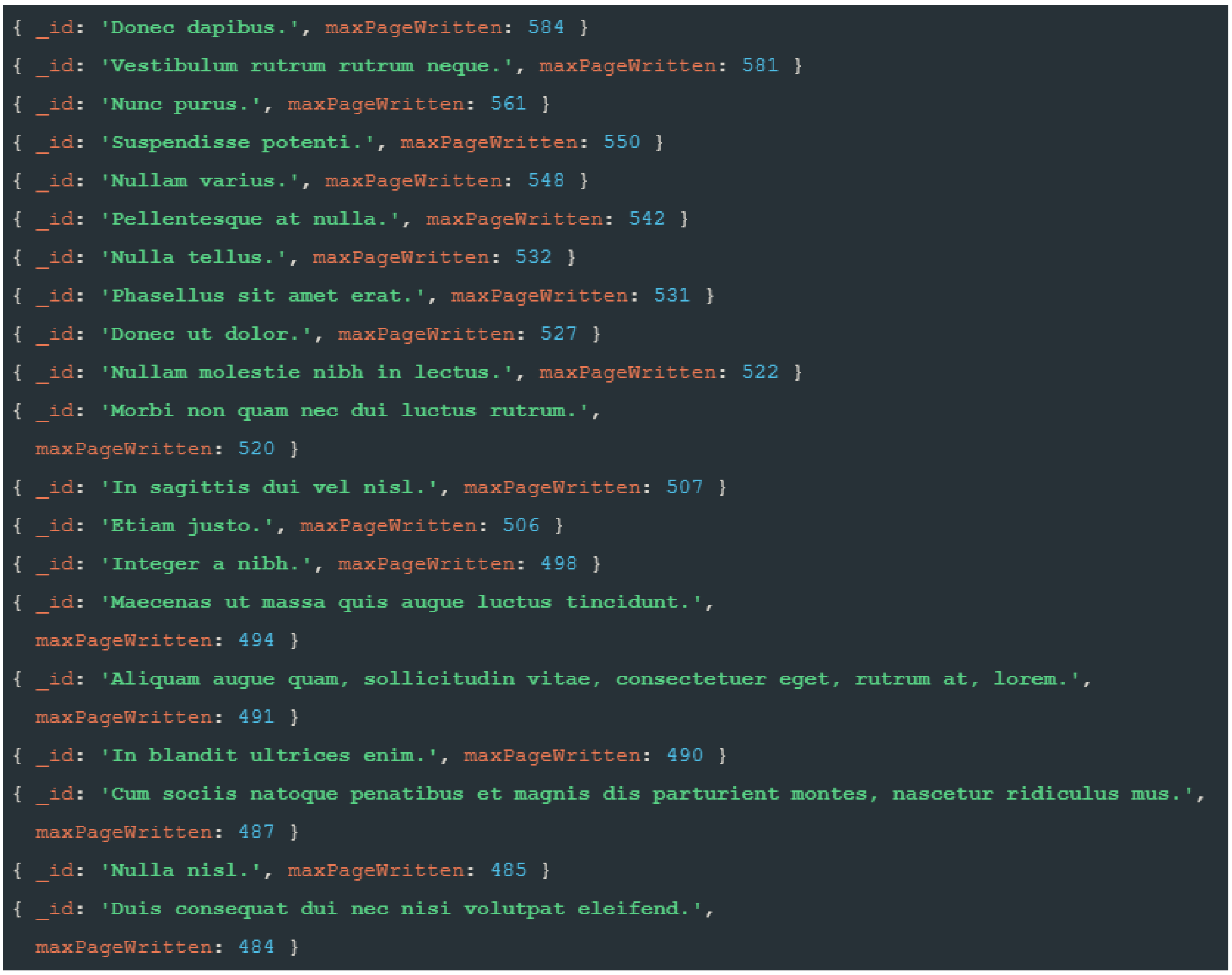
```
db.articles.aggregate([{"$match":{"$or":[{"keywords":"quam"},{"
    keywords":"ut"}]}},{"$count":"keywords"}])
```



```
{ keywords: 10 }
```

- **For each journal, find the article with the most pages. Sort in descending order by number of pages:**

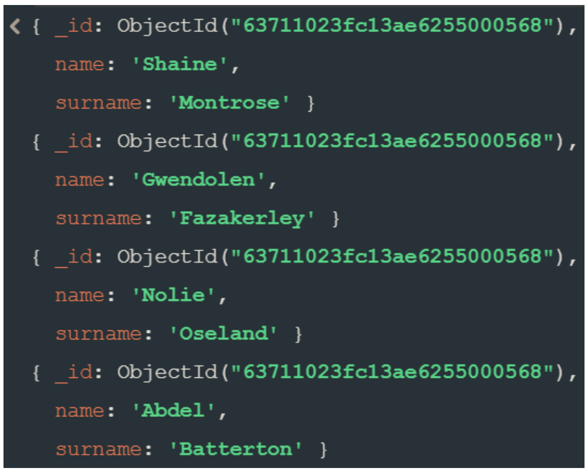
```
db.articles.aggregate([{"$group":{"_id":"$journal","maxPageWritten"
    :{"$max":{"$subtract":["$pageEnd", "$pageStart"]}}}},{"$sort":{"
    maxPageWritten":-1}}])
```



```
{ _id: 'Donec dapibus.', maxPageWritten: 584 }
{ _id: 'Vestibulum rutrum rutrum neque.', maxPageWritten: 581 }
{ _id: 'Nunc purus.', maxPageWritten: 561 }
{ _id: 'Suspendisse potenti.', maxPageWritten: 550 }
{ _id: 'Nullam varius.', maxPageWritten: 548 }
{ _id: 'Pellentesque at nulla.', maxPageWritten: 542 }
{ _id: 'Nulla tellus.', maxPageWritten: 532 }
{ _id: 'Phasellus sit amet erat.', maxPageWritten: 531 }
{ _id: 'Donec ut dolor.', maxPageWritten: 527 }
{ _id: 'Nullam molestie nibh in lectus.', maxPageWritten: 522 }
{ _id: 'Morbi non quam nec dui luctus rutrum.',
  maxPageWritten: 520 }
{ _id: 'In sagittis dui vel nisl.', maxPageWritten: 507 }
{ _id: 'Etiam justo.', maxPageWritten: 506 }
{ _id: 'Integer a nibh.', maxPageWritten: 498 }
{ _id: 'Maecenas ut massa quis augue luctus tincidunt.',
  maxPageWritten: 494 }
{ _id: 'Aliquam augue quam, sollicitudin vitae, consectetuer eget, rutrum at, lorem.',
  maxPageWritten: 491 }
{ _id: 'In blandit ultrices enim.', maxPageWritten: 490 }
{ _id: 'Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.',
  maxPageWritten: 487 }
{ _id: 'Nulla nisl.', maxPageWritten: 485 }
{ _id: 'Duis consequat dui nec nisi volutpat eleifend.',
  maxPageWritten: 484 }
```

- **Unwind on authors of a specific document, retrieving only their names and surnames:**

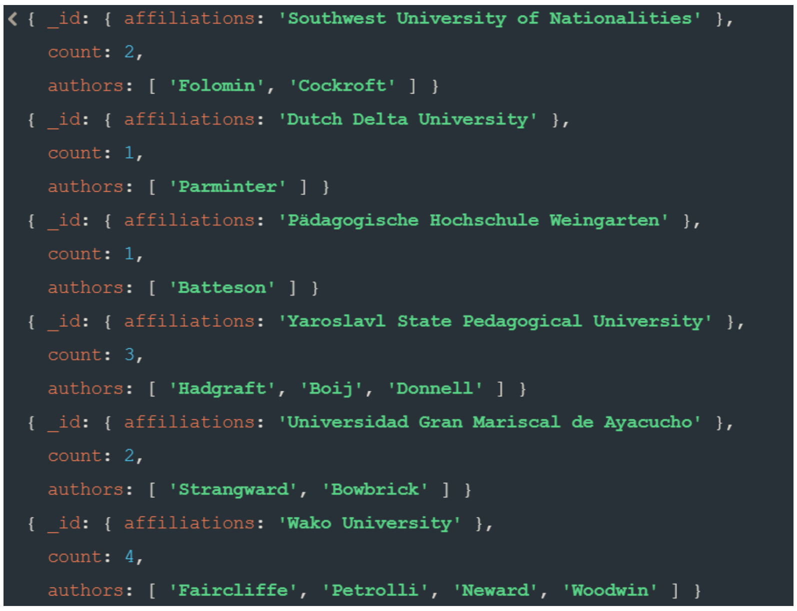
```
db.articles.aggregate([{"$match":{"_id": ObjectId("63711023
    fc13ae6255000568")}},{"$unwind": "$authors"} ,{"$project":{_id:
    true,name: "$authors.name",surname: "$authors.surname"}}])
```



```
< { _id: ObjectId("63711023fc13ae6255000568"),
    name: 'Shaine',
    surname: 'Montrose' }
  { _id: ObjectId("63711023fc13ae6255000568"),
    name: 'Gwendolen',
    surname: 'Fazakerley' }
  { _id: ObjectId("63711023fc13ae6255000568"),
    name: 'Nolie',
    surname: 'Oseland' }
  { _id: ObjectId("63711023fc13ae6255000568"),
    name: 'Abdel',
    surname: 'Batterton' }
```

- **Using unwind, retrieve the authors' surnames and affiliations to count how many and which authors are affiliated to a certain university:**

```
db.articles.aggregate([{"$unwind":"$authors"},{"$unwind":"$authors.
    affiliations"},{"$group": { _id: {affiliations: "$authors.
    affiliations"}, count: {"$count": {}}, authors:{"$addToSet":"
    $authors.surname"}}}])
```



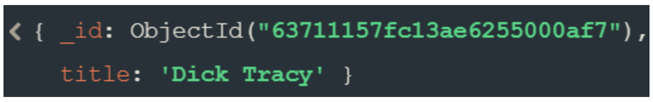
```
< { _id: { affiliations: 'Southwest University of Nationalities' },
    count: 2,
    authors: [ 'Folomin', 'Cockroft' ] }
  { _id: { affiliations: 'Dutch Delta University' },
    count: 1,
    authors: [ 'Parminter' ] }
  { _id: { affiliations: 'Pädagogische Hochschule Weingarten' },
    count: 1,
    authors: [ 'Batteson' ] }
  { _id: { affiliations: 'Yaroslavl State Pedagogical University' },
    count: 3,
    authors: [ 'Hadgraft', 'Boij', 'Donnell' ] }
  { _id: { affiliations: 'Universidad Gran Mariscal de Ayacucho' },
    count: 2,
    authors: [ 'Strangward', 'Bowbrick' ] }
  { _id: { affiliations: 'Wako University' },
    count: 4,
    authors: [ 'Faircliffe', 'Petrolli', 'Neward', 'Woodwin' ] }
```

Performances:

```
executionSuccess: true,
nReturned: 1501,
executionTimeMillis: 52,
totalKeysExamined: 0,
totalDocsExamined: 1501
```

- **Find jobs written by an author named Bob after the year 2000:**

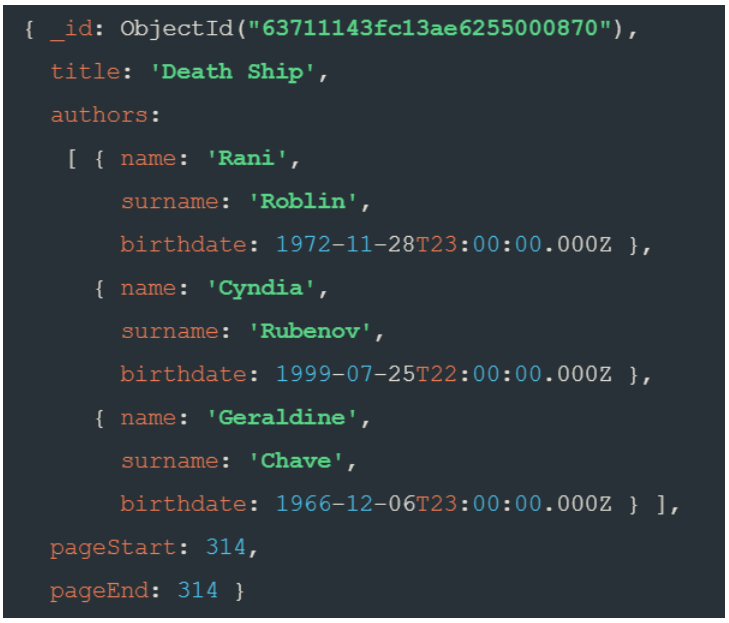
```
db.articles.find({"$and":[{"authors.name":"Bob"},{"pubDate":{"$gt":
    ISODate('2000-01-01T00:00:00Z')}}]},{"title":1})
```



```
< { _id: ObjectId("63711157fc13ae6255000af7"),
    title: 'Dick Tracy' }
```

- **Find all documents of one page written by authors born after a certain date, retrieving only the title, the names of the authors, their birth dates and the pages:**

```
db.articles.find({"$and":[{"pageStart":{"$exists":true}},{"$expr":{
    "$eq":["$pageStart","$pageEnd"]}}, {"authors.birthdate":{"$gt":
    ISODate('1960-01-01T00:00:00Z')}}]},{"title":1,"authors.name":1,
    "authors.surname":1,"authors.birthdate":1,"pageStart":1,"pageEnd
    ":1})
```



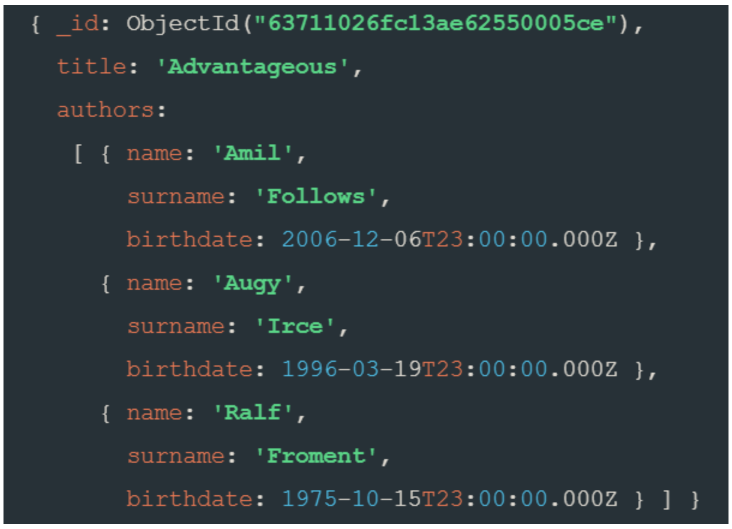
```
{ _id: ObjectId("63711143fc13ae6255000870"),
  title: 'Death Ship',
  authors:
   [ { name: 'Rani',
       surname: 'Roblin',
       birthdate: 1972-11-28T23:00:00.000Z },
     { name: 'Cyndia',
       surname: 'Rubenov',
       birthdate: 1999-07-25T22:00:00.000Z },
     { name: 'Geraldine',
       surname: 'Chave',
       birthdate: 1966-12-06T23:00:00.000Z } ],
  pageStart: 314,
  pageEnd: 314 }
```

Performances:

```
executionSuccess: true,
nReturned: 5,
executionTimeMillis: 16,
totalKeysExamined: 0,
totalDocsExamined: 1501
```

- **Find documents where at least an author born between two certain dates, showing only the title, the authors' names and their birth dates:**

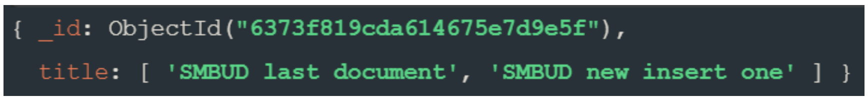
```
db.articles.find({"authors":{"$elemMatch":{"$and":[{"birthdate":{"
    $gt":ISODate('1995-01-01T00:00:00Z')}}, {"birthdate":{"$lt":
    ISODate('2000-01-01T00:00:00Z')}}]}}},{"title":1,"authors.name"
    :1,"authors.surname":1,"authors.birthdate":1})
```



```
{ _id: ObjectId("63711026fc13ae62550005ce"),
  title: 'Advantageous',
  authors:
   [ { name: 'Amil',
       surname: 'Follows',
       birthdate: 2006-12-06T23:00:00.000Z },
     { name: 'Augy',
       surname: 'Irce',
       birthdate: 1996-03-19T23:00:00.000Z },
     { name: 'Ralf',
       surname: 'Froment',
       birthdate: 1975-10-15T23:00:00.000Z } ] }
```

- **Retrieve all the titles of the documents related to the document titled *"SMBUD handbook"*:**

```
db.articles.aggregate([{"$match":{"title":"SMBUD handbook"}},{"
    $lookup":{from:"articles",localField:"bibliography",foreignField
    :"_id",as:"related_docs"}},{"$project":{"_id":true, "title":"
    $related_docs.title"}}])
```



```
{ _id: ObjectId("6373f819cda614675e7d9e5f"),
  title: [ 'SMBUD last document', 'SMBUD new insert one' ] }
```

Performances:

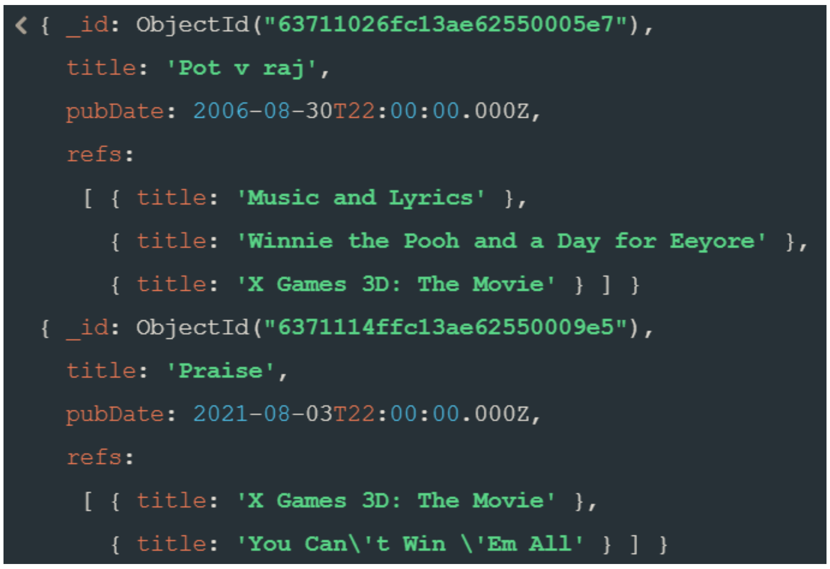
```
executionSuccess: true,
nReturned: 1,
executionTimeMillis: 63,
totalKeysExamined: 2,
totalDocsExamined: 1503
```

By searching the article "SMUBD handbook" with its id instead of the title we can see a decrease in the execution time, but in a real case scenario the user may not have direct access to the id, therefore the user might need an additional query to retrieve it.

```
executionSuccess: true,
nReturned: 1,
executionTimeMillis: 22,
totalKeysExamined: 3,
totalDocsExamined: 3
```

- **Find the name, the publication date and the title of referenced documents of all documents written after 2000 that reference an article called *"X Games 3D: The Movie"*:**

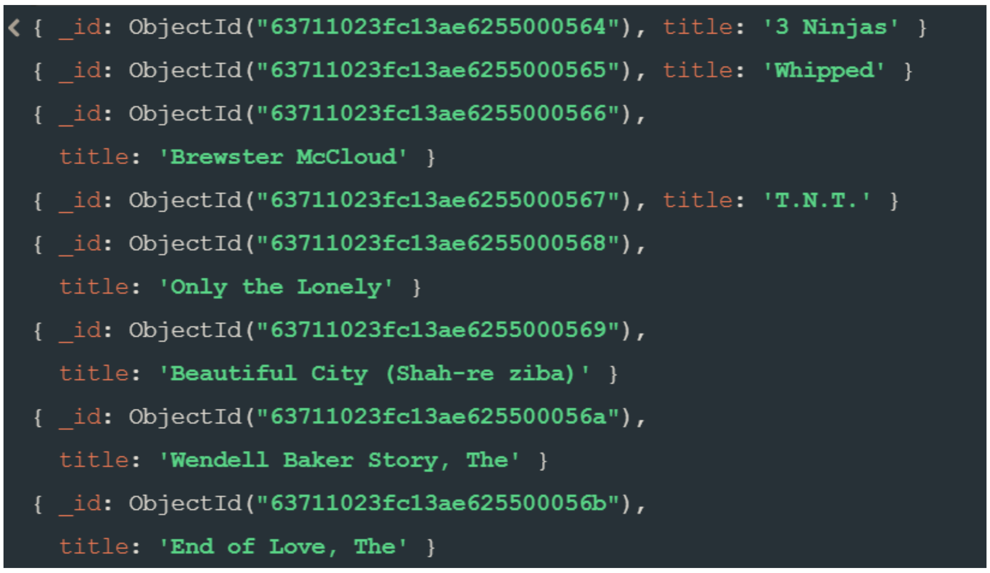
```
db.articles.aggregate([{"$lookup":{from:"articles", localField:"
    bibliography", foreignField:"_id", as:"refs"}},{"$match": {"$and
    ": [{"refs.title": "X Games 3D: The Movie"}, {"pubDate": {"$gt":
     ISODate('2000-01-01T00:00:00Z')}}]}}, {"$project": {"title": 1,
     "pubDate": 1, "refs.title": 1}}])
```



```
< { _id: ObjectId("63711026fc13ae62550005e7"),
    title: 'Pot v raj',
    pubDate: 2006-08-30T22:00:00.000Z,
    refs:
     [ { title: 'Music and Lyrics' },
       { title: 'Winnie the Pooh and a Day for Eeyore' },
       { title: 'X Games 3D: The Movie' } ] }
  { _id: ObjectId("6371114ffc13ae62550009e5"),
    title: 'Praise',
    pubDate: 2021-08-03T22:00:00.000Z,
    refs:
     [ { title: 'X Games 3D: The Movie' },
       { title: 'You Can\'t Win \'Em All' } ] }
```

- **Find the titles of documents with publication date older than the publication date of the documents in their bibliography:**

```
db.articles.aggregate([{"$lookup":{from:"articles",localField:"
    bibliography",foreignField:"_id",as:"new_field"}},{"$match":{"
    $expr":{"$lte":["pubDate","$new_field.lastEdit"]}}},{"$project"
    :{"title":1}}])
```



```
< { _id: ObjectId("63711023fc13ae6255000564"), title: '3 Ninjas' }
  { _id: ObjectId("63711023fc13ae6255000565"), title: 'Whipped' }
  { _id: ObjectId("63711023fc13ae6255000566"),
    title: 'Brewster McCloud' }
  { _id: ObjectId("63711023fc13ae6255000567"), title: 'T.N.T.' }
  { _id: ObjectId("63711023fc13ae6255000568"),
    title: 'Only the Lonely' }
  { _id: ObjectId("63711023fc13ae6255000569"),
    title: 'Beautiful City (Shah-re ziba)' }
  { _id: ObjectId("63711023fc13ae625500056a"),
    title: 'Wendell Baker Story, The' }
  { _id: ObjectId("63711023fc13ae625500056b"),
    title: 'End of Love, The' }
```

3 | Third assignment

3.1. Assignment Description

The purpose of this assignment is handling a big quantity of bibliographic data using Spark. For simplicity, only three entities are used: books, incollection and authors. Spark is an open source project developed by Apache whose main purpose is being a platform for large scale batch, streaming and interactive computations rather than an actual database. The main advantages of Spark are powerful programming model and APIs (Scala, Java, Python), fault-tolerance and efficient memory storage (RDD). For the purpose of our project we used a module built over Spark called SparkSQL (or Shark) with PySpark as its interface. This module provides a programming abstraction, called DataFrame, that can be seen as a kind of relational database table stored in main memory and on which several operations like filtering, joining, grouping can be performed. SparkSQL also acts as a distributed SQL query engine that, thanks to the nature of Spark, is much faster then traditional SQL.

3.2. Dataset Preprocessing

For this delivery it was decided to start from the original dataset, in order to have a fresh copy of the data and be able to freely operate on it. To do this, some pre-processing operations were needed to adapt the data to its usage in Spark.
The first decision taken was to only use some of the available entities, in particular Author, Book and Inproceedings: as they were tightly connected to one another, they allowed to perform queries on a combination of them.

3.2.1. Python scripts

After performing some queries and noticing they were too slow, the dataset was substantially trimmed in order to achieve better performance. This was done by reducing all three entities by a factor of 10: even though much of the data was gone, all the queries

still worked and lead to results, also finishing a lot faster.
Since in the original dataset many authors were not associated with an ORCID, a Python script was used to generate a new one for every author. Each ORCID was generated following the standard ORCID structure and using random numbers.

3.2.2. Spark pre-processing

Since some columns of the three entities were not used and unnecessary, they were dropped from our datasets in order to manipulate only what was useful to achieve our goals.
Then, in order to recreate the relations between Authors, Books and Incollections, some queries were run to create three bridge tables: the first one to associate a Book with its Authors, the second one to associate an Incollection with its Authors, and the last one to associate an Incollection with the Book in which it is contained. The queries used the information contained in the "author" and "booktitle" columns to find the relationships to add in the bridge tables, allowing the association of the IDs of the two entries of the datasets (e.g. incollectionID with its bookID).
In particular, since a book can be associated with various authors (and in that case in the original table their names were concatenated), the "contains" method was used to perform a partial matching between the list of authors and the name of each of them in the Authors table. After this process, the columns used to create the relationships (in the original datasets) were dropped.

3.2.3. Final structure of the data

Here only the structure of the data will be presented, since all the attributes have the same meaning as in the first assignment.

- df_author: ID, author, orcid
- df_book: ID, booktitle, crossref, editor, ee, isbn, key, mdate, month, pages, publisher, school, series, title, url, volume, year
- df_incollection: ID, chapter, crossref, ee, key, mdate, number, pages, publisher, title, url, year
- df_bridge_author_book: bookID, authorID
- df_bridge_book_incollection: bookID, incollectionID
- df_bridge_author_incollection: incollectionID, authorID

3.3. Commands

- **Insert five authors in the dataset. This query was divided in two parts, and each of them was executed in a different way. The first method turned out to be much slower compared to the second one:**

```
#adding three authors with method 1 - slower (creates a list of the entire dataframe and appends)
author_collection = df_author.collect()
author_collection.append({"ID" : 0, "author" : "Francesco Scandale", "orcid" : "0101-9292-8383-7474"})
author_collection.append({"ID" : 1, "author" : "Marta Radaelli", "orcid" : "2345-5678-9823-0022"})
author_collection.append({"ID" : 2, "author" : "Roberto Macaccaro", "orcid" : "3455-4292-8345-1236"})
df_author = spark.createDataFrame(author_collection)

#adding two authors with method 2 - faster (union of two more dataframes)
columns = ['ID', 'author', 'orcid']
vals = [("3", "Elena Montemurro", "2345-9384-4747-1298"),
        ("4", "Luca Rondini", "1234-9876-6574-3456")]

newRow = spark.createDataFrame(vals, columns)
df_author = df_author.union(newRow)
```

- **Add a book associated with two authors:**
  Note: this query is executed before the creation of the bridge tables presented in the previous chapter

```
columns = ["ID","author","booktitle","crossref","editor","ee","isbn","key","mdate","month","pages","publisher","school", \
           "series","title","url","volume","year"]
vals = [("10","Francesco Scandale|Marta Radaelli","SMBUD book","null","Luca Rondini","www.smbud.it/book","123-456-78914-35","sbmud/book","2022-12-5","null","1-150", \
        "Springer","PoliMi","null","SMBUD book","www.smbud.it/book","1","2022")]

newBook = spark.createDataFrame(vals, columns)
df_book = df_book.union(newBook)
df_book.filter(df_book["ID"]=="10").show()
```

- **Modify the book to add other authors (this is actually done by deleting the original book and recreating it):**

```
#update our book: DataFrames are immutable, so we need to delete the inserted one and create a new row
df_book = df_book.filter(df_book["ID"]!="10")
columns = ["ID","author","booktitle","crossref","editor","ee","isbn","key","mdate","month","pages","publisher","school", \
           "series","title","url","volume","year"]
vals = [("10","Francesco Scandale|Marta Radaelli|Luca Rondini|Roberto Macaccaro|Elena Montemurro","SMBUD book","null","Luca Rondini","www.smbud.it/book","123-456-78914-35","sbmud/book","2022-12-5","null","1-150", \
        "Springer","PoliMi","null","SMBUD book","www.smbud.it/book","1","2022")]
updateBook = spark.createDataFrame(vals, columns)
df_book = df_book.union(updateBook)
print(df_book.filter(df_book["ID"]=="10").select("author").collect())
```

- **Delete Luca Rondini from the authors of the book created before:**

```
lucaID = df_author.filter(df_author["author"]=="Luca Rondini").select("ID").collect()
df_bridge_author_book = df_bridge_author_book.filter((df_bridge_author_book["authorID"]!=lucaID[0]["ID"]) | (df_bridge_author_book["bookID"]!="10"))
df_bridge_author_book.filter(df_bridge_author_book["bookID"]=="10").show()
```

- **Create an incollection and associate it to the book created before:**

```
#adding an incollection to our book
columns = ["ID","chapter","crossref","ee","key","mdate","number","pages","publisher","title","url","year"]
vals = [("100","1","null","www.smbud.it/book/chapter1","sbmud/book/chapter1","2022-12-4","null","20-35", \
         "Springer","SMBUD book, chapter 1","www.smbud.it/book/chapter1","2022")]

newIncollection = spark.createDataFrame(vals, columns)
df_incollection = df_incollection.union(newIncollection)
df_incollection.filter(df_incollection["ID"]=="100").show()


#modifying the bridge tables
columns = ["bookID","incollectionID"]
vals = [("10","100")]
newBookIncollection = spark.createDataFrame(vals, columns)
```

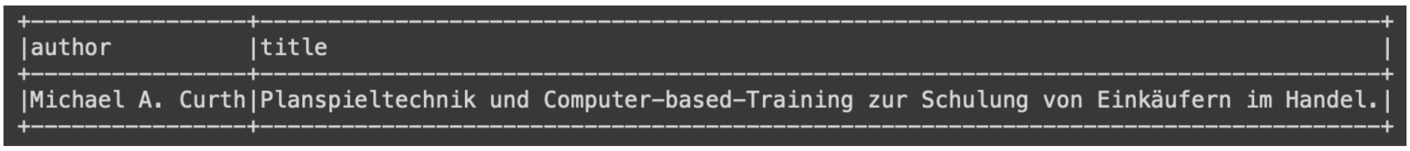
```
df_bridge_book_incollection = df_bridge_book_incollection.union(
    newBookIncollection)
df_bridge_book_incollection.filter(df_bridge_book_incollection["
    incollectionID"]=="100").show()

columns = ["incollectionID","authorID"]
vals = [("100","3")]
newAuthorIncollection = spark.createDataFrame(vals, columns)
df_bridge_author_incollection = df_bridge_author_incollection.union
    (newAuthorIncollection)
df_bridge_author_incollection.filter(df_bridge_author_incollection[
    "incollectionID"]=="100").show()
```

3.4. Queries

- **Find the titles of the books written by Michael A. Curth:**

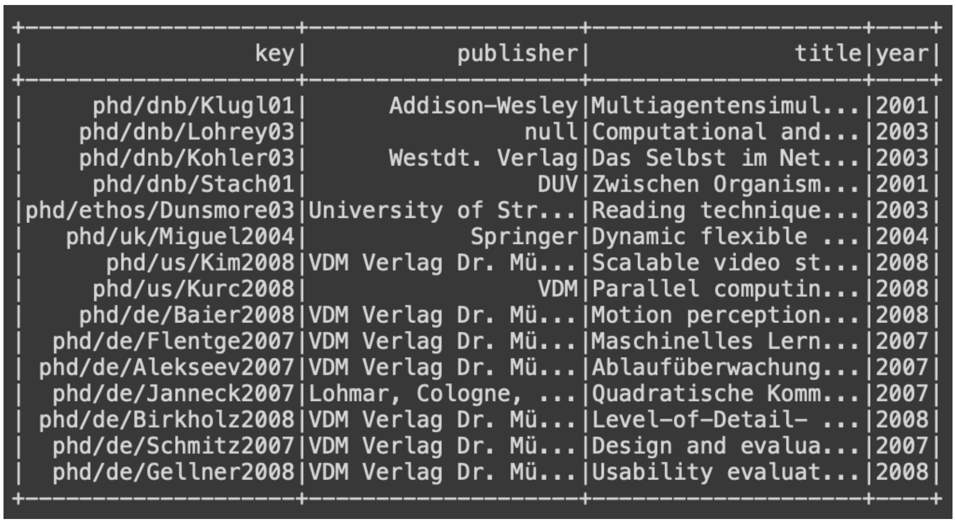
```python
df_author.filter(col("author")=="Michael A. Curth") \
.join(df_bridge_author_book, df_author.ID == df_bridge_author_book.
    authorID, "inner") \
.join(df_book, df_book.ID == df_bridge_author_book.bookID, "inner")
.select("author","title") \
.show(truncate=False)
```



```
+----------------+-----------------------------------------------------------------------------------+
|author          |title                                                                              |
+----------------+-----------------------------------------------------------------------------------+
|Michael A. Curth|Planspieltechnik und Computer-based-Training zur Schulung von Einkäufern im Handel.|
+----------------+-----------------------------------------------------------------------------------+
```

- **Show 15 phd thesis (assuming a book is a phd thesis when the key starts with "phd/") published between 2001 and 2009**

```python
df_book.filter((col("year") > 2000) \
  & (col("year") < 2010) \
  & (col("key").like("phd/%")))\
.limit(15).select("key","publisher","title","year") \
.show()
```

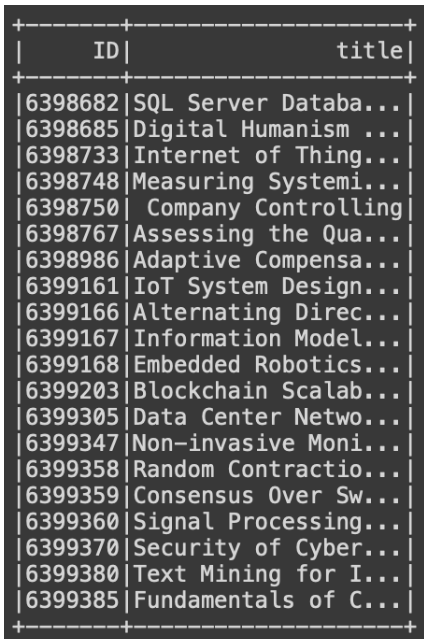


```
+--------------------+--------------------+--------------------+----+
|                 key|           publisher|               title|year|
+--------------------+--------------------+--------------------+----+
|     phd/dnb/Klugl01|      Addison-Wesley|Multiagentensimul...|2001|
|    phd/dnb/Lohrey03|                null|Computational and...|2003|
|    phd/dnb/Kohler03|      Westdt. Verlag|Das Selbst im Net...|2003|
|     phd/dnb/Stach01|                 DUV|Zwischen Organism...|2001|
|phd/ethos/Dunsmore03|University of Str...|Reading technique...|2003|
|   phd/uk/Miguel2004|            Springer|Dynamic flexible ...|2004|
|     phd/us/Kim2008|VDM Verlag Dr. Mü...|Scalable video st...|2008|
|     phd/us/Kurc2008|                 VDM|Parallel computin...|2008|
|    phd/de/Baier2008|VDM Verlag Dr. Mü...|Motion perception...|2008|
|  phd/de/Flentge2007|VDM Verlag Dr. Mü...|Maschinelles Lern...|2007|
| phd/de/Alekseev2007|VDM Verlag Dr. Mü...|Ablaufüberwachung...|2007|
|  phd/de/Janneck2007|Lohmar, Cologne, ...|Quadratische Komm...|2007|
| phd/de/Birkholz2008|VDM Verlag Dr. Mü...|Level-of-Detail- ...|2008|
|  phd/de/Schmitz2007|VDM Verlag Dr. Mü...|Design and evalua...|2007|
|  phd/de/Gellner2008|VDM Verlag Dr. Mü...|Usability evaluat...|2008|
+--------------------+--------------------+--------------------+----+
```

- **Filter all the book IDs written in 2022, and then find them again in the dataset:**

```python
new_books=df_book.filter(col("year")>=2022).select("ID").collect()
new_books = [new[0] for new in new_books]
```

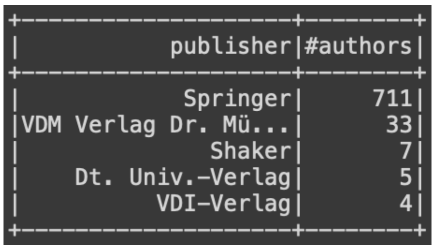
```
df_book.filter(col("ID").isin(new_books)) \
.select("ID","title") \
.show()
```



```
+-------+--------------------+
|     ID|               title|
+-------+--------------------+
|6398682|SQL Server Databa...|
|6398685|Digital Humanism ...|
|6398733|Internet of Thing...|
|6398748|Measuring Systemi...|
|6398750| Company Controlling|
|6398767|Assessing the Qua...|
|6398986|Adaptive Compensa...|
|6399161|IoT System Design...|
|6399166|Alternating Direc...|
|6399167|Information Model...|
|6399168|Embedded Robotics...|
|6399203|Blockchain Scalab...|
|6399305|Data Center Netwo...|
|6399347|Non-invasive Moni...|
|6399358|Random Contractio...|
|6399359|Consensus Over Sw...|
|6399360|Signal Processing...|
|6399370|Security of Cyber...|
|6399380|Text Mining for I...|
|6399385|Fundamentals of C...|
+-------+--------------------+
```

- **Show how many authors published at least one book with each publisher:**
  Note: filter col("publisher")!="null" is needed because the original dataset is incomplete

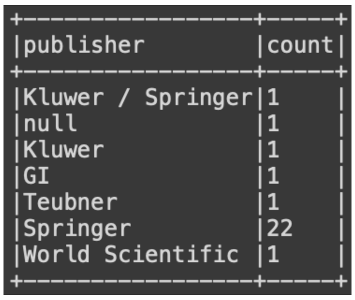
```
query1 = df_book.join(df_bridge_author_book,df_bridge_author_book["
    bookID"]==df_book["ID"],"inner") \
.groupBy("publisher").count().withColumnRenamed("count","#authors")
.filter((col("#authors")>1) & (col("publisher")!="null")) \
.sort(col("#authors").desc()).limit(5).show()
```



```
+--------------------+--------+
|           publisher|#authors|
+--------------------+--------+
|            Springer|     711|
|VDM Verlag Dr. Mü...|      33|
|              Shaker|       7|
|    Dt. Univ.-Verlag|       5|
|          VDI-Verlag|       4|
+--------------------+--------+
```

- **Show how many books each publisher has published in the year 2000:**

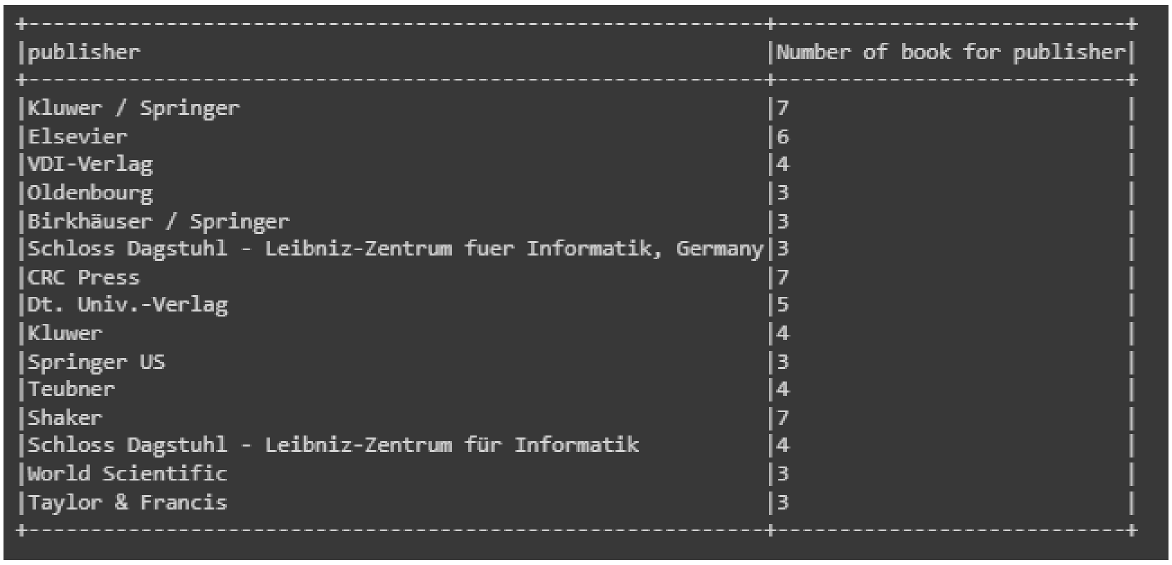
```
df_book.filter(col("year") == 2000) \
.groupBy("publisher").count() \
.show(truncate = False)
```



```
+-----------------+-----+
|publisher        |count|
+-----------------+-----+
|Kluwer / Springer|1    |
|null             |1    |
|Kluwer           |1    |
|GI               |1    |
|Teubner          |1    |
|Springer         |22   |
|World Scientific |1    |
+-----------------+-----+
```

- **Show the publishers that published 3 to 7 books and the number of books:**

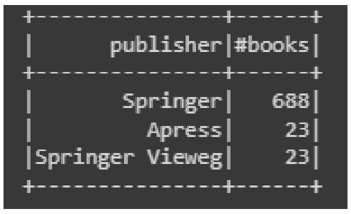
```
df_book.groupBy("publisher").agg(count("ID") \
.alias("Number of book for publisher")) \
.filter((col("Number of book for publisher")>=3)\
& (col("Number of book for publisher")<=7))\
.show(truncate = False)
```



```
+---------------------------------------------------------+----------------------------+
|publisher                                                |Number of book for publisher|
+---------------------------------------------------------+----------------------------+
|Kluwer / Springer                                        |7                           |
|Elsevier                                                 |6                           |
|VDI-Verlag                                               |4                           |
|Oldenbourg                                               |3                           |
|Birkhäuser / Springer                                    |3                           |
|Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany|3                         |
|CRC Press                                                |7                           |
|Dt. Univ.-Verlag                                         |5                           |
|Kluwer                                                   |4                           |
|Springer US                                              |3                           |
|Teubner                                                  |4                           |
|Shaker                                                   |7                           |
|Schloss Dagstuhl - Leibniz-Zentrum für Informatik        |4                           |
|World Scientific                                         |3                           |
|Taylor & Francis                                         |3                           |
+---------------------------------------------------------+----------------------------+
```

- **Filter books with attribute "year" greater than 2015. Group by publisher and keep just who published more than 20 books with the corresponding number of books:**

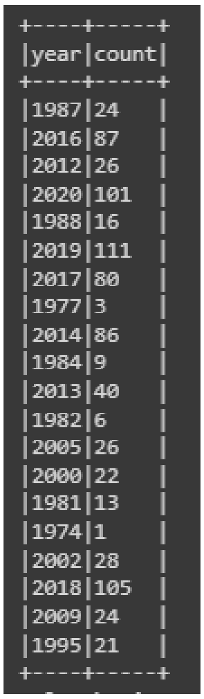
```
query2 = df_book.filter(col("year")>2015).groupBy("publisher"). \
count().withColumnRenamed("count","#books").filter(col("#books")
    >20). \
sort(col("#books").desc())
query2.show()
```



```
+---------------+------+
|      publisher|#books|
+---------------+------+
|       Springer|   688|
|         Apress|    23|
|Springer Vieweg|    23|
+---------------+------+
```

- Find books whose titles are in the set of titles of the books published by Springer; group by year:

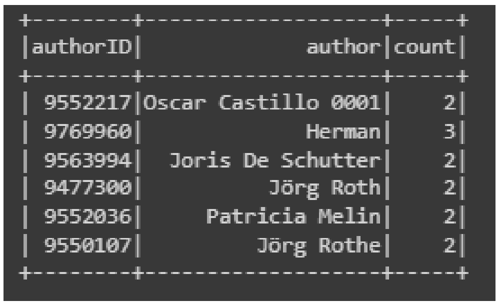
```
springer_books=df_book.filter(col("publisher")=="Springer").select(
    "title").collect()
springer_books = [new[0] for new in springer_books]
df_book.filter(col("title").isin(springer_books)).groupBy("year").
    count().show(truncate = False)
```



```
+----+-----+
|year|count|
+----+-----+
|1987|24   |
|2016|87   |
|2012|26   |
|2020|101  |
|1988|16   |
|2019|111  |
|2017|80   |
|1977|3    |
|2014|86   |
|1984|9    |
|2013|40   |
|1982|6    |
|2005|26   |
|2000|22   |
|1981|13   |
|1974|1    |
|2002|28   |
|2018|105  |
|2009|24   |
|1995|21   |
+----+-----+
```

- Show how many books have been written by every author, considering only books written between 2001 and 2010 and authors that have written at least 3 books:

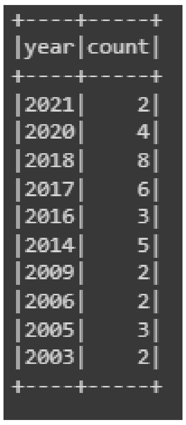
```
df_book.filter((col("year") > 2000) & (col("year") < 2011)) \
.join(df_bridge_author_book, df_book.ID == df_bridge_author_book.
    bookID) \
.join(df_author, df_author.ID == df_bridge_author_book.authorID) \
.groupBy(["authorID", "author"]) \
.count().alias("count") \
.filter(col("count")>1) \
.show()
```



```
+--------+------------------+-----+
|authorID|            author|count|
+--------+------------------+-----+
| 9552217|Oscar Castillo 0001|    2|
| 9769960|            Herman|    3|
| 9563994|  Joris De Schutter|    2|
| 9477300|          Jörg Roth|    2|
| 9552036|    Patricia Melin|    2|
| 9550107|         Jörg Rothe|    2|
+--------+------------------+-----+
```

- **Show the number of books published by authors whose names start with A; group by year (starting from 1996) showing just the ones with at least 2 books:**

```
df_book.filter(col("year")>1995) \
.join(df_bridge_author_book, df_book.ID == df_bridge_author_book.
   bookID) \
.join(df_author, df_author.ID == df_bridge_author_book.authorID) \
.filter(col("author").like("A%")) \
.groupBy("year") \
.agg(count("year").alias("count")) \
.filter(col("count")>=2) \
.sort(col("year").desc()) \
.show()
```



```
+----+-----+
|year|count|
+----+-----+
|2021|    2|
|2020|    4|
|2018|    8|
|2017|    6|
|2016|    3|
|2014|    5|
|2009|    2|
|2006|    2|
|2005|    3|
|2003|    2|
+----+-----+
```

4 | References and sources

- LaTeX editor

4.1. First assignment

- Source Database
- DBLP to csv parser
- Tool for ER modelling
- Neo4J
- Java JDK 11

4.2. Second assignment

- Dataset generator
- Python 3
- MongoDB

4.3. Third assignment

- Source Database
- Python 3
- Scripting environment for Spark