POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Riccardo Cocola**
Alessandro De Florio
Lorenzo Male
Leonardo Marazzi
Cosimo Sguanci

Group Number: **32**

Academic Year: 2022-2023

Contents

1 | Description of the problem

1.1. Description

The problem consists of creating an information system that collects bibliographic information on computer science scientific publications. The database contains information regarding various types of publications (e.g. journal articles, proceedings, theses,...), their authors and publishers, as well as where they are published.

1.2. Assumptions

The assumptions that have been made during the design of the information system are the following:

- A **Person** can be both an **Author** and an **Editor** of some **Publications**.
- Every entity has an unique ID, which will be used as primary key in the E-R, and as key in the graph database.
- The ORCID is an optional attribute and therefore cannot be used as the identifier of a **Person**.
- A **Thesis** cannot be directly included in a **Journal**, **Proceeding** or a **Book**, but another **Publication** must be added with a different type. For example, a **Journal** article may be derived from a **PhD Thesis**.
- An **Article** is not necessarily published: for example, a *preprint* is not published in a **Journal**, **Proceeding**, or **Book**.

1.3. Identified Entities and Relationships

1.3.1. Entities

The identified entities in the problem are the following:

- **Person**, which is an "abstract" entity (it forms a *total* generalization), which has two specializations: **Author** and **Editor**. Since an **Author** can also be an **Editor**, this is a *total* and *overlapping* specialization.
- **Publication**, which is an abstract entity representing a published work in a generic way. Its specializations are:
  - **InProceeding**: a paper presented at a conference and included in a **Proceeding**.
  - **Article**: a paper which is published on a scientific **Journal**.
  - **InCollection**: a work which is contained in a **Book**.
  - **Thesis**: a total and exclusive generalization which can be specified in a **PhD Thesis** or a **Master's Thesis**.
- **PublicationContainer**: a total and exclusive generalization which represents where papers are actually published. Its "concrete" subclasses are:
  - **Proceeding**: a collection of academic papers published and presented in the context of an academic conference.
  - **Journal**: a periodical publication in which peer-reviewed academic articles are published.
  - **Book**: another type of collection of academic research works.
- **Series**: a grouping of various publications. It is optional, therefore a **Publication** can be part of a series or not.
- **Publisher**: the publisher of a collection of research works.

Attributes

- **Person/Author/Editor:**
  - ID (Primary Key)
  - Name
  - ORCID
- **Publication/InProceeding/Article/InCollection:**
  - ID (Primary Key)
  - Title

  - Year
  - Month
  - DOI
  - Link
- **Thesis/PhdThesis/MastersThesis:**
  - School
- **PublicationContainer/Proceeding/Journal/Book:**
  - ID (Primary Key)
  - Name
- **Proceeding:**
  - Year
  - ISBN
  - DOI
- **Book:**
  - Year
  - ISBN
- **Series:**
  - ID (Primary Key)
  - Name
- **Publisher:**
  - ID (Primary Key)
  - Name

1.3.2. Relationships

The relationships between entities are shown below:

- **WRITTEN BY**: a **Publication** is written by some **Authors**
- **EDITED BY**: a collection of academic papers can be edited by some **Editors**

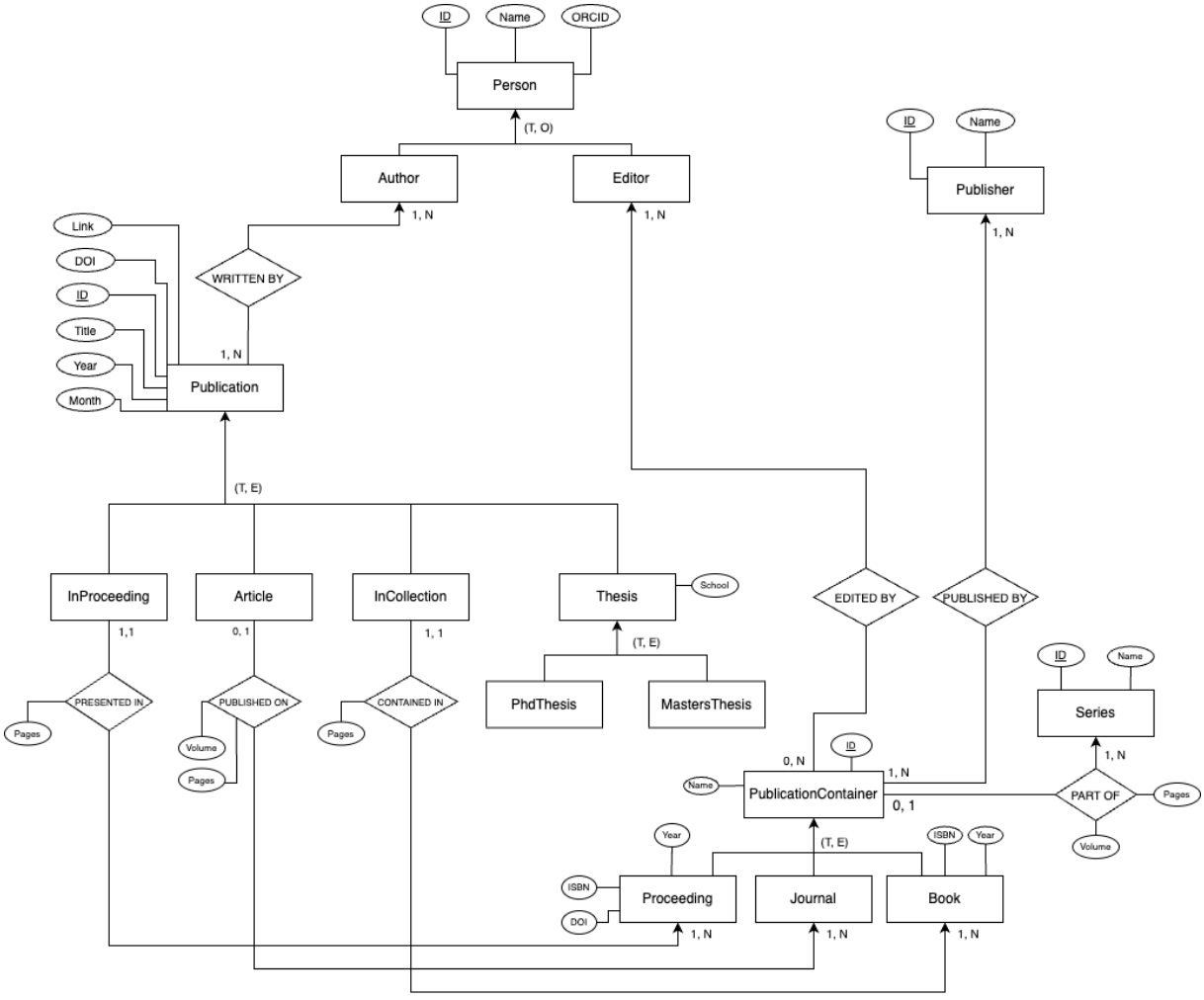
- **PUBLISHED BY**: a collection of academic papers is published by a **Publisher**
- **PART OF**: a **Publication** can be part of a **Series**
- **PRESENTED IN**: an **InProceeding** is presented in a **Proceeding**
- **PUBLISHED ON**: an **Article** can be published on an academic **Journal**
- **CONTAINED IN**: an **InCollection** publication is contained in a **Book**

In addition, we have the following attributes contained in relationships:

- **PRESENTED IN**:
  - Pages
- **PUBLISHED ON**:
  - Volume
  - Pages
- **CONTAINED IN**:
  - Pages
- **PART OF**:
  - Volume
  - Pages

2 | ER Diagram

As an explanatory example, a Publication can be part of 0 or 1 Series, while a Series can include 1 to N Publications



3 | Dataset Description

3.1. General description of the dataset

The *dblp* dataset[1] is in XML format and its size is about 3.5 GB. After pre-processing the database size has been reduced to ≈ 1.5 MB and contains 9288 nodes and 11431 relationships:

- 3490 publications
- 2770 authors or editors
- 19 schools
- 18 journals
- 13 publishers
- 9 series
- 2969 links (e.g., URLs to publication's web pages)

In addition, as will be explained in depth in the data-preprocessing section, we wanted to include from the dataset at least a minimum number (10 in this example) of publications of each type:

- 2867 `incollection`
- 10 `inproceedings`
- 10 `proceedings`
- 12 `books`
- 10 `phdthesis`
- 10 `mastersthesis`
- 565 `articles` published in a journal

[1] It can be found on `https://dblp.uni-trier.de/xml/`

The dataset comes with a DTD (Document Type Definition), which however is not very strict, as the dblp website states in the description of the dataset. Therefore, due to the few restrictions made on the XML file, tags can have child tags which are not expected (for example a `<school>` in a journal `<article>`). While we must try to enforce some sort of structure of the data on the E-R diagram, we have to be quite flexible in the actual graph database.

4 | Data Upload

4.1. Dataset pre-processing

An extensive pre-processing was needed to support the creation of the graph database and the subsequent queries. All the pre-processing was performed through a Python script which parses the dataset XML file.

4.1.1. Usage of the DTD

The DTD, which can be found with the *dblp* dataset, was used mainly to represent the special characters contained in the XML file, defining the mapping between the codes used in the file and special characters, compatible with our graph database.

4.1.2. Reduction of the size of the dataset

To make the dataset usable within Neo4j, it was necessary to reduce the size of the XML file. For this purpose, the pre-processing script reads the XML line by line, and limits the size of the dataset to a certain (configurable) number of lines (30,000 at the moment).
It was decided to discard the *www* tag, since it was not considered relevant in this context. Selecting a subset of the total number of lines of the XML file made the dataset "unbalanced", because the resulting reduced dataset did not contain an instance of all the entities. Therefore, the pre-processing of the dataset was modified to include at least k instances of all the entities: this means that, for example, once we include the first 30,000 lines of the dataset, we continue until we have included at least k of all the main publication types. In this way we were able to make the dataset more balanced and useful to create search queries which return meaningful information.

4.2. Upload process

In order to load all the relevant information from a local XML file, the "APOC for Neo4j" plugin was used. Thus, in order to replicate the database on another local machine in-

stalling the plugin is mandatory, as well as configuring it to allow local data upload.
A single db operation is needed to load the XML file onto the DB, this operation first loads from the XML a table where every row is some type of publication, loaded with all the relevant information, such as its title, its year of publication and similar information which are attributes of the publication itself.

```
CALL apoc.load.xml("file:///db.xml") yield value
UNWIND value._children as p
WITH p.id as elem_id, p._type as type,
[item in p._children where item._type = "title "][0]. _text as title ,
[item in p._children where item._type = "year"][0]._text as year,
[item in p._children where item._type = "month"][0]._text as month,
[item in p._children where item._type = "isbn"][0]._text as isbn_book,
[item in p._children where item._type = "booktitle"][0]._text as
    book_title_incollection,
[item in p._children where item._type = "volume"][0]._text as volume,
[item in p._children where item._type = "pages"][0]._text as pages,
```

Each publication in the XML file also contains attributes which may be shared with others, such as its authors/editors, its publisher, or the series it belongs to; these attributes also have a tendency of being of variable cardinality (meaning that multiple of them may appear in the same publication or they may be absent as well, e.g. an article with multiple authors but no editors).

```
[item in p._children where item._type = "author"] as authors,
[item in p._children where item._type = "editor"] as editors,
[item in p._children where item._type = "journal"] as journals,
[item in p._children where item._type = "school"] as schools,
[item in p._children where item._type = "publisher"] as publishers,
[item in p._children where item._type = "series"] as serieses ,
[item in p._children where item._type = "ee"] as ee,
[item in p._children where item._type = "crossref"] as crossref
```

All these attributes are then generated, if absent from the DB, or updated if already present, through the merge command; at the same time, fitting relationships are established between the publication and the aforementioned elements.[1]

[1]Bear in mind: this is supposed to be a single operation, these three blocks are not single functions and are only split for clarity's sake.

```
MERGE (e:Publication {id:elem_id})
SET e. title  = title ,e.type = type, e.year = year, e.month = month, e.isbn =
    isbn_book, e.booktitle = book_title_incollection, e.pages = pages, e.volume
    = volume

WITH *, e, e.type as typ
CALL apoc.create.addLabels(e, [typ]) YIELD node
remove e.type
FOREACH (author in authors|
    merge (a: Author {name: author._text})
    SET a.orcid = author.orcid
    MERGE (e) - [:WRITTEN_BY] -> (a))

FOREACH (author in editors|
    MERGE (a: Author {name: author._text})
    SET a.orcid = author.orcid
    MERGE (e) - [:EDITED_BY] -> (a))

FOREACH (publisher in publishers|
    MERGE(pub: Publisher {name: publisher._text})
    MERGE (e) - [:PUBLISHED_BY] -> (pub))

FOREACH (journal in journals|
    MERGE (j: Journal {name:journal._text})
    MERGE (e) - [r:PUBLISHED_ON] -> (j)
    SET r.volume = volume, r.pages= pages)

FOREACH (school in schools|
    MERGE (sc: School {name:school._text})
    MERGE (e) -[:FOR_SCHOOL]-> (sc))

FOREACH (series in serieses |
    MERGE (s:Series {name:series._text})
    MERGE (e) - [r:PART_OF] -> (s)
    SET r.volume = volume, r.pages = pages)

FOREACH (link in ee|
    MERGE(l: link {url:link._text})
    MERGE (e) - [:LINK] -> (l))

FOREACH (ref in crossref |
    MERGE (other:Publication {id:ref._text})
```

```
MERGE (e) −[r:REFERENCES]−> (other)
SET r.volume = volume, r.pages= pages)
```

5 | Graph Diagram

5.1. Nodes

The Graph database has been built with the intent of being as faithful as possible to the proposed E-R diagram within the limits of reason.
Each publication has a second label which identifies its type (i.e. whether they are a book rather than an article or others), and the following attributes:

- an id, which uniquely identifies the publication;
- its title;
- the publication's month and year;
- its International Standard Book Number (or ISBN);
- the title of the book the publication belongs to (if any);
- the volume it is contained it as well as the pages in which it appears.

As one may notice, some attributes do not make sense for some types of publication (e.g. an article cannot have an ISBN since it is not a book). This is due to the fact that the dblp dataset, since it belongs to a bibliographic database, was meant to be as flexible as possible in regards to the available information (for example one may want to add a book to the db without knowing its ISBN or publication year).
This, though isn't a problem with a graph database such as Neo4j: a field, if null, is discarded and thus does not appear as a field (e.g. an article won't ever have an ISBN unless there are errors in the given dataset).

The other node types are more streamlined, since little to no information was present in the data set.
The Author node represents a person who has written or edited a publication; if given, the author node also contains the ORCID, a unique identifier for researchers.
The other node types are, in quick order, as follows:

- Publisher: the entity who releases the publications(e.g. MIT Press);
- Journal: the scientific journal where papers and other publications are published;
- School: the university in which a publication has been released;
- Series: a collection of publications.

These node types have their names as their only attibute.
The last node type is the *"link"* node type, which contains an URL for accessing the publication online information (e.g. the publication's DOI).

5.2. Links

Almost every relation is uni-directional and from a publication towards another node.

5.2.1. Written by, Edited by

The ***WRITTEN_BY*** relation links a publication to its author.
Similarly, the ***EDITED_BY*** relation links a publication to the person who reviewed the publication.

5.2.2. Published by

The ***PUBLISHED_BY*** relation links a publication to its publisher.

5.2.3. Published on

The ***PUBLISHED_ON*** relation links the publication (usually an article) to the scientific journal it was published on.

5.2.4. For School

The ***FOR_SCHOOL*** relation links a publication node (PhD or Master's thesis) to a School node.

5.2.5. Part of

The ***PART_OF*** relation links a publication to the series it belongs, if any.

5.2.6. Link

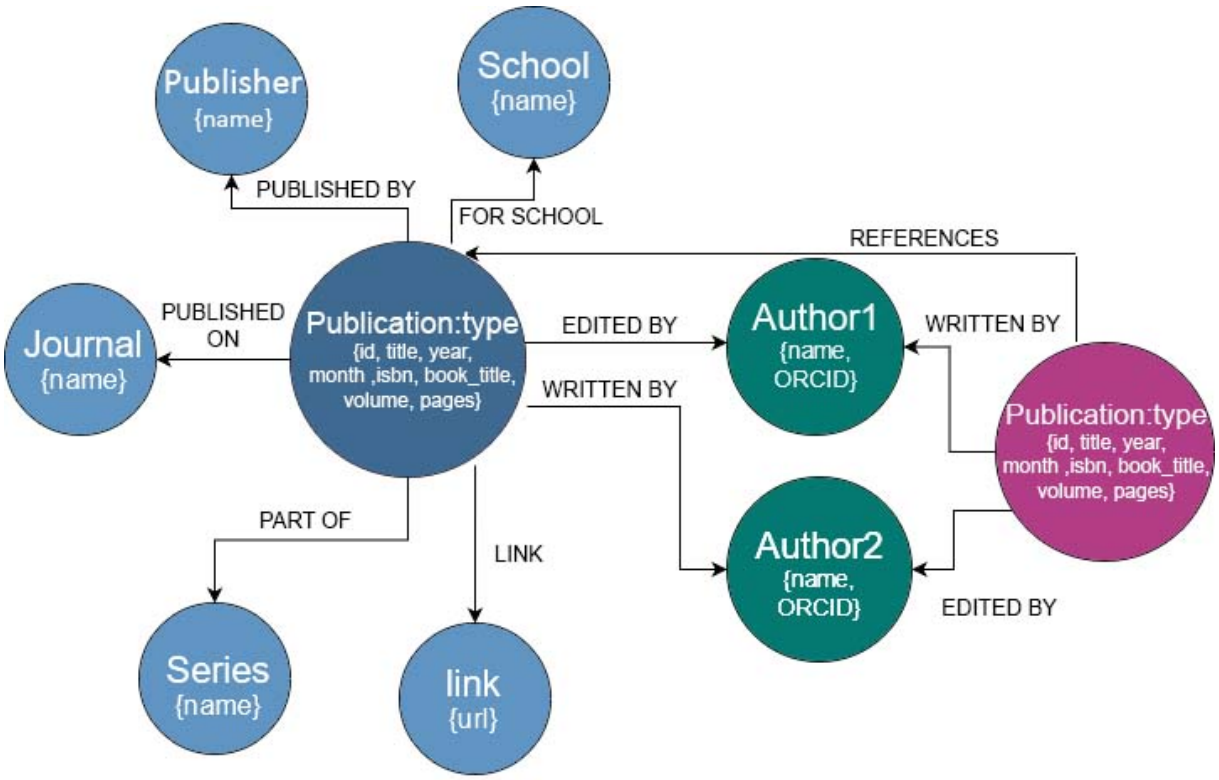
The ***LINK*** relation links a publication to all its relevant URLs, such as its DOI or a link to download its pdf. Links are nodes related to a publication instead of fields because the we observed that the number of links for a given publication is variable.

5.2.7. References

The ***REFERENCES*** relation links a publication node to another publication node that indicates that the first node is part of the second publication, for example it links a chapter of a book to the corresponding book.

5.3. Picture of the diagram

The following image is a schematic representation of the resulting graph database, as one may notice, all the relations sprout from a Publication node toward the other node types. The *":type"* label in the Publication node means that a Publication node has a second label which indicates the particular type of publication it is (e.g. an article, a book, a thesis...).[1]



[1]The only exception to this is when a publication references a publication which does not exist in the db at that moment in time; in such a case, a *placeholder* node is added which is just a publication and only contains the reference id (if and when a Publication with such an id is added to the db at a different point in time, the placeholder will be filled with all the given information, including the particular *type* label).

5.4. Queries

5.4.1. Creation commands

Creation of a new article for a given author

```
MERGE (a:Author{name:'TestAuthor'})
CREATE (a)<-[:WRITTEN_BY]-(ar:Publication:article{title:'TestArticle', year
    :'2022'})
RETURN a, ar
```

This query creates a new publication written by an author. Since **MERGE** was used, if the Author is not found in the DB, it is created, together with the new article and the WRITTEN_BY relationship. If **MATCH** was used, the query execution would have stopped in case the Author was not found in the DB.

Figure 5.1: Creation of a new publication for a given author

Creation of the relationship between an article and the journal

```
MATCH (ar: article { title :' TestArticle '})-[:WRITTEN_BY]->(a:Author)
MERGE (ar)-[:PUBLISHED_ON]->(j:Journal{name:'Digital System Research
    Center Report'})
RETURN a, ar, j
```

Once the article has been created, we create the relationship to the journal in which it has been published. **MERGE** is used instead of **CREATE**, to avoid creating the relationship if it is already present in the DB.

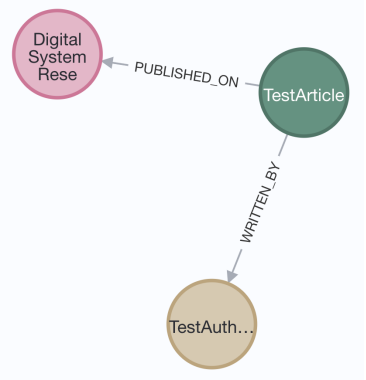


Figure 5.2: Creation of the new relationship from an article to its journal

Creation of a PhD thesis of the author and the relative school

```
MATCH (ar: article { title :' TestArticle '})-[:WRITTEN_BY]->(a:Author)
MERGE (s:School{name:'Politecnico di Milano'})<-[:FOR_SCHOOL]-(t:
    Publication:phdthesis{title:'A test PhD thesis'})-[:WRITTEN_BY]->(a)
RETURN a,ar,t,s
```

With this creation command, we want to create the PhD thesis of the author of the article we previously created. In addition, we also create the school associated with the PhD, if not already present in the DB.

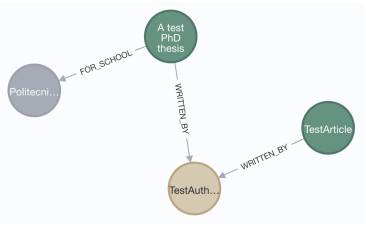


Figure 5.3: Creation of the PhD thesis of the author of the article previously created

Creation of a conference proceeding and the publications contained in the proceeding

```
MERGE (p:Publication:proceedings{ title : "Proceeding of a Conference"})
CREATE
(p)<-[:REFERENCES]-(p1:Publication:inproceedings{title: "Paper 1"}),
(p)<-[:REFERENCES]-(p2:Publication:inproceedings{title: "Paper 2"}),
(p)<-[:REFERENCES]-(p3:Publication:inproceedings{title: "Paper 3"})

return p, p1, p2, p3
```

The query creates both the proceeding of a conference and the papers presented at the conference and therefore included in the proceedings.

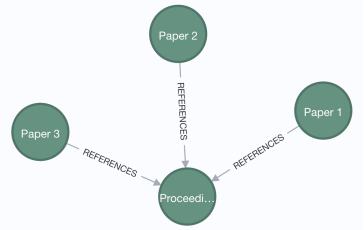


Figure 5.4: Creation of a conference proceeding and the published papers

Creation of the series the new proceeding is part of

```
MERGE(s:Series{name:'A new Series'})
MERGE(s)<-[:PART_OF]-(p:Publication:proceedings{title:'Proceeding of a
    Conference'})
return p, s
```

This query creates the new series, while also creating the link to the proceeding just added to the DB. If the relationship, or just the series, already exist, the **MERGE** command avoids the creation of duplicates.

Figure 5.5: Creation of a series for the new proceeding

5.4.2. Search queries

Authors who have at least two publications no more than two years apart

```
MATCH(p1:Publication)−[:WRITTEN_BY]−>(a:Author)<−[:WRITTEN_BY]−(
    p2:Publication)
WHERE abs(toInteger(p1.year) − toInteger(p2.year)) < 2 AND ID(p1) < ID(p2)
RETURN DISTINCT a, p1, p2
```

This query tries to match two different publications for each author, and then returns only the authors who have at least two publications written in less than two years. The condition `ID(p1) < ID(p2)` let us consider each pair of publications only once.

For clarity, only a partial output of the query is shown below.

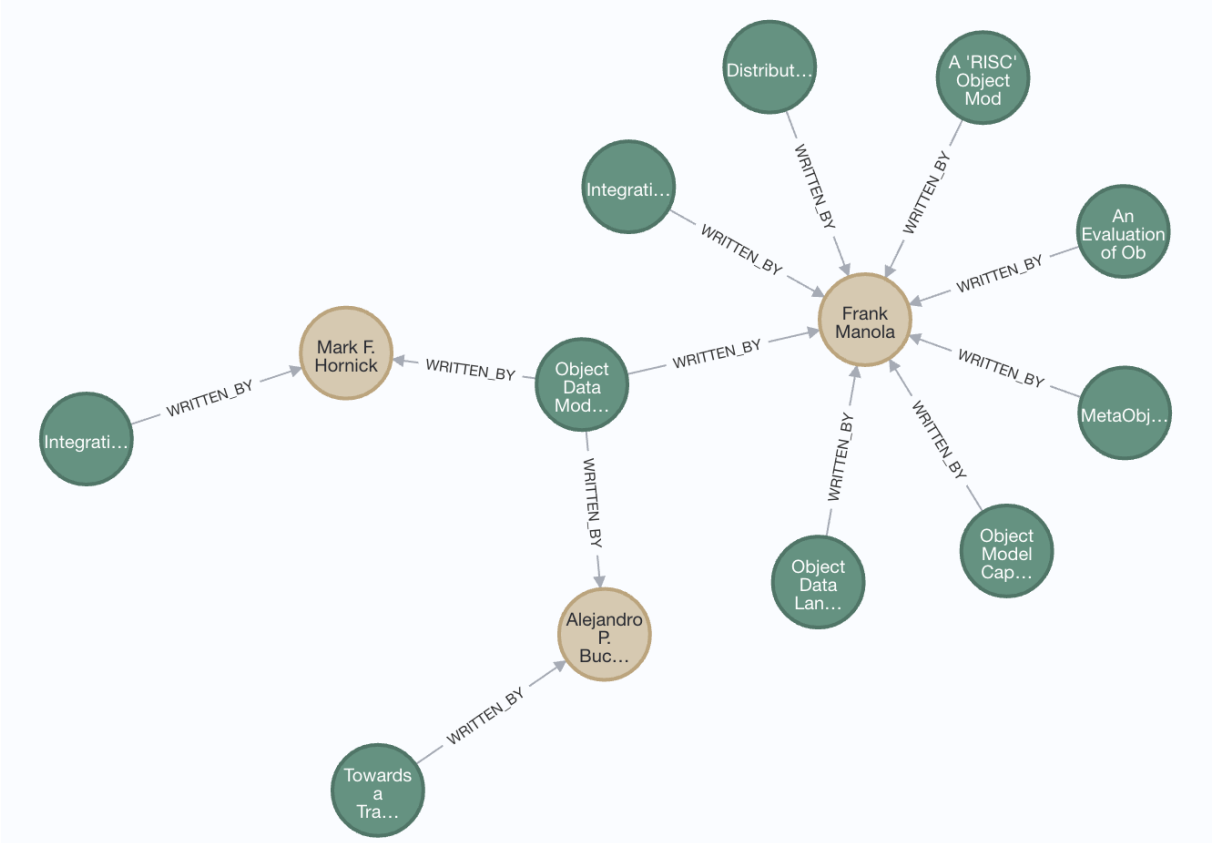


Figure 5.6: Authors with at least two publications in two years

The data related to the profiling of the query can be found in the following table:

Started Streaming (ms)	Completed Streaming (ms)	Memory (bytes)	pagecache hits	pagecache misses	Estimated rows	db hits
2	78	128664	69126	0	668	231010

Top 10 pairs of authors by common publications

```
MATCH (a1:Author)<-[r1:WRITTEN_BY]-(p:Publication)-[r2:WRITTEN_BY
    ]->(a2:Author)
WITH COUNT(p) AS commonPapersCount, a1, a2
WHERE ID(a1) < ID(a2)
RETURN a1.name AS Author1, a2.name AS Author2, commonPapersCount ORDER BY
     commonPapersCount DESC LIMIT 10
```

With this query, we are searching for the ten pairs of authors with the highest number of common publication (i.e., the pairs of authors who are coauthors more often in the database). In this case, our approach of using multiple labels to represent the publication generalization is useful since we can consider all the types of publications of which the authors might be coauthors. The condition `ID(a1) < ID(a2)` is used to consider each pair only once (e.g. we consider only ["Author1", "Author2"] and not also ["Author2", "Author1]).

"Author1"	"Author2"	"commonPapersCount"
"Darrel Hankerson"	"Alfred Menezes"	22
"Christoph Meinel"	"Anna Slobodová"	13
"Rainer Tichatschke"	"Alexander Kaplan"	11
"Christoph Meinel"	"Harald Sack"	11
"Peter Gritzmann"	"Victor Klee"	10
"Christoph Beierle"	"Udo Pletat"	9
"Nguyen V. Thoai"	"Reiner Horst"	7
"Christoph Meinel"	"Jochen Bern"	6

Figure 5.7: Top 10 pairs of authors by common papers count

The data related to the profiling of the query can be found in the following table:

Started Streaming (ms)	Completed Streaming (ms)	Memory (bytes)	pagecache hits	pagecache misses	Estimated rows	db hits
32	58	105040	N/A	N/A	10	46282

Publications published in proceedings with an author with at least other 5 publications

```
MATCH(paper:Publication:inproceedings)−[:WRITTEN_BY]−>(a:Author)<−[w:
    WRITTEN_BY]−(:Publication)
WITH paper, a, COUNT(w) AS numberOfWrittenPapers
WHERE numberOfWrittenPapers >= 5
RETURN DISTINCT paper
```

This query returns the list of papers presented in a conference (and therefore included in a proceeding), which have at least an author who has written at least other 5 publications.


```
╒══════════════════════════════════════════════════════════════════════════════╕
│"paper"                                                                       │
╞══════════════════════════════════════════════════════════════════════════════╡
│{"pages":"179-200","month":"January","year":"1974","id":"persons/Codd7│
│4","booktitle":"IFIP Working Conference Data Base Management","title":│
│"Seven Steps to Rendezvous with the Casual User."}                            │
├──────────────────────────────────────────────────────────────────────────────┤
│{"pages":"402-427","year":"1991","id":"journals/lncs/BollingerLP91","t│
│itle":"The LILOG Inference Engine.","booktitle":"Text Understanding in│
│ LILOG"}                                                                      │
├──────────────────────────────────────────────────────────────────────────────┤
│{"pages":"55-62","year":"1991","id":"journals/lncs/DorreR91","title":"│
│The STUF Workbench.","booktitle":"Text Understanding in LILOG"}               │
├──────────────────────────────────────────────────────────────────────────────┤
│{"pages":"39-50","year":"1991","id":"journals/lncs/Dorre91","title":"T│
│he Language STUF.","booktitle":"Text Understanding in LILOG"}                 │
└──────────────────────────────────────────────────────────────────────────────┘
```


Figure 5.8: Proceedings papers with an author with at least other 5 publications

The data related to the profiling of the query can be found in the following table:

Started Streaming (ms)	Completed Streaming (ms)	Memory (bytes)	pagecache hits	pagecache misses	Estimated rows	db hits
2	6	5792	2	0	0	248

Co-authors and related publications of a certain author

```
MATCH (a1:Author{name:'Paul Kocher'})<-[:WRITTEN_BY|EDITED_BY]-(p:
    Publication)-[:WRITTEN_BY|EDITED_BY]->(a2:Author)
RETURN a2 as coAuthor, collect(p) as papers
```

This query returns the list of co-authors (or co-editors) of a certain author, and the `collect` aggregate function is used to also return all the publications that the two authors have written or edited together.

Figure 5.9: Co-authors and co-editors of a certain author, with all the co-authored publications

The data related to the profiling of the query can be found in the following table:

Started Streaming (ms)	Completed Streaming (ms)	Memory (bytes)	pagecache hits	pagecache misses	Estimated rows	db hits
30	38	3448	37	0	18	8450

Authors who have written a thesis and also published an article on an academic journal

```
MATCH(school:School)<-[:FOR_SCHOOL]-(thesis:Publication)-[:
    WRITTEN_BY]->(a:Author)<-[:WRITTEN_BY]-(art:article)-[:
    PUBLISHED_ON]->(j:Journal)
RETURN school, a, thesis, art, j
```

This query considers those authors who have written both a thesis and an article published on a Journal, and also return their school, thesis, article and Journal.

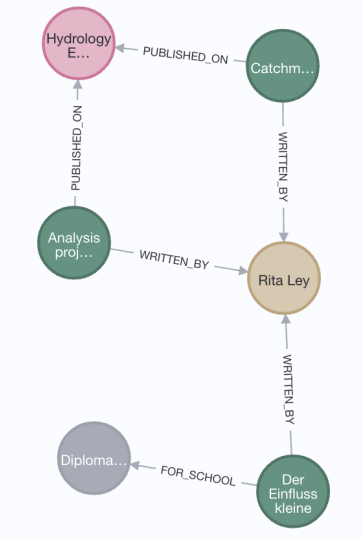


Figure 5.10: School, thesis, article, and journal of authors who have written both a thesis and a journal article

The data related to the profiling of the query can be found in the following table:

Started Streaming (ms)	Completed Streaming (ms)	Memory (bytes)	pagecache hits	pagecache misses	Estimated rows	db hits
25	26	200	N/A	N/A	9	307

Top 10 publishers by number of authors

```
MATCH (publisher:Publisher)<-[:PUBLISHED_BY]-(p:Publication)<-[:
    REFERENCES*0..1]-(p2:Publication)-[:WRITTEN_BY|EDITED_BY]->(
    a:Author)
RETURN publisher.name AS publisher, COUNT(DISTINCT a) as
    number_of_authors
ORDER BY number_of_authors DESC
LIMIT 10
```

The query returns the ten publishers which have, among their publications, the most number of distinct authors. To achieve this, path-length pattern matching has been used,

because publisher can publish both a collection of papers (proceeding, book) but also a single article. Therefore, without `*0..1` the result would not contain the single articles that have a publisher but only those that are contained in a "collection" of research works.

For clarity, the first five rows of the result are shown.

	publisher	number_of_authors
1	"Springer"	995
2	"John Wiley & Sons, Inc."	535
3	"IOS Press"	174
4	"IGI Global"	146
5	"IBM Germany Science Center, Institute for Knowledge Based Systems"	133

Figure 5.11: Top ten publishers by number of authors

The data related to the profiling of the query can be found in the following table:

Started Streaming (ms)	Completed Streaming (ms)	Memory (bytes)	pagecache hits	pagecache misses	Estimated rows	db hits
40	55	73192	N/A	N/A	10	22813

Given two publishers, return all the authors which have written at least a paper for each of the two publishers

```
MATCH(p:Publisher{name:"Springer"})<−[:PUBLISHED_BY]−(:Publication)<−[:
    REFERENCES∗0..1]−(:Publication)−[:WRITTEN_BY|EDITED_BY]−>(a:
    Author)<−[:WRITTEN_BY|EDITED_BY]−(:Publication)−[:
    REFERENCES∗0..1]−>(:Publication)−[:PUBLISHED_BY]−>(:Publisher{
    name:'John Wiley & Sons, Inc.'})
RETURN DISTINCT(a.name) as author
```

This query returns all the authors who have written at least a paper contained in a collection published by "Springer", and have written at least a paper contained in a collection published by "John Wiley & Sons, Inc.". The query was written by making use of a path-length pattern match (for the same reason of the previous query), and its result is simply a list of the returned authors.

	author
1	"Nadia Magnenat-Thalmann"
2	"Daniel Thalmann"
3	"Sushil Jajodia"
4	"Sabrina De Capitani di Vimercati"
5	"Pierangela Samarati"
6	"Tor Helleseth"

Figure 5.12: Given two publishers, return all the authors which have written at least a paper for each of the two publishers

The data related to the profiling of the query can be found in the following table:

Started Streaming (ms)	Completed Streaming (ms)	Memory (bytes)	pagecache hits	pagecache misses	Estimated rows	db hits
138	144	1616	N/A	N/A	0	10026

Shortest path between two journals

```
MATCH s=shortestPath((a1:Journal)−[∗1..4]−(a2:Journal)) WHERE a1<>a2 RETURN s
```

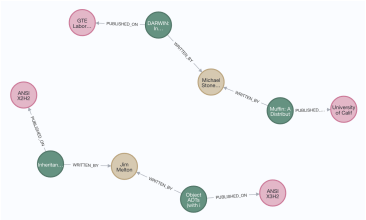


Figure 5.13: Shortest path between two journals

Started Streaming (ms)	Completed Streaming (ms)	Memory (bytes)	pagecache hits	pagecache misses	Estimated rows	db hits
1	28	62864	218	0	292	700

Shortest path between two schools

```
MATCH s=shortestPath((s1:School)−[∗1..10]−(s2:School)) WHERE s1<>s2 RETURN s
```

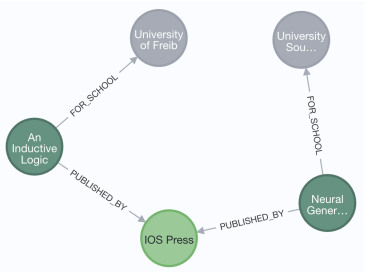


Figure 5.14: Shortest path between two schools

Started Streaming (ms)	Completed Streaming (ms)	Memory (bytes)	pagecache hits	pagecache misses	Estimated rows	db hits
1	8	220616	24	0	325	430

For each pair of publishers, return the number of common authors

```
MATCH (p1:Publisher)<−[:PUBLISHED_BY]−(:Publication)<−[:REFERENCES
    ∗0..1]−(:Publication) −[:WRITTEN_BY|:EDITED_BY]−>(a:Author)<−[:
    WRITTEN_BY|:EDITED_BY]−(:Publication)−[:REFERENCES∗0..1]−>(r:
    Publication)−[:PUBLISHED_BY]−>(p2:Publisher)
WHERE ID(p1) > ID(p2)
RETURN DISTINCT p1.name AS publisher, p2.name AS second_publisher, COUNT(
    DISTINCT a) AS common_authors ORDER BY common_authors DESC
```

As before, the path-length pattern match is necessary to also include single articles that have a publisher, while the condition `ID(p1) > ID(p2)` is needed to consider each pair of

publishers only once. The pairs of publishers in the result are then ordered in descending order, by the count of common authors.

publisher	second_publisher	common_authors
"IBM Deutschland GmbH"	"IBM Germany Science Center, Institute for Knowledge Based Systems"	26
"John Wiley & Sons, Inc."	"Springer"	14
"John Wiley & Sons, Inc."	"CRC Press"	2
"Springer"	"IBM Germany Science Center, Institute for Knowledge Based Systems"	1
"IBM Deutschland GmbH"	"Springer"	1
"IOS Press"	"IBM Germany Science Center, Institute for Knowledge Based Systems"	1

Figure 5.15: For each pair of publishers, return the number of common authors

The data related to the profiling of the query can be found in the following table:

Started Streaming (ms)	Completed Streaming (ms)	Memory (bytes)	pagecache hits	pagecache misses	Estimated rows	db hits
89	115	1479480	N/A	N/A	4	42058

The author who have written the most articles for a given journal

```
MATCH(j:Journal)<−[:PUBLISHED_ON]−(art:article)−[w:WRITTEN_BY]−>(a:
    Author) WHERE j.name='IWBS Report'
RETURN a.name, COUNT(w) AS number_of_articles ORDER BY number_of_articles
    DESC LIMIT 1
```

The name of the author who have written most articles for a given journal (in this case "IWBS Report") is returned. The data related to the profiling of the query can be found in the table below.

	a.name	number_of_articles
1	"Christoph Beierle"	13

Figure 5.16: The author who have written the most articles for a given journal

Started Streaming (ms)	Completed Streaming (ms)	Memory (bytes)	pagecache hits	pagecache misses	Estimated rows	db hits
8	9	22832	N/A	N/A	1	1798

Links associated with articles which have been published on journals which also contain articles published by former students

```
MATCH(l:link)<-[:LINK]-(p:article)-[:PUBLISHED_ON]->(j:Journal)<-[:
    PUBLISHED_ON]-(:article)-[:WRITTEN_BY]->(:Author)<-[:
    WRITTEN_BY]->(t)
WHERE t:phdthesis OR t:mastersthesis
RETURN DISTINCT(l.url) AS URL
```

This query first matches all the URLs associated with articles which have been published on a journal, then it restricts the scope to the journals which also contain publications written by authors who have also written a PhD or Master's thesis.

	URL
1	"https://doi.org/10.5194/hess-15-2947-2011"
2	"https://doi.org/10.5194/hess-16-409-2012"

Figure 5.17: Links associated with articles which have been published on journals which also contain articles published by former students

The data related to the profiling of the query can be found in the following table:

Started Streaming (ms)	Completed Streaming (ms)	Memory (bytes)	pagecache hits	pagecache misses	Estimated rows	db hits
10	12	1720	N/A	N/A	3	249

6 | MongoDB

6.1. Document Structure

In our MongoDB implementation of the bibliography DB, we have one collection (`papers`) which contains all the papers in the database. Obviously, not all the attributes are mandatory, and some of them are optional and not included in all papers.
Design choices for the structure of the document were application-driven (depending on our data queries). For example, we decided to use embedded documents for authors, and not references. In this way, when a publication is returned by a query, it also contains all its authors, and we don't need to perform further operations (joins) to obtain all the needed information. Instead, when implementing citations (papers cited by other papers), we decided to use manual references, since each paper is a large document, therefore duplicating it would make the size of the results much larger than it should in many cases, especially if the citations are not needed for a certain query.
The structure of each document representing a scientific paper is shown below.

```
{
  "title": String,
  "journal": String,
  "ee": String,
  "publisher": String,
  "year": Integer,
  "month": String,
  "volume": String,
  "pages": String,
  "booktitle": String,
  "abstract": String,
  "documentBody": [
    {
      "chapter": {
        "title": String,
```

```
        "sections":[
          {
            "section":{
              "title": String,
              "paragraphs":[
                {
                    "text": String
                },
                ...
              ],
              "figures": [
                {
                  "url": String,
                  "caption": String
                },
                ...
              ]
            }
          }
        ]
      },
      ...
    }],
    "keywords": [String],
    "language": String,
    "authors":[
      {
        "name": String,
        "email: String,
        "affiliation": String,
        "bio": String,
        "birth_year": Integer
      },
      ...
    ],
    "citations": [ObjectId]
  ]}
```

6.2. Data Upload and Transformation

6.2.1. Data pre-processing

To implement the MongoDB collection of scientific papers, we started from the XML file generated as the output of the pre-processing of Delivery 1. We decided to also add additional attributes, that were not present in the previous dataset, generating them randomly.
The first phase of the pre-processing consists of extracting papers and the attributes of interest. The following attributes for each publication are extracted (when present) from the XML file:

- Title
- Journal
- Publisher
- Year
- Month
- Volume
- Pages
- Booktitle
- Authors
- ee (URL)
- Type (article, inproceedings,...)

In the next phase we must enrich the dataset with the data related to each paper. To achieve this, we created a sample text file `text.txt`; in this way, we can randomly extract both a group sentences or a single word. For each publication, the pre-processing proceeds as follows:

1. We generate a random abstract by extracting random sentences from the sample text file `text.txt`
2. We generate keywords from the sample text file `text.txt` as random words
3. We generate a set of cited papers: we take a variable number of random papers and we create an array of titles of cited papers (this will be used then in the upload

process to create manual references)

4. Each paper has a 25% chance of having a language attribute, which is randomly extracted from the file `languages.txt`
5. We need to generate a random ***documentBody*** for the paper:
   - Each paper has a variable number of chapters
   - Each chapter has a variable number of sections
   - Each section includes a title, and a variable number of paragraphs and figures. A paragraph is simply a set of random sentences from the text file, while a figure has a random URL and caption.
6. For each author, we create a subdocument that contains:
   - Name
   - Email (generated as name@mail.com)
   - Bio (randomly generated)
   - Affiliation (picked randomly from a set of universities in the file `universities.txt`)
   - Birth year (randomly generated)

The final output of the pre-processing is a JSON file containing a JSON array of documents representing publications.

6.2.2. Data upload

After the pre-processing, we created a Python script to create the collection and insert all the generated documents. To handle the citations as an array of manual references, we create an association between paper titles and ObjectIds:

```
papers_id = {}
data = json.load(open('mongodb-documents.json'))

for paper in data:
  id = ObjectId(uuid.uuid4().hex[:24])
  papers_id[paper["title"]] = id
```

Then, we reuse the populated dictionary to create documents and references between them in the `citations` attribute.

```
for paper in data:
  paper["_id"] = papers_id[paper["title"]]
  citations = []
  for cited_paper in paper["citations"]:
    citations.append(papers_id[cited_paper])
  paper["citations"] = citations
  papersCollection.insert_one(paper)
```

6.3. Queries

6.3.1. Data creation and update commands

a) Inserting a new paper

```
db.papers.insertOne({
  authors: [
    {
      name: "Author",
      email: "author@mail.com",
      bio: "Sample Bio",
       affiliations : "University",
      birth_year: 1980
    }
  ],
  title : "Title",
  year: 2022,
  type: "article",
  abstract: "Sample abstract",
  documentBody: [
  {
    chapter: {
       title : "Chapter title",
      sections: [
        { section: {
             title : "Section title",
            paragraphs: [
              {
                text: "Sample text"
              }
            ],
            figures : [
              {
                url: "url",
                caption: "caption"
              }
            ]
          }
        },
      ]
    }
  }],
```

```
  citations :  [],
  keywords: [ "keyword" ]
})
```

Results:

```
{
  acknowledged: true,
  insertedId: ObjectId("637ba64b2dd099d0e47dfb83")
}
```

b) Update authors of a paper

```
db.papers.updateOne(
    { _id: ObjectId("637ba64b2dd099d0e47dfb83") },
    { $push: { authors: {
      name: "Author2",
      email: "author2@mail.com",
      bio: "Sample Bio 2",
       affiliations : "University",
      birth_year: 1985
    } } }
)
```

Results:

```
{
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
}
```

c) Update the bio of a given author

```
db.papers.update({"authors.name": 'E. F. Codd'},{"$set": {"authors.$.bio":" hello
    world!"}}, {"multi": true})
```

Results:

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

d) Remove all articles from a given journal

```
db.papers.remove({journal: 'meltdownattack.com'})
```

Results:

```
{ acknowledged: true, deletedCount: 2 }
```

e) Remove all authors from a certain university

```
db.papers.updateMany(
                {"authors":
                    {$elemMatch: {affiliations: 'Politecnico di Torino'}}},
                    {$pull: {authors: { affiliations : 'Politecnico di Torino'}}}
                        )
```

Results:

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 60,
  modifiedCount: 60,
  upsertedCount: 0
}
```

6.3.2. Data queries

f) All papers with more than five authors

```
db.papers.aggregate([
  { "$project":
    { "title": 1,
      "numAuthors": { $cond:{if:{$isArray:"$authors"}, then: {$size: "$authors
    "}, else:"1"}}}
  },
  { "$match":{
    "numAuthors": {"$gt": 5}
    }
  }
])
```

Result of the query (first 3 papers[1]):

```
[
  {
    _id: ObjectId("d1429d44a2594d0388662d28"),
    title: 'Spectre Attacks: Exploiting Speculative
   Execution.',
    numAuthors: 10
  },
  {
    _id: ObjectId("7c5df6e4fd3b4e019ba561f5"),
```

[1]The results of the query are partial to improve readability where indicated

```
    title: 'Meltdown',
    numAuthors: 10
  },
  {
    _id: ObjectId("241f8b284e98446889fc40fe"),
    title: 'LILOG - Linguistische und logische Methoden
   fr das maschinelle Verstehen des Deutschen -
   Projektbeschreibung',
    numAuthors: 11
  }
]
```

g) Papers with at least one author born between 1950 and 1960 who has a specific affiliation (MIT)

```
db.papers.find(
  {"authors":
    {$elemMatch:
      {
           " affiliations ": "MIT",
           $and: [ {"birth_year": {$gt: 1950}}, {"birth_year": {$lt: 1960}}]
      }
    }
  },
    {
        " title ":1, "authors.name":1, "authors. affiliations ": 1, "authors.
    birth_year": 1
    }
)
```

Result of the query (first 3 papers):

```
[
  {
    _id: ObjectId("58a56dd415ff42ef873d31fe"),
    authors: [
      { name: 'Ipke Wachsmuth', affiliations: 'MIT',
```

```
 birth_year: 1952 }
  ],
  title: 'Zur intelligenten Organisation von
 Wissensbestnden in knstlichen Systemen'
},
{
  _id: ObjectId("7bd4d99d6a394b468c4fae4f"),
  authors: [ { name: 'Rolf Mayer', affiliations: 'MIT',
  birth_year: 1956 } ],
  title: 'Domain Restriction and Other Kinds of
 Reference Set Operations in Sentence and Discourse
 Semantics.'
},
{
  _id: ObjectId("51a1728132024095bd27f4e3"),
  authors: [
    {
      name: 'Claus-Rainer Rollinger',
      affiliations: 'Politecnico di Torino',
      birth_year: 1955
    },
    {
      name: 'Rudi Studer',
      affiliations: 'Technical University of Munich',
      birth_year: 1952
    },
    {
      name: 'Hans Uszkoreit',
      affiliations: 'ETH Zurich',
      birth_year: 1968
    },
    { name: 'Ipke Wachsmuth', affiliations: 'MIT',
 birth_year: 1952 }
  ],
  title: 'Textunderstanding in LILOG - Sorts and
 Reference Objects'
}]
```

h) Authors of at least two papers sorted in descending order

```
db.papers.aggregate([
    { "$unwind":
        { "path": "$authors" } },
        { "$group": { "_id": "$authors.name", "count": { "$sum": 1 } } },
        { "$match": { "count": { "$gte": 2 } } },
        { "$sort" : { count : -1 } },
        { "$limit" : 10 }
])
```

Result of the query (limited to 10 authors):

```
[
  { _id: 'Christoph Beierle', count: 18 },
  { _id: 'Udo Pletat', count: 11 },
  { _id: 'Thomas Ludwig 0001', count: 9 },
  { _id: 'Peter H. Schmitt', count: 8 },
  { _id: 'Frank Manola', count: 8 },
  { _id: 'Claus-Rainer Rollinger', count: 6 },
  { _id: 'Egon Brger', count: 6 },
  { _id: 'Rudi Studer', count: 6 },
  { _id: 'Stefan Bttcher', count: 6 },
  { _id: 'Gert Smolka', count: 5 }
]
```

i) Number of total chapters of papers published after 1999

```
db.papers.aggregate([
        {"$match":
                {"year":{"$gte":2000}}
        },
        {"$unwind":
                {"path":"$documentBody"}
        },
        {"$group":
                {"_id":"chapter", "count":{"$sum":1}}
```

```
        }
])
```

Result of the query:

```
[ { _id: 'chapter', count: 1653 } ]
```

j) Name of the author with the maximum number of publications

```
db.papers.aggregate([
        { "$unwind":
                { "path": "$authors" }
        },
        { "$group":
                { "_id": "$authors.name", "count": { "$sum": 1} } },
        { "$sort ":
                {"count": -1 }
        },
        {
            "$limit ": 1
        }
])
```

Result of the query:

```
[ { _id: 'Christoph Beierle', count: 18 } ]
```

k) Three most prolific authors

```
db.papers.aggregate([
        {$unwind :
            "$authors"
        },
        {$group:
            { _id: "$authors", count: {$sum: 1}}
        },
```

```
        {
            $sort: {count : -1}
        },
        {
            $limit: 3
        }
])
```

Result of the query:


```
[
  {
    _id: {
      name: 'Christoph Beierle',
      email: 'christoph_beierle@mail.com',
      bio: ' Nam posuere, purus nec mollis tristique,
   ipsum ante pretium velit, non sagittis orci sem a
   ligula',
      affiliations: 'Politecnico di Torino',
      birth_year: 1954
    },
    count: 18
  },
  {
    _id: {
      name: 'Udo Pletat',
      email: 'udo_pletat@mail.com',
      bio: ' Donec suscipit luctus lorem, ac ultrices est
   tincidunt facilisis',
      affiliations: 'Politecnico di Milano',
      birth_year: 1953
    },
    count: 11
  },
  {
    _id: {
      name: 'Thomas Ludwig 0001',
```

```
      email: 'thomas_ludwig_0001@mail.com',
      bio: ' Morbi fermentum at nulla et imperdiet',
      affiliations: 'Politecnico di Torino',
      birth_year: 1963
    },
    count: 9
  }
]
```

l) Email of the three most prolific authors of a given journal (IWBS Report)

```
db.papers.aggregate([
  {
    $match: {
    journal: "IWBS Report"
    }
  },
  {
    $unwind: "$authors"
  },
  {
    $group: {
    _id : "$authors.name",
    count: {$sum : 1},
    email: { $first : "$authors.email"}
    }
  },
  {
    $sort: { count : -1}
  },
  {
    $limit : 3
  }
])
```

Result of the query:

```
[
  {
    _id: 'Christoph Beierle',
    count: 11,
    email: 'christoph_beierle@mail.com'
  },
  { _id: 'Egon Brger', count: 6, email: 'egon_brger@mail.
   com' },
  {
    _id: 'Stefan Bttcher',
    count: 6,
    email: 'stefan_bttcher@mail.com'
  }
]
```

m) Title and authors of papers published on a journal

```
db.papers.find(
        {
            "journal":{ $exists : true}
        },
        {
            " title ":1,  "authors.name":1
        }
)
```

Result of the query (limited to three papers):

```
[
  {
    _id: ObjectId("cf61e4d221e64a3981e7b3e8"),
    authors: [ { name: 'Phil Shaw' } ],
    title: 'Modification of User Defined Types'
  },
  {
    _id: ObjectId("397cf90f9cb3408f8b375c67"),
```

```
    authors: [
      { name: 'Jim Melton' },
      { name: 'Jonathan Bauer' },
      { name: 'Krishna G. Kulkarni' }
    ],
    title: 'Object ADTs (with improvements for Value ADTs
   )'
  },
  {
    _id: ObjectId("9fd452ef3ed1429783064488"),
    authors: [ { name: 'David Beech' } ],
    title: 'Unification of Value and Object ADTs'
  }
]
```

n) Titles containing given words ("Computer" and "Knowledge")

```
db.papers.find(
        {"$or":
            [{ " title ": /Computer/ },{ "title": /Knowledge/ }]
        },
        {
            " title ":1
        }
)
```

Result of the query (limited to three titles):

```
[
  {
    _id: ObjectId("9c77d02284d642ed8f00cbc1"),
    title: 'Using Knowledge-Based Methods to Administrate
    an Access Control System'
  },
  {
    _id: ObjectId("7e1f26131ee3427f892517eb"),
```

```
    title: 'An Order-Sorted Logic for Knowledge
   Representation Systems'
  },
  {
    _id: ObjectId("b303c2e512a048529ca187e3"),
    title: 'Erster EXCEPT-Workshop: Computeruntersttzte
   Umweltvertrglichkeitsprfung'
  }
]
```

o) Most used keywords

```
db.papers.aggregate([
  {
    $unwind : "$keywords"
  },
  {
    $group: {
   _id: "$keywords",
   count: {$sum: 1}
    }
  },
  {
    $sort: {count : -1}
  },
  {
    $limit: 5
  }
])
```

Result of the query:

```
[
  { _id: 'urna', count: 37 },
  { _id: 'mattis', count: 34 },
  { _id: 'lorem', count: 31 },
  { _id: 'a', count: 28 },
```

```
  { _id: 'dolor', count: 27 }
]
```

p) Year of the last publication for each type

```
db.papers.aggregate(
  [
    { $group:
              {
                _id:"$type",
                Result:{$max:"$year"}
              }
    }
  ]
)
```

Result of the query:

```
[ { _id: 'article', Result: 2022 }, { _id: 'book', Result
    : 2013 } ]
```

q) Publications ordered by authors and by language

```
db.papers.aggregate([
    {$unwind: {path: "$authors"}},
     {$group:
        {_id:
            {"author": "$authors",
            "lang": {$ifNull: ["$language", "unknown"]}},
        "count": {$sum: 1},
        "works": {$push: "$title"}
        }
    },
    {$group:
        {_id:"$_id.author",
        "work_by_lang":
            {$push:
```

```
                {
                    "lang": "$_id.lang",
                    "count": "$count",
                    "works": "$works"
                }
            }
        }
    }
])
```

Result of the query (limited to one author):

```
[
  {
    _id: {
      name: 'Petra Steffens',
      email: 'petra_steffens@mail.com',
      bio: ' Fusce nec tincidunt augue, nec suscipit
   velit',
      affiliations: 'Politecnico di Milano',
      birth_year: 1980
    },
    work_by_lang: [
      {
        lang: 'unknown',
        count: 1,
        works: [
          'LILOG - Linguistische und logische Methoden fr
    das maschinelle Verstehen des Deutschen -
   Projektbeschreibung'
        ]
      },
      {
        lang: 'french',
        count: 1,
        works: [
          'Zur Syntax prnominaler Elemente in einer
```

```
    kategorialen Unifikationsgrammatik des Deutschen'
        ]
      }
    ]
  }, ...
]
```

r) For all papers, find and count all the papers in which they are cited

```
db.papers.aggregate([
        { $project: { "_id": 1,
                      " title ": 1,
                     " citations ": 1
                     }
            },
        { $unwind:
            { path: " $citations " } },
        { $group:
            { _id: " $citations ",
            "count": { $sum: 1 },
            "works": { $push: " $title " }
              }
            }
        ])
```

Result of the query (first two results):

```
[
 {
    _id: ObjectId("b576ec2fa28141f9b073c0d5"),
    count: 6,
    works: [
      'Object Oriented DBMS as a Generalization of
   Relational DBMS',
      'Modification of User Defined Types',
      'On the Problem of Masking Special Errors by Serial
```

```
    Signature Analysis',
      'Die Behandlung von mehrdeutigen Verben in der
   Maschinellen bersetzung',
      'Using Knowledge-Based Methods to Administrate an
   Access Control System',
      'The Many-Valued Theorem Prover '
    ]
  },
  {
    _id: ObjectId("1a2436c063fd45688bc727d0"),
    count: 10,
    works: [
      'Die Reprsentation rumlichen Wissens und die
   Behandlung von Einbettungsproblemen mit
   Quadtreedepiktionen',
      'A Logical Operational Semantics of Full Prolog',
      'On the Interpretation of Equality, Sorts, and
   Logic Programming',
      'Ereignisse: Ihre Logik und Ontologie aus
   textsemantischer Sicht',
      'A Model Elimination Calculus for Generalized
   Clauses',
      'Tableau Calculus for Order Sorted Logic',
      'Strategien zur Pronominalisierung',
      'WebS - ein System zur Generierung von
   Wegbeschreibungen',
      'Mentale Bilder und Wegbedeutungen',
      'Textunderstanding in LILOG - Sorts and Reference
   Objects'
    ]
  }, ...
]
```

s) For a given article find all articles who refer to it

```
db.papers.find(
    {citations: {$in: [ObjectId("35a22842b8554a9ab4e8fbd0")]}},
    {_id:1, title: 1})
```

Result of the query (limited to three results):

```
[
  {
    _id: ObjectId("d1429d44a2594d0388662d28"),
    title: 'Spectre Attacks: Exploiting Speculative
   Execution.'
  },
  {
    _id: ObjectId("35a22842b8554a9ab4e8fbd0"),
    title: 'Object Model Capabilities For Distributed
   Object Management.'
  },
  {
    _id: ObjectId("3dd1f655bd4f422ea2c7f471"),
    title: 'Dokumentation der Syntax der LILOG-Grammatik'
  }, ...
]
```

t) For each journal, return the articles which cites the most other articles

```
db.papers.aggregate(
        {$project:
            {"journal": 1,
            "num_of_cit": {$size: "$citations"} , "title": 1
                }
            },
        {$sort:
            {"num_of_cit": -1}
            },
        {$group:
```

```
        {_id: "$journal",
        "best": { $first : "$$ROOT" }
            }
        },
    {$replaceRoot:
        {newRoot: "$best"}
        },
    {$sort: {"num_of_cit": -1}}
        )
```

Result of the query (first 3 results):


```
[
  {
    _id: ObjectId("63c4af25f94545779bf76a20"),
    title: 'KL-ONE: Eine Einfhrung',
    journal: 'IWBS Report',
    num_of_cit: 15
  },
  {
    _id: ObjectId("670e497969fc4dec8b3c0cab"),
    title: 'LILOG-DB: Database Support for Knowledge-
   Based Systems',
    journal: 'LILOG-Report',
    num_of_cit: 15
  },
  {
    _id: ObjectId("cf61e4d221e64a3981e7b3e8"),
    title: 'Modification of User Defined Types',
    journal: 'ANSI X3H2',
    num_of_cit: 15
  }
]
```


u) Papers with more than 19 chapters

```
db.papers.aggregate([
  {
  $match: {
    $expr: {
      $gt: [{$cond: {if: {$isArray: "$documentBody"}, then: {$size: "
    $documentBody"}, else:1}}, 19]
      }
    }
  },
  {
  $project: {
    " title ": 1,
    "documentBody.chapter.title": 1
    }
  }
])
```

Result of the query (first 2 results, first 3 chapters for each result):

```
  {
    _id: ObjectId("ad4e9ba0130a40f9890aaeab"),
    title: 'LILOG-DB: Database Support for Knowledge-
   Based Systems',
    documentBody: [
      { chapter: { title: ' Nam ut convallis quam' } },
      {
        chapter: { title: ' Duis interdum laoreet felis
   eget rutrum' }
      },
      {
        chapter: {
          title: '\n' +
            '\n' +
            'Curabitur lorem urna, ultrices vitae
   efficitur eu, varius eget risus'
        }
      },
```

[OTHER CHAPTERS...]

```
{
  _id: ObjectId("9ce8a7860f9443ef9bbf06f6"),
  title: 'Ein praktischer Algorithmus fr die E-
Unifikation',
  documentBody: [
    {
      chapter: {
        title: ' Mauris pretium commodo metus, vel
ultrices metus ultrices tempor'
      }
    },
    { chapter: { title: ' Aliquam convallis eleifend
hendrerit' } },
    {
      chapter: { title: ' Nunc ullamcorper eu enim
malesuada sollicitudin' }
    },
```

Query builder

The last commands isn't as much as a query, but more of a builder object, written in Python, able to generate queries, based on the given parameters.
The builder allows to filter the documents enabling certain research parameters, such as:

- minimum publication year;
- maximum publication year;
- title (instead of strictly looking for the given string, all items containing the given string as a sub-string in the title are returned);
- a list of authors' names
- the document's language
- the journal the paper was published on;
- a list of keywords

Beside the title field, each of these fields can have either an inclusive or an exclusive search mode.
By inclusive search we mean that an article will pass the filter if its given field contains any of the values passed to the builder while with an exclusive search only the articles whose field contains all the given values will pass (e.g. in an inclusive search by author, each article containing at least one of the given authors will be returned, while an exclusive search this will happen only if all of the given authors have participated to the article's writing) [2].
The code for the builder is as follows:

```
class MDBQueryBuilder:
    def ___init___(self):
        self . pipeline = []
        self .ranges = {"year": [None, None]}
        self .range_on = {"year": False}
        self . filters = {"title": [], "keywords":[], "authors.name":[],
 "language": [], "type":[], "journal": [] }
        self . filter_on= {"title": False, "keywords": False, "authors.
 name": False, "language": False, "type": False, "journal": False }
```

[2]exclusive search only makes sense for fields that contain arrays, such as authors and keywords; despite this, the builder allows for exclusive search for other fields but will be unable to return anything if exclusive search is enabled and more than one value is given

```
        self.filter_for_all = {"title": False, "keywords": False, "authors.
    name": False, "language": False, "type": False, "journal": False }


    def copy(self):
        ret_val = MDBQueryBuilder()
        ret_val.ranges = self.ranges.copy()
        ret_val.range_on = self.range_on.copy()
        ret_val.filters = self.filters.copy()
        ret_val.filter_on = self.filter_on.copy()
        ret_val.filter_for_all = self.filter_for_all.copy()
        return ret_val
    # toggles the filter for a particular numeric field
    def _toggle_range(self, field:str):
        ret_val = self.copy()
        ret_val.range_on[field] = not self.range_on[field]
        return ret_val
    # inserts a range for a field
    def _add_range(self, field:str, lower_bound= None, upper_bound = None):
        ret_val = self.copy()
        ret_val.ranges[field] = [lower_bound, upper_bound]
        return ret_val
    # toggles the filter for a given field (e.g. keywords)
    def _toggle_search_by_field(self, field:str):
        ret_val = self.copy()
        ret_val.filter_on[field] = not self.filter_on[field]
        return ret_val
    #  toggles exclusive search for a given field
    def _toggle_search_all_by_field(self, field:str):
        ret_val = self.copy()
        ret_val.filter_for_all[field] = not self.filter_for_all[field]
        return ret_val
    # insert a value to be search through the given filter
    def _add_elem(self, field:str, elem:str):
        ret_val = self.copy()
        ret_val.filters[field].append(elem)
        return ret_val

    # creates the query
    def get_query(self):
        print(self.ranges)
        print(self.range_on)
        filters_to_apply = [f'{{ "title":{{ $regex: \".*{self.filters["
    title"][0]}.*\" }}}}'] if self.filter_on["title"] is True else []
```

```
        filters_to_apply += [f'{{"{key}": {{{f"$gte: {val[0]}, " if val
    [0] is not None else "" }}{f"$lte: {val[1]}" if val[1] is not None
     else "" } }} }}' for key, val in self.ranges.items() if self.range_on[key]
    is True]
        filters_to_apply += [f'{{"{key}": {{ {"$all" if self.
    filter_for_all[key] is True else "$in"}: {val}}}}}' for key, val in
    self.filters.items() if self.filter_on[key] is True and key != "title"]
        pipeline = "db.papers.find({$and: [" + ",".join(filters_to_apply) +
    "]})"
        return (pipeline)
```

The code below shows how to use the builder through an example.

```
b = MDBQueryBuilder()
b = b\
    ._toggle_search_by_field("title")\
    ._add_elem("title",'down' )\
    ._toggle_search_by_field("authors.name")\
    ._toggle_search_all_by_field("authors.name")\
    ._add_elem("authors.name",'Paul Kocher')\
    ._add_elem("authors.name",'Daniel Genkin')\
    ._toggle_search_by_field("type")\
    ._add_elem("type", "article")\
    ._toggle_range("year")\
    ._add_range("year",1915, 2022)\
    ._toggle_search_by_field("journal")\
    ._add_elem("journal", 'meltdownattack.com')\
    .get_query()
print(b)
```

The result of the above-written command is as follows:

```
db.papers.find({$and: [
    { "title":{ $regex: ".*down.*" }},
    {"year": {$gte: 1915, $lte: 2022 } },
    {"authors.name":
        { $all: ['Paul Kocher', 'Daniel Genkin']}},
    {"type":
        { $in: ['article']}},
```

```
        {"journal":
            { $in: ['meltdownattack.com']}}
    ]})
```

Of course the use of this builder can be simplified for the end user through the use of a director object, or by just connecting all the given functions to a user interface (e.g. a button would allow to toggle exclusive search for certain fields, or a text field with a "add" button could be used to add an item to a specific field).

Benchmark

Benchmark has been revised after increasing the database size so that the comparison between different queries could be more significant.

query	avg execution time (milliseconds)
f	16
g	217
h	11
i	61
j	8
k	11
l	8
m	5
n	2
o	7
p	3
q	22
r	33
s	8
t	14
u	1162

7 | PySpark

7.1. Data Structure

Working with Pyspark's dataframes presents a challenge that was not addressed before: we are now working with a more restricted schema.
The dataframes do not allow for elastic structures like the ones adopted with the Neo4j and MongoDB databases, where null fields would be eliminated and custom fields added on the fly.
A more traditional, SQL-like approach was adopted instead. The dataframes behave like tables, with nullable columns to represent *"a priori"* defined properties, such as the title, publisher, or the type of publication. The relations linking different entities are dataframes as well.

7.1.1. The Model

The Model revolves around 2 entities and 3 relationships.

Entities

1. **Publications**
   The **"publications"** dataframe will contain all the relevant information about the dblp publications.
   A **publication** row has the following columns:
   (a) **title**: the title of the publication, acts as the primary key;
   (b) **type**: the type of publication, it represents whether the publication is an article rather than a book, an incollection, etc...;
   (c) **publisher**: the publisher that released the publication;
   (d) **journal**: scientific magazine where the publication was published;
   (e) **month**: month during which the publication was released, written as a string;

(f) **year**: year during which the document was released, saved in integer form;

(g) **language**: language in which the document was written;

(h) **booktitle**: present for the incollection rows, it is the title of the book from which the publication was extracted;

(i) **pages**: present for the incollection rows, it is the page range in which the publication may be found on the mentioned book (the one in the **booktitle** column);

(j) **ee**: electronic link through which one may access the article content.

2. **Authors**
   The **"authors"** dataframe will contain all the relevant information for any given author, namely:

   (a) **name**: name of the author, used as a primary key;

   (b) **birth_year**: year of birth of the author;

   (c) **email** : email of the author;

   (d) **affiliations**: for whom the author works, e.g. the university they work in;

   (e) **bio**: a short biographical description of the author.

Relations

1. **Publications to Authors**
   The **"pub2auth"** dataframe represents the relation between publications and authors.
   It is a many-to-many relationship, since a publication may have been written by any number of authors and, at the same time, an author may have written any number of publications in the past.
   The schema of the dataframe consists of two columns: **title** and **auth_name**.

2. **Citations**
   The **"citations"** relation is a many-to-many relationship, mapping any publication to all the other publications it mentions

3. **Keywords**
   Calling the **"keywords"** dataframe a many-to-many relationship would not be entirely correct: it is true that a publication may have any number of keywords and

that any keywords may be used in any number of publications; however, the keyword is just a string, it is not a foreign key for any other table.
Regardless, the **"keywords"** dataframe is used to determine the keywords (or tags) used in the publications.

```
<class 'pyspark.sql.dataframe.DataFrame'>
root
 |-- title: string (nullable = true)
 |-- type: string (nullable = true)
 |-- publisher: string (nullable = true)
 |-- journal: string (nullable = true)
 |-- month: string (nullable = true)
 |-- year: integer (nullable = true)
 |-- language: string (nullable = true)
 |-- booktitle: string (nullable = true)
 |-- volume: string (nullable = true)
 |-- pages: string (nullable = true)
 |-- ee: string (nullable = true)

<class 'pyspark.sql.dataframe.DataFrame'>
root
 |-- name: string (nullable = true)
 |-- birth_year: integer (nullable = true)
 |-- email: string (nullable = true)
 |-- affiliations: string (nullable = true)
 |-- bio: string (nullable = true)

<class 'pyspark.sql.dataframe.DataFrame'>
root
 |-- title: string (nullable = true)
 |-- auth_name: string (nullable = true)

<class 'pyspark.sql.dataframe.DataFrame'>
root
 |-- citing: string (nullable = true)
 |-- cited: string (nullable = true)

<class 'pyspark.sql.dataframe.DataFrame'>
root
 |-- title: string (nullable = true)
 |-- keyword: string (nullable = true)
```

Figure 7.1: Data-frames schemata

7.1.2. Reasoning behind a relational approach

When faced with the necessity of establishing relations, two main approaches come to mind: storing the relationship in a column of one of the tables (and later expanding it, if

needed, through the use of the "explode" method) or creating the aforementioned relation tables.
Both approaches have their merits; on the one hand nesting some relationships such as the authors in a column of the **publications** dataframe would have greatly simplified the pre-processing and perhaps could have improved the performance for some queries. Preferring the use of a relation table, however, allows improved performances in certain scenarios. For example, querying for all publications of a certain author using the lists approach would require either using the explode method to expand the dataframe and then filter for the author(essentially recreating the relation at run-time) or filtering through the content of the authors' lists through some PySpark UDF (which are relatively slow operations when custom made). The **pub2auth** table allows for a much quicker search by simply applying a filtering operation on itself and then using this result with a semi-join on the **publications** dataframe operation to quickly retrieve all the information on the relevant publications.

7.2. Pre-processing

The dataset used for this part of the project is the same as the one generated for the MongoDB delivery [1] but without the document body.
In order to convert the content of this JSON file into relational dataframes, it is necessary to store the information in a series of *".csv"* files.

7.2.1. Procedure

1. First, the content of the JSON file is loaded in a list of ordered dictionaries, each `OrderedDict` represents a publication (the content is the same as a document present in the MongoDB database except for the fact that there is no document body).

```
import json
import csv
import collections
import os

with open("mongodb-documents.json","r") as f:
    dic = json.load(f)
```

[1] See Section 6.2

2. All the publications contain a list of their authors (each author's entry contains all the information of the author), as well as a list of the articles it cites if it has any, and keywords if it has any.
   These lists need to be extracted.

```
def flatten(l):
    if isinstance(l, list):
        return [subelem for elem in l for subelem in flatten(elem)]
    else:
        return [l]

def normalize_item(elem, args):
    ret_val =   tuple(elem[arg] for arg in args)\
              if isinstance(elem, collections.OrderedDict) and set(args).
      issubset(elem.keys())\
              else tuple([elem]+ [None for arg in range(len(args) - 1)])
    return ret_val[0] if len(ret_val) == 1 else ret_val

def extract_one_to_many_relation(collection_in_use, first_arg,
      second_arg, normalize=False, normalizing_params=[]):
    ret_val = [(pub[first_arg], pub[second_arg]) for pub in
      collection_in_use if second_arg in pub.keys()]
    return [(key, normalize_item(item, normalizing_params) if normalize
      is True else item)\
          for key,val in ret_val\
          for item in flatten(val)]

pub_to_authors = extract_one_to_many_relation(dic, "title", "
    authors")
pub_to_keywords = extract_one_to_many_relation(dic, "title", "
    keywords")
citations = extract_one_to_many_relation(dic, "title", "citations")
```

3. The relationships have now been extracted, but the **"publications2authors"** relationship, in its author part, contains all the information of the authors, which needs to be extracted in yet another collection.

```
temp_authors = list(val[1] for val in pub_to_authors)
authors_names = set()
authors = []
```

```
for val in temp_authors:
    if val["name"] not in authors_names:
        authors.append(val)
        authors_names.add(val["name"])
#leave only the name of the authors in the relation as 2nd key
pub_to_authors = list(map(lambda rel: (rel[0], rel[1]["name"]),
    pub_to_authors))
```

4. Now that the relationships have been mapped, **publications** and **authors** must be mapped from a JSON format to a CSV-compliant one while filtering unnecessary information (the JSON still contains the authors, the keyword, and the citations, but they have no further use).

```
def make_publication_csv_compliant(publication, args):
    return tuple(publication[arg] if arg in publication.keys() else None
     for arg in args)
def map_to_csv(collection_in_use, args):
    return list(map(lambda x: make_publication_csv_compliant(x, args),
     collection_in_use))
pub_params = ["title", "type","publisher","journal", "month", "year
    ", "language", "booktitle", "volume", "pages", "ee"]
pub_csv = map_to_csv(dic, pub_params)

auth_params = ["name", "birth_year", "email", "affiliations", "bio"]
auth_csv = map_to_csv(authors, auth_params)
```

5. The last step consists in storing the processed information in the relative *".csv"* files

```
CSVs_to_make = [
    ("publications",pub_params, pub_csv),
    ("authors", auth_params, auth_csv),
    ("publications2authors", ["title","auth_name"], pub_to_authors),
    ("citations", ["citing", "cited"], citations),
    ("keywords", ["title", "keyword"], pub_to_keywords)
]
dir_path = os.path.abspath("") + os.sep + "data"
if not os.path.exists(dir_path):
    os.makedirs(dir_path)
    print("directory added")
for table in CSVs_to_make:
```

```
        path_of_file = dir_path + os.sep + table[0] + ".csv"
        with open(path_of_file, "w") as f:
            writer = csv.writer(f, quoting=csv.QUOTE_NONNUMERIC)
            writer.writerow(table[1])
            writer.writerows(table[2])
        print(f"done with {table[0]}.csv! (num of elements was: {len(
          table[2])})")
    print("done")
```

```
done with publications.csv! (num of elements was: 2000)
done with authors.csv! (num of elements was: 1134)
done with publications2authors.csv! (num of elements was: 2406)
done with citations.csv! (num of elements was: 20227)
done with keywords.csv! (num of elements was: 4060)
done
```

Figure 7.2: output for the pre-processing script

6. In the end, a new folder **data** is created containing the 5 aforementioned files

7.3. Data-frames upload

In order to load the data into the Spark session's data-frames, the following code must be run.

```
from pyspark.sql import SparkSession, Row
from pyspark.sql.functions import col, sum, avg, count, max, min, lower, substring,
    desc, from_unixtime, unix_timestamp, when
from pyspark.sql.types import *
import os


spark = SparkSession.\
        builder.\
        appName("project")\
        .getOrCreate()


dir_path = os.path.abspath("") + os.sep + "data" + os.sep
```

```
load = lambda filename: spark.read.csv(dir_path + filename, header=True,
    inferSchema=True)
publications = load("publications.csv")
authors = load("authors.csv")

pub2auth = load("publications2authors.csv")
citations = load("citations.csv")
keywords = load("keywords.csv")
```

7.4. Queries

7.4.1. Create/Update/Delete operations

Disclaimer: outside of query 3, the procedure for CUD operations is contained within their defined functions, the rest of the code's purpose is to highlight what changed.

Query 1

```
def add_an_author(df_to_update,\
                  name,\
                  birth_year, email,  affiliations , bio):
    val = [(name, birth_year, email,  affiliations , bio)]
    cols = df_to_update.schema
    print(cols)
    new_row = spark.createDataFrame(val, cols)
    return new_row.union(df_to_update)
add_an_author(authors, "Mauro", 1950, "Mauro@mail.it", "Universit della
    Calabria", "Hey there!").show()
```

This creation query adds an author that we can see in the first row of the table.

name	birth_year	email	affiliations	bio
Mauro	1950	Mauro@mail.it	Università della ...	Hey there!
Paul Kocher	1961	paul_kocher@mail.com	ETH Zurich	Pellentesque cur...
Daniel Genkin	1992	daniel_genkin@mai...	The University of...	Proin vel lobort...
Daniel Gruss	1983	daniel_gruss@mail...	The University of...	Aenean nisl mass...
Werner Haas 0004	1968	werner_haas_0004@...	ETH Zurich	Pellentesque et ...
Mike Hamburg	1953	mike_hamburg@mail...	Technical Univers...	Nullam quis maur...
Moritz Lipp	1984	moritz_lipp@mail.com	University of Ill...	Nullam tincidunt...
Stefan Mangard	1962	stefan_mangard@ma...	ETH Zurich	Etiam sed sem id...
Thomas Prescher 0002	1991	thomas_prescher_0...	The University of...	Proin elit mauri...
Michael Schwarz 0001	1999	michael_schwarz_0...	Politecnico di To...	Aliquam convalli...
Yuval Yarom	1998	yuval_yarom@mail.com	Technical Univers...	Fusce eget elit...
Frank Manola	1959	frank_manola@mail...	Politecnico di Mi...	Mauris viverra e...
Michael L. Brodie	1961	michael_l._brodie...	Technical Univers...	Aliquam erat vol...
Michael Stonebraker	1988	michael_stonebrak...	The University of...	Fusce vel blandi...
Mark F. Hornick	1990	mark_f._hornick@m...	University of Ill...	Proin auctor a m...
Joe D. Morrison	1999	joe_d._morrison@m...	The University of...	Etiam ex est, in...
Farshad Nayeri	1964	farshad_nayeri@ma...	Technical Univers...	Mauris at cursus...
Alejandro P. Buch...	1991	alejandro_p._buch...	Politecnico di Mi...	Etiam id nulla r...
M. Tamer Özsu	1995	m._tamer_özsu@mai...	ETH Zurich	Duis rhoncus, ar...
Dimitrios Georgak...	1971	dimitrios_georgak...	The University of...	Integer tincidun...

Query 2

```
def delete_author(authors, pub2auth, auth_name):
    authors = authors.filter(col("name") != auth_name)
    pub2auth = pub2auth.filter(col("auth_name") != auth_name)
    return authors, pub2auth

new_auth, newpub2auth = delete_author(authors, pub2auth, "Paul Kocher")

authors.filter(col("name").contains("Paul")).show()
new_auth.filter(col("name").contains("Paul")).show()
```

This creation query removes all traces of a certain author.
In the tables below we show the tables of the authors (containing "Paul" in the name to make it smaller) before and after the query.

```
+------------------+----------+------------------+------------------+------------------+
|              name|birth_year|             email|       affiliations|               bio|
+------------------+----------+------------------+------------------+------------------+
|       Paul Kocher|      1961|paul_kocher@mail.com|         ETH Zurich| Pellentesque cur...|
|       Paul Ferring|      1979|paul_ferring@mail...|               MIT| Ut tellus dui, p...|
|    Paul B. Hermanns|      1956|paul_b._hermanns@...|The University of...| Fusce malesuada ...|
|Paulo S. L. M. Ba...|      1982|paulo_s._l._m._ba...|Politecnico di To...| Suspendisse sit ...|
|       Paul England|      1952|paul_england@mail...|               MIT| Vestibulum ante ...|
|       Paul Leyland|      1992|paul_leyland@mail...|University of Ill...| Duis non ligula ...|
|     Paul Zimmermann|      2000|paul_zimmermann@m...|Politecnico di Mi...| Proin congue tur...|
|  Paul J. M. Havinga|      1979|paul_j._m._having...|Politecnico di To...| Praesent a ullam...|
|   Paul J. McCullagh|      1975|paul_j._mccullagh...|               MIT| Praesent nibh en...|
+------------------+----------+------------------+------------------+------------------+

+------------------+----------+------------------+------------------+------------------+
|              name|birth_year|             email|       affiliations|               bio|
+------------------+----------+------------------+------------------+------------------+
|       Paul Ferring|      1979|paul_ferring@mail...|               MIT| Ut tellus dui, p...|
|    Paul B. Hermanns|      1956|paul_b._hermanns@...|The University of...| Fusce malesuada ...|
|Paulo S. L. M. Ba...|      1982|paulo_s._l._m._ba...|Politecnico di To...| Suspendisse sit ...|
|       Paul England|      1952|paul_england@mail...|               MIT| Vestibulum ante ...|
|       Paul Leyland|      1992|paul_leyland@mail...|University of Ill...| Duis non ligula ...|
|     Paul Zimmermann|      2000|paul_zimmermann@m...|Politecnico di Mi...| Proin congue tur...|
|  Paul J. M. Havinga|      1979|paul_j._m._having...|Politecnico di To...| Praesent a ullam...|
|   Paul J. McCullagh|      1975|paul_j._mccullagh...|               MIT| Praesent nibh en...|
+------------------+----------+------------------+------------------+------------------+
```

Query 3 (a)

```
publications_month_number_col = publications.fillna({"month": "January", "
    year": 2022})\
.withColumn("month", substring("month",1, 3))\
.withColumn("month_number",from_unixtime(unix_timestamp(col("month"),'
    MMM'),'MM').cast("int"))


publications_month_number_col.select("title", "month", "year", "
    month_number").show()
publications_month_number_col.printSchema()
```

This CRUD operation changes the **publications** data-frame in the following way:

1. 2022 is set as a default year, replacing all possible null values in the column;
2. January is set as default month, replacing all possible null values in the column;
3. All months strings are replaced with a 3 letter equivalent, e.g.: January becomes

Jan;

4. Lastly, a *month_number* column is added to the publication data-frame: this column represents the month number as an integer

```
+--------------------+-----+----+------------+
|               title|month|year|month_number|
+--------------------+-----+----+------------+
|Spectre Attacks: ...|  Jan|2021|           1|
|            Meltdown|  Jan|2020|           1|
|Computer Science ...|  Jan|2013|           1|
|An Evaluation of ...|  Aug|1981|           8|
|DARWIN: On the In...|  Mar|2019|           3|
|Integrating Heter...|  Dec|2021|          12|
|Object Model Capa...|  Jun|2022|           6|
|Integrating Objec...|  Nov|1992|          11|
|Towards a Transac...|  Jun|2018|           6|
|A 'RISC' Object M...|  Aug|2019|           8|
|MetaObject Protoc...|  Dec|1983|          12|
|Object Data Langu...|  Dec|1995|          12|
|Object Data Model...|  Dec|2019|          12|
|Distributed Objec...|  Jun|2007|           6|
|Experiments with ...|  Jul|2009|           7|
|Muffin: A Distrib...|  May|2010|           5|
|Object Oriented D...|  Jan|2021|           1|
|Inheritance for A...|  Jul|2011|           7|
|Modification of U...|  Jan|2006|           1|
|Object ADTs (with...|  Apr|2011|           4|
+--------------------+-----+----+------------+
only showing top 20 rows

root
 |-- title: string (nullable = true)
 |-- type: string (nullable = true)
 |-- publisher: string (nullable = true)
 |-- journal: string (nullable = true)
 |-- month: string (nullable = false)
 |-- year: integer (nullable = false)
 |-- language: string (nullable = true)
 |-- booktitle: string (nullable = true)
 |-- volume: string (nullable = true)
 |-- pages: string (nullable = true)
 |-- ee: string (nullable = true)
 |-- month_number: integer (nullable = true)
```

Query 3 (b)

```
publications_replace_month_col_and_cast = publications.fillna({"month": "
    January", "year": 2022})\
.withColumn("month", substring("month",1, 3))\
.withColumn("month",from_unixtime(unix_timestamp(col("month"),'MMM'),'MM'
    ).cast("int"))


publications_replace_month_col_and_cast.select("title", "month", "year").
    show()
publications_replace_month_col_and_cast.printSchema()
```

This creation query is a variant version of the previous query; the only change is that, instead of adding a **month_number** column to the data-frame, these new values replace the content of the **month** column with said number (the data-frame schema is also changed since the new "month" column is of type int).

```
+--------------------+-----+----+
|               title|month|year|
+--------------------+-----+----+
|Spectre Attacks: ...|    1|2021|
|            Meltdown|    1|2020|
|Computer Science ...|    1|2013|
|An Evaluation of ...|    8|1981|
|DARWIN: On the In...|    3|2019|
|Integrating Heter...|   12|2021|
|Object Model Capa...|    6|2022|
|Integrating Objec...|   11|1992|
|Towards a Transac...|    6|2018|
|A 'RISC' Object M...|    8|2019|
|MetaObject Protoc...|   12|1983|
|Object Data Langu...|   12|1995|
|Object Data Model...|   12|2019|
|Distributed Objec...|    6|2007|
|Experiments with ...|    7|2009|
|Muffin: A Distrib...|    5|2010|
|Object Oriented D...|    1|2021|
|Inheritance for A...|    7|2011|
|Modification of U...|    1|2006|
|Object ADTs (with...|    4|2011|
+--------------------+-----+----+
only showing top 20 rows

root
 |-- title: string (nullable = true)
 |-- type: string (nullable = true)
 |-- publisher: string (nullable = true)
 |-- journal: string (nullable = true)
 |-- month: integer (nullable = true)
 |-- year: integer (nullable = false)
 |-- language: string (nullable = true)
 |-- booktitle: string (nullable = true)
 |-- volume: string (nullable = true)
 |-- pages: string (nullable = true)
 |-- ee: string (nullable = true)
```

Query 4

```
def remove_problematic_keywords(publications, keywords, problematic_tags):
    problematic_publications = keywords.filter(col("keyword").isin(
     problematic_tags))
    return publications\
    .join(
        problematic_publications, publications.title ==
     problematic_publications.title, "leftanti"
    ),\
    keywords.filter(~col("keyword").isin(problematic_tags))


problematic_tags = ["vestibulum", "Sed"]
timer.start()
new_publications, new_keywords = remove_problematic_keywords(publications,
     keywords, problematic_tags)
```

```
timer.stop()

print("old pub keywords vs new pub keywords")
keywords_of_old_publications = publications\
.select("title")\
.join(keywords, publications.title  == keywords.title)\
.select("keyword").distinct().sort("keyword")\
.show()

keywords_of_new_publications = new_publications\
.select("title")\
.join(keywords, publications.title  == keywords.title)\
.select("keyword").distinct().sort("keyword")\
.show()

print("old keywords vs new keywords")
keywords.select(col("keyword")).distinct().sort("keyword").show()

new_keywords.select(col("keyword")).distinct().sort("keyword").show()
```

This creation query removes all the publications with a certain tag.
In the following table, we can see the old keywords and the new keywords.

```
old pub keywords vs new pub keywords
+----------+
|   keyword|
+----------+
|       Sed|
|      diam|
|       non|
|   posuere|
|    sapien|
| tincidunt|
|tincidunt,|
|     velit|
|vestibulum|
|     vitae|
+----------+

+----------+
|   keyword|
+----------+
|      diam|
|       non|
|   posuere|
|    sapien|
| tincidunt|
|tincidunt,|
|     velit|
|     vitae|
+----------+
```

```
old keywords vs new keywords
+----------+
|   keyword|
+----------+
|       Sed|
|      diam|
|       non|
|   posuere|
|    sapien|
| tincidunt|
|tincidunt,|
|     velit|
|vestibulum|
|     vitae|
+----------+

+----------+
|   keyword|
+----------+
|      diam|
|       non|
|   posuere|
|    sapien|
| tincidunt|
|tincidunt,|
|     velit|
|     vitae|
+----------+
```

Query 5

```
def change_language(publications, title, new_lang):
    return publications\
            .withColumn("language",
                        when(
                            col("title") == title, new_lang
                        ).otherwise(col("language")))
new_publications = change_language(publications, "Meltdown", "Russian")
new_publications.select("title", "language").show(truncate=False)
```

This creation query assigns a language to a certain publication.

In the following tables we can see how the language of the publication with the title "Meltdown" is Russian as we selected

```
+--------------------------------------------------------------------------------------------+--------+
|title                                                                                       |language|
+--------------------------------------------------------------------------------------------+--------+
|Spectre Attacks: Exploiting Speculative Execution.                                          |null    |
|Meltdown                                                                                    |Russian |
|Computer Science Curricula 2013                                                             |null    |
|An Evaluation of Object-Oriented DBMS Developments: 1994 Edition.                           |null    |
|DARWIN: On the Incremental Migration of Legacy Information Systems                          |null    |
|Integrating Heterogeneous, Autonomous, Distributed Applications Using the DOM Prototype.|null    |
|Object Model Capabilities For Distributed Object Management.                                |null    |
|Integrating Object-Oriented Applications and Middleware with Relational Databases.          |null    |
|Towards a Transaction Management System for DOM.                                            |null    |
|A 'RISC' Object Model for Object System Interoperation: Concepts and Applications.          |null    |
|MetaObject Protocol Concepts for a RISC Object Model.                                       |null    |
|Object Data Language Facilities for Multimedia Data Types.                                  |null    |
|Object Data Model Facilities for Multimedia Data Types.                                     |null    |
|Distributed Object Management Technology.                                                   |null    |
|Experiments with Dispatching in a Distributed Object System.                                |null    |
|Muffin: A Distributed Database Machine                                                      |null    |
|Object Oriented DBMS as a Generalization of Relational DBMS                                 |null    |
|Inheritance for ADTs (revised)                                                              |null    |
|Modification of User Defined Types                                                          |null    |
|Object ADTs (with improvements for Value ADTs)                                              |italian |
+--------------------------------------------------------------------------------------------+--------+
only showing top 20 rows
```

7.4.2. Search queries

Query 1

```
pub2auth\
.filter(pub2auth.auth_name == "Paul Kocher")\
.join(publications, pub2auth.title == publications.title , "leftouter" )\
.show()
```

This first query returns the information about all the publications written by a given author.

The result of the query is in the following table:

	title	auth_name	title	type	publisher	journal	month	year	language	booktitle	volume	pages	ee
0	Spectre Attacks: Exploiting Speculative Execut...	Paul Kocher	Spectre Attacks: Exploiting Speculative Execut...	article	None	meltdownattack.com	None	2021	None	None	None	None	https://spectreattack.com/spectre.pdf
1	Meltdown	Paul Kocher	Meltdown	article	None	meltdownattack.com	None	2020	None	None	None	None	https://meltdownattack.com/meltdown.pdf

Query 2

```
publications\
.filter(publications.year > 1990)\
.filter(col("title").rlike("computer|Computer"))\
.limit(4)
```

This query returns the publications, limited to four, that have been written after 1990 and contain the word "computer".
The result of the query is in the following table:

	title	type	publisher	journal	month	year	language	booktitle	volume	pages	ee
0	Computer Science Curricula 2013	book	ACM Press and IEEE Computer Society Press	None	None	2013	None	None	None	None	https://www.wikidata.org/entity/Q107021707
1	Das Projekt EXCEPT: Expert-System for Computer...	article	IBM Germany Science Center, Institute for Know...	IWBS Report	None	2016	spanish	None	114	None	None
2	Zur Systematik morphologischer Paradigmen: Die...	article	IBM Germany Science Center, Institute for Know...	IWBS Report	None	2012	afrikaans	None	217	None	None
3	Erster EXCEPT-Workshop: Computerunterst◆tzte U...	article	IBM Germany Science Center, Institute for Know...	IWBS Report	None	2018	None	None	125	None	None

Query 3

```
langs = ["spanish", "english"]

foreign_speakers = pub2auth\
        .join(
            publications\
            .select(col("title"), col("language"))\
            .filter(col("language").isin(langs)), pub2auth.title ==
    publications.title, "leftsemi"
        )\
        .select(col("auth_name"))

articles_written_by_foreign_speakers = pub2auth\
.join(foreign_speakers, pub2auth.auth_name == foreign_speakers.auth_name,
    "leftsemi")\

publications\
.select(col("title"), col("language"))\
.filter(~(col("language").isin(langs)) | col("language").isNull())\
.select(col("title"))\
```

```
.join(articles_written_by_foreign_speakers, publications.title  ==
    articles_written_by_foreign_speakers.title, "leftsemi")\
.show(truncate=False)
```

This query returns the title of all publications that were written neither in English nor Spanish but at least one of the authors has written some other publication in one of these two languages.

	title
0	Algebraical Optimization of FTA-Expressions
1	An Algebraic Characterization of STUF
2	A Combined Symbolic-Empirical Apprach for the Automatic Translation of Compounds
3	Zur Systemarchitektur von LILOG
4	Mengenorientierte Auswertung von Anfragen in der Logikprogrammiersprache PROLOG
5	Definite Resolution over Constraint Languages
6	A Logical Operational Semantics of Full Prolog
7	Ein Fact Manager zur persistenten Speicherung variabel strukturierter komplexer Objekte
8	How could a good system of practical NLP look like?
9	Attribute Inheritance Implemented on Top of a Relational Database System
10	Implementation Aspects of a Natural Language Understanding System in a Prolog/DB Environment
11	Sort Processing in a Deductive Database System
12	EFTA: A Database Retrieval Algebra for Feature Terms
13	Knowledge in Operation
14	An Overview on Planning Applications in PROTOS-L
15	Attributive Concept Descriptions with Unions and Complements
16	On the Interpretation of Equality, Sorts, and Logic Programming
17	Prozedurale Semantik. Repr&#65533;sentation der Sprechergegenwart.
18	Semantics of Logic Programs with Equational Abstract Data Type Specifications
19	Symmetric Coordination: An Alternative Theory of Phrase Structure

Query 4

```
pub2auth\
.join(publications.select(col("title"), col("year")),
      publications.title  == pub2auth.title)\
.groupBy("auth_name")\
.agg(
    max("year").alias("last wrote in"),
    count(publications.title).alias("wrote") #as
)\
.sort(col("wrote").desc())\
.show(truncate=False)
```

This query returns, for every author, the date of the last publication and the number of

articles written.
The result of the query is in the following table:

	auth_name	last wrote in	wrote
0	Christoph Meinel	2022	54
1	Gerrit Bleumer	2022	35
2	Alex Biryukov	2020	35
3	Bart Preneel	2022	28
4	Friedrich L. Bauer	2021	27
5	Carlisle Adams	2022	26
6	Burt Kaliski	2022	23
7	Peter Landrock	2022	21
8	Dieter Baum	2021	21
9	Christoph Beierle	2022	20
10	Christophe De Canni�re	2022	19
11	Anne Canteaut	2019	19
12	Yvo Desmedt	2020	19
13	Klaus Jansen	2022	18
14	Peter Gritzmann	2022	17
15	Burton S. Kaliski Jr.	2022	17
16	Helmut Seidl	2022	16
17	Lothar Breuer	2018	16
18	David Naccache	2022	16
19	Sabrina De Capitani di Vimercati	2020	15

Query 5

```
publications\
. select (col("language"), col("year"), col("publisher"))\
.filter(col("language") == "english")\
.groupBy(col("publisher"))\
.min("year")\
.na. fill ("unknown")\
.show(truncate=False)
```

This query returns, for every publisher, the date of the first article published in English.
The result of the query is in the following table:

	publisher	min(year)
0	unknown	1975
1	IBM Germany Science Center, Institute for Know...	1987
2	IBM Deutschland GmbH	1997

Query 6

```
pub2auth\
.groupBy(pub2auth.auth_name)\
.agg(
count("title").alias("pubXauthor")
)\
.filter(col("pubXauthor") > 10)\
.sort(col("pubXauthor").desc())\
.limit(10)\
.show(truncate=False)
```

This query returns the top 10 authors with more than 10 publications, by the number of published papers.
The result of the query is in the following table:

	auth_name	pubXauthor
0	Christoph Meinel	54
1	Alex Biryukov	35
2	Gerrit Bleumer	35
3	Bart Preneel	28
4	Carlisle Adams	26
5	Friedrich L. Bauer	25
6	Burt Kaliski	23
7	Peter Landrock	21
8	Dieter Baum	21
9	Christoph Beierle	20

Query 7

```
publications\
.filter((publications.year > 2000)  & (publications.year < 2015))\
.groupBy(publications.year)\
.agg(
count("title").alias("pubXyear")
)\
.filter(col("pubXyear") >50)\
.sort(col("pubXyear").desc())\
.show(truncate=False)
```

This query returns the number of publications grouped by year, for all the years which have more than 50 publications.
The result of the query is in the following table:

	year	pubXyear
0	2011	458
1	2014	71
2	2012	61
3	2013	60
4	2010	51

Query 8

```
written_by_old =  pub2auth\
                        .join(
                            authors.\
                            select(col("name"),col("birth_year"))\
                            .filter(col("birth_year") <= 1960), col("
    auth_name") == col("name")
```

```
    ).select("title")

q = publications\
.select(col("title"), col("type"))\
.join(written_by_old, publications.title == written_by_old.title, "leftanti")\
.join(pub2auth, publications.title == pub2auth.title)\
.join(authors.select(col("name"), col("birth_year")), col("auth_name") == col(
    "name"))\
.groupBy(col("type"), publications.title).agg(
    min(col("birth_year")).alias("oldest"),
    max(col("birth_year")).alias("youngest")
)\
.groupBy(col("type")).agg(
    avg(col("youngest")).alias("avg_youngest"),
    avg(col("oldest")).alias("avg_oldest")
)
```

This query returns the average year of birth of the youngest (and oldest) author per type of publication, which has been written by a team of authors all born after 1960.
The result of the query is in the following table:

	type	avg_youngest	avg_oldest
0	inproceedings	1969.500000	1969.500000
1	mastersthesis	1974.800000	1974.800000
2	article	1984.980337	1978.516854
3	incollection	1980.837912	1978.108516
4	book	1967.000000	1967.000000

Query 9

```
pub2auth\
.join(publications.select(col("title"), col("year"), col("type")), publications
    .title == pub2auth.title)\
.filter(col("type") == "article")\
.filter(col("year") == 2022)\
.groupBy("auth_name")\
```

```
.agg(
    count(publications. title ). alias ("numXauthor")
)\
.filter(col("numXauthor") > 2)\
.show(truncate=False)
```

This query returns the authors with more than two publications of type "article" published in 2022.
The result of the query is in the following table:

	auth_name	numXauthor
0	Helmut Seidl	3
1	Udo Pletat	3
2	Klaus Jansen	4
3	Thomas Ludwig 0001	5
4	Christoph Beierle	3

Query 10

```
pub2auth\
.join(publications. select (col("title"), col("year"), col("type")),  publications
    . title  == pub2auth.title)\
.join(keywords.select(col("title"), col("keyword")), keywords.title ==
    pub2auth.title)\
.filter(col("type") == "article")\
.filter(col("keyword") == "vitae")\
.groupBy(publications. title )\
.agg(
    count("auth_name").alias("numAuthorsXpub")
)\
.filter(col("numAuthorsXpub") >= 5)\
.show(truncate=False)
```

This query returns the articles with at least 5 authors and the "vitae" keyword.

The result of the query is in the following table:

	title	numAuthorsXpub
0	Spectre Attacks: Exploiting Speculative Execut...	10
1	Nonapproximability Results for Partially Obser...	6
2	Scheduling with Incompatible Jobs	6
3	LILOG-DB: Database Support for Knowledge-Based...	5

Query 11

```
topKeyword = (keywords\
.groupBy(keywords.keyword)\
.agg(
    count(col("title")).alias("KeyCounter")
)\
.sort(col("KeyCounter").desc())\
.limit(1)\
.join(keywords, keywords.keyword == keywords.keyword, "inner").drop(
    keywords.keyword)
)

topKeyword\
.join(publications.select(col("title"), col("year")), topKeyword.title ==
    publications.title, "inner").drop(topKeyword.title)\
.filter(col("year") == 2010)\
.show(truncate=False)
```

This query returns all publications of 2010 containing the most used keyword. The result of the query is in the following table:

	KeyCounter	keyword	title	year
0	451	vestibulum	Muffin: A Distributed Database Machine	2010
1	451	vestibulum	On Structuring Domain-Specific Knowledge	2010
2	451	vestibulum	Do We Really Need Common Variable Orders for S...	2010
3	451	vestibulum	A Decomposition Algorithm for Optimization ove...	2010
4	451	vestibulum	Parallel versus Sequential Task-Processing: A ...	2010
5	451	vestibulum	An Efficient Method for Aerodynamic Shape Opti...	2010
6	451	vestibulum	An Even Faster Solver for General Systems of E...	2010
7	451	vestibulum	On MAPA/G/K/K Stations	2010
8	451	vestibulum	On the GI/G/k Queue with Lebesgue-Dominated In...	2010
9	451	vestibulum	Right-to-Left Exponentiation.	2010
10	451	vestibulum	"Secure signatures from the ""strong RSA"" ass...	2010
11	451	vestibulum	Sensor Code Attestation.	2010
12	451	vestibulum	Cramer-Shoup Public Key System.	2010
13	451	vestibulum	NESSIE Project.	2010
14	451	vestibulum	Point Counting.	2010
15	451	vestibulum	Private Information Retrieval.	2010
16	451	vestibulum	Monitoring Patterns of Inactivity in the Home ...	2010
17	451	vestibulum	Online Advertising in Social Networks.	2010

Query 12

```
critical_keywords = ["velit", "non"]
pubs_with_keywords =  keywords.filter(col("keyword").isin(critical_keywords)
    )

publications\
.select(col("title"), col("year"))\
.filter(col("year") <= 1980)\
.join(pubs_with_keywords, pubs_with_keywords.title == publications.title)\
.select(publications.title)\
.show(truncate=False)
```

This query returns the title of all publications written in 1980 at the latest and with at least one of the keywords present in the *critical_keywords* list.
The result of the query is in the following table:

	title
0	Two Program Comprehension Tools for Automatic Parallelization: A Comparative Study
1	The Infinite Server Queue with Markov Additive Arrivals in Space
2	Performance Analysis of SDN Specific Error Procedures: Comparison of Step-by-Step and End-to-End Schemes
3	Ein Newtonverfahren zur zeitoptimalen Vibrationsd�mpfung
4	BMAP/G/1-Queues: Properties of the Fundamental-Period-Matrix G
5	Biometric Authentication.
6	Biometric Authentication.
7	Threshold Signature.
8	Binary Euclidean Algorithm.
9	Factorization Circuits.

Query 13

```
prolific_publisher = publications\
.select(col("title"), col("publisher"))\
.filter(col("publisher").isNotNull())\
.groupBy(col("publisher"))\
.agg(
    count(col("title")).alias("count")
)\
.filter(col("count") >= 10)\

publications\
.join(prolific_publisher, publications.publisher == prolific_publisher.publisher
    , "leftsemi")\
.show(truncate=False)
```

This query returns all the publications released by a publisher which has released at least 10 publications.
The result of the query is in the following table:

	title	type	publisher	journal	month	year	language	booktitle	volume	pages	ee
0	Die Repr$\diamond$sentation r$\diamond$umlichen Wissens und die ...	article	IBM Germany Science Center, Institute for Know...	IWBS Report	None	2019	spanish	None	191	None	None
1	Algebraical Optimization of FTA-Expressions	article	IBM Deutschland GmbH	LILOG-Report	None	2020	None	None	59	None	None
2	Wissensrepr$\diamond$sentation und Maschinelles Lernen	article	IBM Deutschland GmbH	LILOG-Report	None	2010	None	None	15	None	None
3	An Algebraic Characterization of STUF	article	IBM Deutschland GmbH	LILOG-Report	None	2021	None	None	40	None	None
4	A Combined Symbolic-Empirical Approach for the ...	article	IBM Germany Science Center, Institute for Know...	IWBS Report	None	2012	None	None	225	None	None
5	Zur Systemarchitektur von LILOG	article	IBM Deutschland GmbH	LILOG-Memo	None	1994	None	None	2	None	None
6	Mengenorientierte Auswertung von Anfragen in d...	article	IBM Deutschland GmbH	LILOG-Report	None	1991	None	None	61	None	None
7	Definite Resolution over Constraint Languages	article	IBM Deutschland GmbH	LILOG-Report	None	2020	None	None	53	None	None
8	Dokumentation der Syntax der LILOG-Grammatik	article	IBM Deutschland GmbH	LILOG-Memo	None	1999	english	None	9	None	None
9	Cognitive Linguistics: The Processing of Spati...	article	IBM Deutschland GmbH	LILOG-Report	None	2015	spanish	None	45	None	None

Query 14

```
publications\
.join(citations, publications.title==citations.citing, "left")\
.groupby(col("title")).agg(
    count("cited").alias("num of citations")
).sort(col("num of citations").desc())\
.limit(3)\
.show()
```

The query returns the three publications which cite the largest number of publications.

	title	num of citations
0	Vigen$\diamond$re Encryption.	38
1	Fast Generation of Random Permutations via Net...	15
2	Macrodata Disclosure Protection.	15

List of Figures