POLITECNICO

MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Erika**

Giovanni

Matteo

Giuseppe

Vito

Group Number: **37**

Academic Year: 2022-2023

Contents

1 | Introduction

1.1. Problem specification

The purpose of this section of the project is to design and implement a graph database which stores bibliographic information on major computer science journals, papers and authors, inspired by already existing systems, such as DBLP.

Starting from an Entity-Relationship diagram, this will be translated into a graph diagram with a few minor changes.

The main objects in this scenario are the following: Publication, Author, Institution, Publisher, Editor, Conference, Journal and Keyword and they need to be linked with meaningful relationships.

1.2. Assumptions

The assumptions taken into account are the following:

- The database makes available truthful data;

- There are 2 types of people, Author and Editor, each one has a global unique identifier (ORCID[1]);

- All authors are affiliated with one university;

- Each publication has a global unique identifier (DOI[2]);

- A publication can be written by more authors ordered as first, middle and last;

- Publication types are the following: book, article, incollection, inproceeding, thesis;

- Multiple editors can work on a single publication.

---

[1]Open Researcher and Contributor ID
[2]Digital Object Identifier

2 | ER Diagram

The objects listed in chapter 1 are now related together and these relationships are reported and formalized in the ER Diagram (Figure 2.1). The designed ER diagram contains the following entities:

- Publication: it is the core entity in the ER diagram, in fact it is related to most entities. Each publication has the following attributes: `id_pub` as primary key, `doi`, `title`, `year_of_publication`, `volume`, `pages`, `language`, `url` (link of the paper's location). This entity is the parent of a total and exclusive ISA hierarchy and its children are:

  - Book (includes the `isbn` as extra attribute)

  - Article (from a journal or magazine)

  - InCollection (part or chapter in a monograph)

  - Inproceeding (paper in a conference)

  - Thesis (PhD thesis or Master thesis)

- Person: it has `id_person` as primary key and other attributes, namely: `name`, `orcid` and `url` (link of the person website). This entity is the parent of a total and overlapping ISA hierarchy and its children are:

  - Author (includes `month_of_birth` and `year_of_birth` as extra attributes)

  - Editor

- Publisher: it has `id_publisher` as primary key and a `name`.

- Institution: it has `id_inst` as primary key and other attributes, namely: `name`, `country`, `national_rank` and `world_rank`.

- Keyword: it has `id_keyword` as primary key and the attribute `word`.

- Journal: it has `id_journal` as primary key and a `name`.

- Conference: it has `id_conf` as primary key and other attributes, namely `name` and `year`.

Among the described entities, the following relationships hold:

- Citing: it indicates that one publication can cite another one or more.

- Publishing: it links one or more publishers to a publication.

- Has: it links a publication with one or more keywords that describe its main topics of interest.

- Editing: it links one or more editors to a publication.

- Writing: it links a publication to the author(s) who have written it. This relationship contains also an attribute that highlights the position of the author's signature in the corresponding paper.

- Containing: it indicates the journal which an article belongs to.

- Holding: it indicates the conferences from which the inproceedings have been extracted.

- Affiliating: it links an institution to each author.
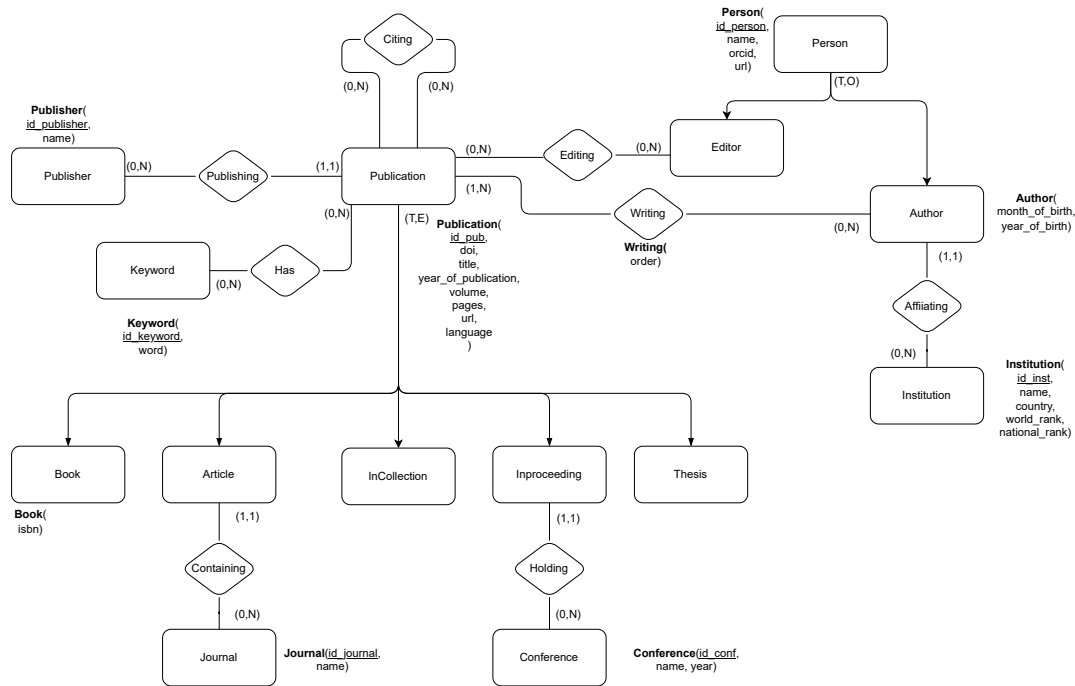
Figure 2.1: ER Diagram

The cardinality of a relationship refers to the nearest entity: for example, in the relationship "Publishing" we consider only 1 publisher for each publication and more publications for each publisher.

2.1. Design Choices and Assumptions

Before designing the ER Diagram, some design choices and assumptions have been made:

- The choice of some entities and attributes is based on some statistics as shown in: Figure 2.2[1] and Figure 2.3[2]. The first one highlights the most frequent types of publications, while the second one the relevant fields related to them (graphs on the next page).

- Each author is associated to a single institution: when an author changes institution we consider him/her to be another instance of Author.

- We assume that an author can be also an editor and viceversa; in fact we modelled the entities Author and Editor as children of a total and overlapping ISA hierarchy.

- Even if the specializations of publication do not have specific attributes, we decided to implement the ISA hierarchy for completeness, clarity and also because it could be helpful for future purpose.

---

[1]https://dblp.org/statistics/publicationsperyear.html
[2]https://github.com/IsaacChanghau/DBLPParser

Publications per year

For this diagram the different publication are grouped by their publication type and publication year. The diagram shows the number of publications of one type per year. The lower diagram can be used to descrease the range of depicted years.
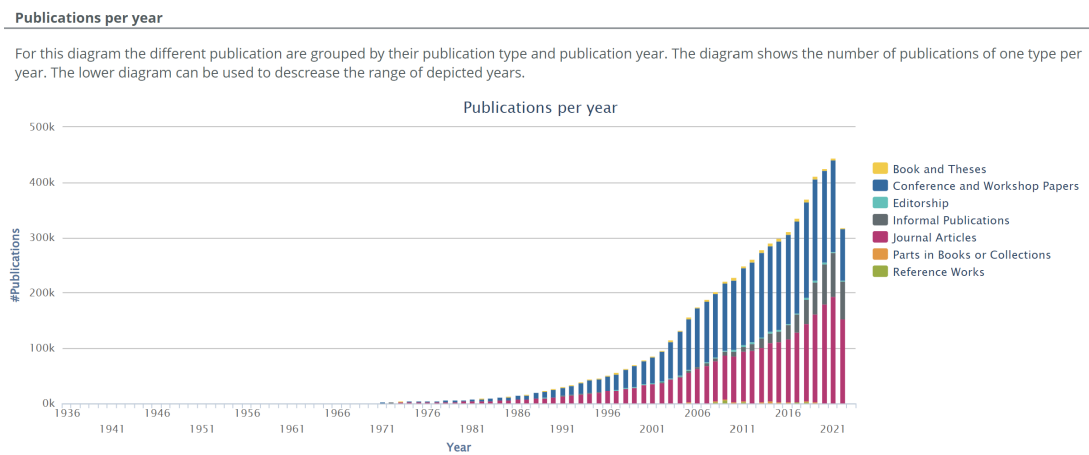
Figure 2.2: Publication per Year
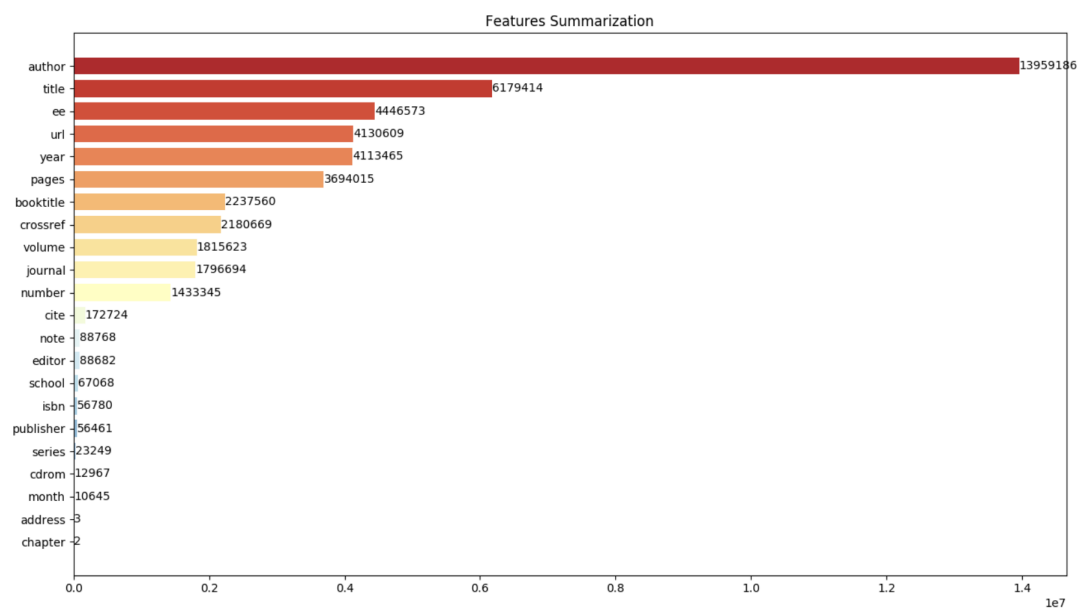
Figure 2.3: Feature Summarization

3 | Graph Diagram

3.1. Design Choices and Assumptions

- Publication is a single node that contains the attribute `doc_type` defining the type of document (book, article, incollection, inproceeding, thesis);

- Some attributes represented in the ER diagram have been removed (due to the lack of data): `volume` and `language` of Publication, `url` of Author;

- When not provided, publication's DOI is replaced with respective resource url;

- The ISA hierarchy involving Author and Editor has been traduced by using a node for each entity;

- Due to lack of data, some editors may have less attributes than authors (sometimes just the name attribute is given). However, this is possible thanks to the schema-less feature of a non-relational database;

- In the graph designed no nodes with double labels are allowed, except for Author and Editor ones: a node, which represents a person, can be labeled with only Author, Editor or both.

3.2. Nodes and Relation

The NODES in the graph are:

- Publication

  - Description: it is the core of the graph, connected to most types of nodes. It includes all the children' attributes of the ISA hierarchy present in the ER diagram.

    A constraint on the uniqueness of the publication DOI has been added.

  - Attributes: doi, title, isbn, pages, year, doc_type

- Author
    - Description: this node represents the author of a publication.

      A constraint on the uniqueness of the author's name has been added.
    - Attributes: name, orcid, month_of_birth, year_of_birth

- Editor
    - Description: this node represents the editor of a publication.

      A constraint on the uniqueness of the editor's name has been added.
    - Attributes: name, orcid, month_of_birth, year_of_birth

- Publisher
    - Description: this node represents the publisher of a publication.

      A constraint on the uniqueness of the publisher's name has been added.
    - Attributes: name

- Journal
    - Description: This node represents the journal in which publications (articles) are published.

      A constraint on the uniqueness of the journal's name has been added.
    - Attributes: name

- Conference
    - Description: this node represents a conference in which publications (inproceedings) are presented.

      A constraint on the uniqueness of the conference's name has been added.
    - Attributes: name, year

- Institution
    - Description: this node represents the institution, in particular a university, associated with an author.

      A constraint on the uniqueness of the institution's name has been added.
    - Attributes: name, country, national_rank, world_rank

- Keyword

  - Description: this node represents the keyword of a publication.

    A constraint on the uniqueness of the word has been added.

  - Attributes: word

The LINKS of the graph are:

- CITES

  - Description: it describes the relation between two different publications, in which the first cites the second.

- CONTAINED_IN

  - Description: relation between Journal and Publication of type article.

- HELD

  - Description: relation between Conference and Publication of type in-proceeding.

- PUBLISHED_BY

  - Description: relation between Publisher and Publication, the same as in the ER diagram.

- HAS_WRITTEN

  - Description: relation between Author and Publication, with an attribute that specifies the order of authors in the publication (first, middle, last).

- AFFILIATING

  - Description: relation between Author and Institution, the same as in the ER diagram.

- HAS

  - Description: relation between Keyword and Publication, the same as in the ER diagram.

- EDITS

  - Description: relation between Keyword and Publication, the same as in the ER diagram.
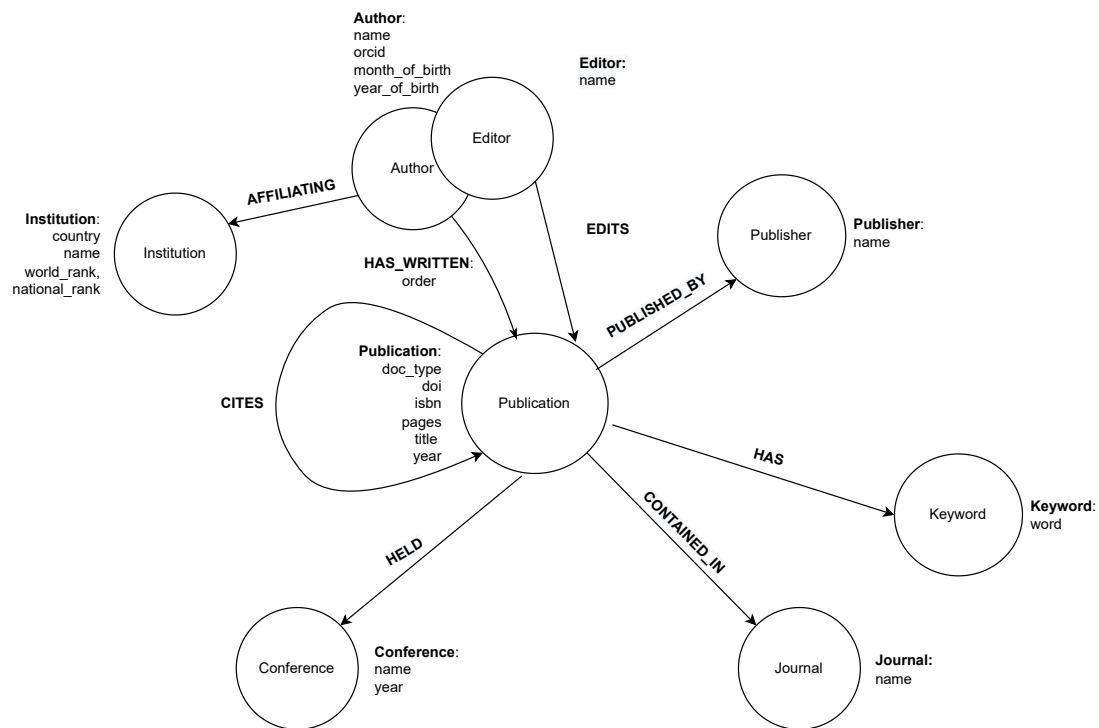
Figure 3.1: Graph Diagram

Note: Author end Editor are represented overlapped because a single node can be both an author and an editor

4 | Neo4j Dataset

4.1. Dataset description

The dataset used is mostly based on data extracted from a subset of the *dblp* dataset[1]. Moreover, the analysis made on xml dataset has been supported by xml respective DTD[2] file, provided by dblp itself. Some significant examples are reported in Listing 1, Listing 2 and Listing 3.

Data extraction and parsing has been made using a `Python` parser[3] found online. The parser takes as input the `XML` file mentioned before and produces as output a collection of `CSV`s, one file for each entity and relationship identified. In fact, all CSV files cited further are part of the parser's output.

The file *authors.csv* contains the whole set of authors. Each author has his own *name* (first name + second nam) and *orcid*, a unique identifier of 16 digits. *month_of_birth* and *year_of_birth* fields have been randomly generated. ORCID has been generated as well only for those who don't have one in dblp dataset.

The file *institution.csv* contains the whole set of institutions. Each institution has its own national and world ranking, name and country. Relationships between institutions and authors have been randomly generated and written into *work_rel.csv* file.

The file *publications.csv* contains the whole set of dblp publications. There are several types of publications, distinguished by the attribute *doc_type* which can assume one of the following values: *book*, *article*, *incollection* and *inproceedings*. The other attributes are the following: *id*, which contains the DOI of each publication, *title*, *year*, *pages*, *isbn* (specified only for books). The total amount of publications is huge; that's why only a total of 4000 entries have been considered: 1000 instances for each type of publication, in order to preserve data heterogeneity.

The file *relationship.csv* contains all the associations between authors and their publica-

---

[1]https://dblp.org/xml/
[2]Document Type Definition
[3]https://github.com/IsaacChanghau/DBLPParser

tions: values of field *author_name* points to *name* field of *author.csv*, while values of field *id* points to *id* field of *publications.csv*. Relationship between authors and publications comes with the *author_order* field: can have only one value among *first*, *middle* and *last*, based on the order in which the authors appears in dblp.

The file *publisher.csv* contains the whole set of dblp publishers. Each publisher comes with his own name. Relationship between publishers and publications are contained in the file *publisher_relationship.csv*.

The file *conferences.csv* contains the whole set of dblp conferences. Each conference comes with the year when was held. Moreover, each Inproceeding (which is a specific type of publication) belongs to a scientific conference. In fact, file *conferences_relationship.csv* contains relationships between publications of *doc_type inproceedings* and conferences.

The journal articles, *doc_type article*, are associated to the journal or magazine, in which they are published, in the *journals_relationship.csv* file.

The following CSV files have been generated by a custom parser. This is due to the fact that these information are not processed by the parser mentioned before.

The file *keywords.csv* contains keywords extracted from the titles of the publications.

Another randomly generated CSV is *citations.csv* which contains the data concerning citations of each publication: the *document* attribute contains the DOI of the publication that cites and the *cite* attribute indicates the publication cited by the former one.

Even editors have been generated randomly and stored in files *editors_relationship.csv* and *editors_authors_relationship.csv*. These files links editors to (some) publications as well. In order to remain consistent with the graph, some editors share some authors' names: this means that those editors are also authors and vice versa.

Listing 1 XML Example - Article

```xml
<article key="journals/cacm/HafleyL62" mdate="2018-11-06">
        <author>W. L. Hafley</author>
        <author>J. S. Lewis</author>
        <title>Algorithm 142: Triangular regression.</title>
        <pages>603-604</pages>
        <year>1962</year>
        <volume>5</volume>
        <journal>Commun. ACM</journal>
        <number>12</number>
        <ee>https://doi.org/10.1145/355580.369094</ee>
        <url>db/journals/cacm/cacm5.html#HafleyL62</url>
</article>
```

Listing 2 XML Example - Proceeding

```xml
<proceedings key="conf/bics/2013" mdate="2021-05-25">
        <editor>Derong Liu 0001</editor>
        <editor>Cesare Alippi</editor>
        <editor>Dongbin Zhao</editor>
        <editor orcid="0000-0002-8080-082X">Amir Hussain 0001</editor>
        <title>Advances in Brain Inspired Cognitive Systems -
            6th International Conference, BICS 2013, Beijing, China,
            June 9-11, 2013. Proceedings</title>
        <year>2013</year>
        <publisher>Springer</publisher>
        <series href="db/series/lncs/index.html">Lecture Notes
            in Computer Science</series>
        <volume>7888</volume>
        <ee>https://doi.org/10.1007/978-3-642-38786-9</ee>
        <isbn>978-3-642-38785-2</isbn>
        <booktitle>BICS</booktitle>
        <url>db/conf/bics/bics2013.html</url>
</proceedings>
```

Listing 3 XML Example - InProceeding

```xml
<inproceedings mdate="2018-11-06" key="conf/sigir/TurnbullBTL07">
    <author>Douglas Turnbull</author>
    <author>Luke Barrington</author>
    <author>David A. Torres</author>
    <author>Gert R. G. Lanckriet</author>
    <title>Towards musical query-by-semantic-description
        using the CAL500 data set.</title>
    <pages>439-446</pages>
    <year>2007</year>
    <crossref>conf/sigir/2007</crossref>
    <booktitle>SIGIR</booktitle>
    <ee>https://doi.org/10.1145/1277741.1277817</ee>
    <url>db/conf/sigir/sigir2007.html#TurnbullBTL07</url>
</inproceedings>
```

4.2.   Neo4j Dataset Upload

In this section it is reported the list of Cypher commands used to extract data from CSVs and import it into Neo4j, according to the graph diagram described in chapter 3 (Figure 7.1).

We preferred to use MERGE instead of CREATE to avoid loading possible duplicates from our dataset.

Listing 4 Load 1.1 - Authors

```
LOAD CSV WITH HEADERS FROM 'file:///author.csv'
    AS row FIELDTERMINATOR "|"
WITH row WHERE row.name IS NOT NULL
MERGE (a:Author {
    name: row.name,
    orcid: row.orcid,
    year_of_birth: toIntegerOrNull(row.year_of_birth),
    month_of_birth: toIntegerOrNull(row.month_of_birth)
});
```

Listing 5 Load 1.2 - Conferences

```
LOAD CSV WITH HEADERS FROM 'file:///conferences.csv'
    AS row FIELDTERMINATOR "|"
WITH row WHERE row.name IS NOT NULL
MERGE (con:Conference {
    name: row.name,
    year: toIntegerOrNull(row.year)
});
```

Listing 6 Load 1.3 - Institutions

```
LOAD CSV WITH HEADERS FROM 'file:///institution.csv'
    AS row FIELDTERMINATOR '|'
WITH row WHERE row.institution IS NOT NULL
CREATE (inst:Institution {
    name: row.institution,
    country: row.country,
    world_rank: toIntegerOrNull(row.world_rank),
    national_rank: toIntegerOrNull(row.national_rank)
});
```

Listing 7 Load 1.4 - Journals

```
LOAD CSV WITH HEADERS FROM 'file:///journals.csv'
    AS row
WITH row WHERE row.name IS NOT NULL
MERGE (j:Journal {
    name: row.name
});
```

Listing 8 Load 1.5 - Publications

```
LOAD CSV WITH HEADERS FROM 'file:///publications.csv'
    AS row FIELDTERMINATOR '|'
CREATE (p:Publication {
    doc_type: row.doc_type,
    doi: row.id,
    isbn: row.isbn,
    pages: row.pages,
    title: row.title,
    year: toIntegerOrNull(row.year)
});
```

Listing 9 Load 1.6 - Publishers

```
LOAD CSV WITH HEADERS FROM 'file:///publisher.csv'
    AS row
WITH row WHERE row.name IS NOT NULL
MERGE (ps:Publisher {
    name: row.name
});
```

Listing 10 Load 2.1 - Citations

```
LOAD CSV WITH HEADERS FROM "file:///citation.csv"
    AS row FIELDTERMINATOR "|"
MATCH (p1:Publication{doi:row.document})
MATCH (p2:Publication{doi:row.cite})
MERGE (p1)-[:CITES]->(p2)
```

Listing 11 Load 2.2 - Publications -> Conferences

```
LOAD CSV WITH HEADERS FROM "file:///conferences_relationship.csv"
    AS row FIELDTERMINATOR "|"
MATCH (p:Publication{doi:row.id})
MATCH (c:Conference{name:row.name})
MERGE (p)-[:HELD]->(c)
```

Listing 12 Load 2.3 - Publications -> Journals

```
LOAD CSV WITH HEADERS FROM "file:///journals_relationship.csv"
    AS row FIELDTERMINATOR "|"
MATCH (p:Publication{doi:row.id})
MATCH (j:Journal{name:row.name})
MERGE (p)-[:CONTAINED_IN]->(j)
```

Listing 13 Load 2.4 - Publications -> Keywords

```
LOAD CSV WITH HEADERS FROM "file:///keywords.csv"
    AS row FIELDTERMINATOR ";"
MATCH (p:Publication {doi: row.pubID})
MERGE (k:Keyword {word: row.keyword})
MERGE (p)-[:HAS]->(k)
```

Listing 14 Load 2.5 - Publications -> Publishers

```
LOAD CSV WITH HEADERS FROM "file:///publisher_relationship.csv"
    AS row FIELDTERMINATOR "|"
MATCH (publc:Publication{doi:row.id})
MATCH (publs:Publisher{name:row.name})
MERGE (publc)-[:PUBLISHED_BY]->(publs)
```

Listing 15 Load 2.6 - Authors -> Institutions

```
LOAD CSV WITH HEADERS FROM "file:///work_rel.csv"
    AS row FIELDTERMINATOR "|"
MATCH (a:Author {name: row.name})
MATCH (i:Institution {name: row.university})
MERGE (a)-[:AFFILIATING]->(i)
```

Listing 16 Load 2.7 - Authors -> Publications

```
LOAD CSV with headers from "file:///relationship.csv"
    AS row FIELDTERMINATOR "|"
MATCH (a:Author{name:row.author_name})
MATCH (p:Publication{doi:row.id})
MERGE (a)-[:HAS_WRITTEN{order:row.author_order}]->(p)
```

Listing 17 Load 2.8 - Editors -> Publications

```
LOAD CSV WITH HEADERS FROM "file:///editors_relationship.csv"
    AS row FIELDTERMINATOR "|"
MATCH (p:Publication {doi: row.doi})
MERGE (e:Editor {name: row.editor})
MERGE (e)-[:EDITED]->(p)
```

Listing 18 Load 2.9 - Editors(Authors) -> Publications

```
LOAD CSV WITH HEADERS FROM "file:///editor_authors_relationship.csv"
    AS row FIELDTERMINATOR "|"
MATCH (p:Publication {doi: row.doi})
MATCH (a:Author {name: row.editor})
MERGE (a)-[:EDITS]->(p)
SET a:Editor
```

5 | Neo4j Queries and Create-Update

5.1. Queries

5.1.1. Query 1

Top 5 keywords linked to publications contained in a journal, written by authors working in an institution located in $country.

```
MATCH (pub:Publication)-[:CONTAINED_IN]->(j:Journal)
WITH pub
MATCH (inst:Institution)<-[:AFFILIATING]-(aut:Author)-[:HAS_WRITTEN]->
(pub)-[:HAS]->(kw:Keyword)
WHERE inst.country = $country
WITH kw, COUNT(DISTINCT pub) as num_pub
RETURN kw.word, num_pub
ORDER BY num_pub DESC
LIMIT 5
```

In this example we set $country="USA"

"kw.word"	"num_pub"
"Data"	55
"Dynamic"	8
"machine"	8
"Optimization"	7
"sensor"	6

Figure 5.1: Result of query number 1

5.1.2.   Query 2

Top 5 authors with more than $num citations of their publications.

```
MATCH (aut:Author)-[:HAS_WRITTEN]->()<-[c:CITES]-(pub_cit:Publication)
WITH aut.name as author, COUNT(DISTINCT pub_cit) AS num_cit
WHERE num_cit > $num
RETURN author, num_cit
ORDER BY num_cit DESC
LIMIT 5
```

In this example we set $num=10

```
|"author"             |"num_cit"|

|"Edmond Bianco"      |121      |

|"Jean-Michel Knippel"|57       |

|"Patrick Isoardi"    |23       |

|"Tiziana Catarci"    |17       |

|"Eric Olivier"       |15       |
```

Figure 5.2: Result of query number 2

5.1.3.   Query 3

Top 5 authors having more than $num papers taken from a conference.

```
MATCH (aut:Author)-[:HAS_WRITTEN]->(p:Publication)-[:HELD]->(conf:Conference)
WITH aut.name as author, COUNT(DISTINCT conf) AS num_conf
WHERE num_conf > $num
RETURN author, num_conf
ORDER BY num_conf DESC, author
LIMIT 5
```

In this example we set $num=2

"author"	"num_conf"
"Karl Aberer"	4
"Patrick Valduriez"	4
"Tiziana Catarci"	4
"Anastasia Analyti"	3
"Clarence A. Ellis"	3

Figure 5.3: Result of query number 3

5.1.4. Query 4

The 5 pairs of authors, with the greatest age difference, who wrote together a paper linked to the conference $name.

Listing 19 Query 4

```
MATCH (p:Publication)-[:HELD]->(c:Conference{name:$name})
WITH p
MATCH (a1:Author)-[w1:HAS_WRITTEN]->(p)<-[w2:HAS_WRITTEN]-(a2:Author)
WHERE a1.year_of_birth > a2.year_of_birth
WITH a1, a2, (a1.year_of_birth - a2.year_of_birth) as AgeDifference, p
RETURN a1.name, a2.name, AgeDifference, p.doi
ORDER BY AgeDifference DESC
LIMIT 5
```

In this example we set $name="CoopIS"

"author1"	"author2"	"AgeDifference"	"publication"
"Günter von Bültzingsloewen"	"Arne Koschel"	48	"https://doi.org/10.1109/COOPIS.1996.555017"
"Günter von Bültzingsloewen"	"Arne Koschel"	48	"https://doi.org/10.1109/COOPIS.1997.613823"
"Nassim Laga"	"Emmanuel Bertin"	48	"https://doi.org/10.1007/978-3-031-17834-4_13"
"Meriana Kobeissi"	"Ali Nour Eldin"	47	"https://doi.org/10.1007/978-3-031-17834-4_7"
"Johann Christoph Freytag"	"Michael Stillger"	45	"https://doi.org/10.1109/COOPIS.1997.613817"

Figure 5.4: Result of query number 4

5.1.5.   Query 5

This query returns the pairs of conferences in which the papers of the second are cited, even indirectly, by the papers of the first, ordered by number of citations. The check "c1.year >= c2.year" is done for performance reason because the number of papers of an older conference "linked" to a more recent conference is negligible.

```
MATCH (c1:Conference)<-[:HELD]-(p1:Publication)
MATCH (c2:Conference)<-[:HELD]-(p2:Publication)
WHERE c1 <> c2 AND c1.year >= c2.year
WITH c1, c2, p1, p2
MATCH (p1)-[r1:CITES*1..3]->(p2)
RETURN c1.name, c1.year, c2.name, c2.year, COUNT(r1) AS ConferenceCitation
ORDER by ConferenceCitation DESC
LIMIT 5
```

Analyzing this query with "PROFILE" we noticed that the order of the performed operations is not the one we expected. Therefore we rewrote the query following the actual order of the performed operations and we noticed that this one leads to the same result, but with better performance.

```
MATCH (c1:Conference)<-[:HELD]-(p1:Publication)
MATCH (c2:Conference)
WHERE c1 <> c2 AND c1.year >= c2.year
WITH c1, c2, p1
MATCH (p1)-[r1:CITES*1..3]->(p2)-[:HELD]->(c2)
RETURN c1.name,c1.year,c2.name,c2.year,count(r1) AS ConferenceCitation
ORDER BY ConferenceCitation DESC
LIMIT 5
```

"c1.name"	"c1.year"	"c2.name"	"c2.year"	"ConferenceCitation"
"GOODTECHS"	"2016"	"CoopIS"	"1994"	40
"GOODTECHS"	"2016"	"OTM"	"2002"	33
"OTM"	"2002"	"CoopIS"	"1994"	29
"GOODTECHS"	"2016"	"NOTERE"	"2008"	29
"NOTERE"	"2008"	"CoopIS"	"1994"	14

Figure 5.5: Result of query number 5

5.1.6.   Query 6

Top 3 publishers of book written by authors with average age between 30 and 40.

The author's age is an approximation based only on the current year since there is insufficient data to compute the exact value.

```
MATCH (p2:Publisher)<-[:PUBLISHED_BY]-(p1:Publication{doc_type:"book"})
<-[:HAS_WRITTEN]-(a:Author)
WITH p2 AS publisher, p1 AS publication, avg(date().year - a.year_of_birth)
AS avgAge
WHERE avgAge > 30 AND avgAge < 40
WITH publisher, COUNT(DISTINCT publication) AS num_publications
RETURN publisher.name, num_publications
ORDER BY num_publications DESC
LIMIT 3
```

"publisher.name"	"num_publications"
"Springer"	75
"Springer Vieweg"	3
"Utz"	1

Figure 5.6: Result of query number 6

5.1.7.   Query 7

Distribution of publication types in the year $year.

The use of OPTIONAL MATCH is required otherwise, if a type of publication does not exist for a specific year, the computation made at the end would produce no results.

```
OPTIONAL MATCH (p1:Publication{doc_type:"article", year:$year})
WITH COUNT(p1) AS article
OPTIONAL MATCH (p2:Publication{doc_type:"book", year:$year})
WITH COUNT(p2) AS book, article
OPTIONAL MATCH (p3:Publication{doc_type:"incollection", year:$year})
WITH COUNT(p3) AS incollection, article, book
OPTIONAL MATCH (p4:Publication{doc_type:"inproceedings", year:$year})
WITH COUNT(p4) AS inproceeding, article, book,
incollection
WITH (article + book + incollection + inproceeding) AS tot, article, book,
incollection, inproceeding
RETURN toFloat(article)/tot * 100 AS percentageArticle,
toFloat(book)/tot * 100 AS percentageBook,
toFloat(incollection)/tot * 100 AS percentageIncollection,
toFloat(inproceeding)/tot * 100 AS percentageInproceeding
```

In this example we set $year=2018

"percentageArticle"	"percentageBook"	"percentageIncollection"	"percentageInproceeding"
3.949730700179533	9.874326750448834	76.12208258527828	10.053859964093357

Figure 5.7: Result of query number 7

5.1.8. Query 8

Find the keywords of the 3 publications that are cited the most and that are written by an author working in a top-5 institution in the world.

```
MATCH (p1:Publication)-[:CITES]->(p2:Publication)
MATCH (p2:Publication)-[:HAS]->(k:Keyword)
MATCH (p2)<-[:HAS_WRITTEN]-()-[:AFFILIATING]->(inst:Institution)
WITH inst.world_rank AS wr, k, p2, p1
WHERE wr <= 5
RETURN k.word AS keyword, p2.title AS publication, COUNT(DISTINCT p1)
AS num_citations, wr
ORDER BY num_citations DESC, wr LIMIT 3
```

"keyword"	"publication"	"num_citations"	"wr"
"network"	"An enhanced algorithm for MANET clustering based on multi hops and network density."	4	4
"Data"	"Selectively Materializing Data in Mediators by Analyzing User Queries."	2	1
"Data"	"A Multi-version Transaction Model to Improve Data Availability in Mobile Computing."	2	2

Figure 5.8: Result of query number 8

5.1.9. Query 9

Find the authors working in a $country institution, who wrote a book in $year with the greatest number of pages.

```
MATCH (aut:Author)-[:HAS_WRITTEN]->(pub:Publication{doc_type:"book",year:$year})
MATCH (aut)-[:AFFILIATING]->(inst:Institution{country:$country})
WITH pub, aut, inst
RETURN aut.name AS author, inst.name AS institution, pub.title AS publication,
pub.pages AS pages
ORDER BY pages DESC
LIMIT 3
```

In this example we set $year=2010, $country="USA"

```
|"author"        |"institution"            |"publication"            |"pages"|
|"Hak-Keung Lam" |"University of Kansas"    |"Stability Analysis of Fuzz|215   |
|                |                         |y-Model-Based Control Syste|      |
|                |                         |ms - Linear-Matrix-Inequali|      |
|                |                         |ty Approach"             |      |
|"David Ribas"   |"University of Colorado Bou|"Underwater SLAM for Struct|119   |
|                |lder"                    |ured Environments Using an |      |
|                |                         |Imaging Sonar"           |      |
|"José Neira"    |"Brigham Young University"|"Underwater SLAM for Struct|119   |
|                |                         |ured Environments Using an |      |
|                |                         |Imaging Sonar"           |      |
```

Figure 5.9: Result of query number 9

5.1.10.   Query 10

Find the 5 publications of an year between 2010 and 2020 contained in a journal, that are written by the greatest number of authors coming from institutions located in different countries.

```
MATCH (pub:Publication)-[:CONTAINED_IN]->(j:Journal)
MATCH (pub)<-[:HAS_WRITTEN]-()-[:AFFILIATING]->(inst:Institution)
WITH COUNT(DISTINCT inst.country) AS num_countries, pub
WHERE pub.year > 2010 AND pub.year < 2020
RETURN num_countries, pub.title AS publication, pub.year AS year
ORDER BY num_countries DESC LIMIT 5
```

```
|"num_countries"|"publication"                                     |"year"|
|9              |"ALACRITY: Analytics-Driven Lossless Data Compressio|2013  |
|               |n for Rapid In-Situ Indexing, Storing, and Querying.|      |
|               |"                                                 |      |
|8              |"Contribution of the Living Lab approach to the deve|2013  |
|               |lopment, assessment and provision of assistive techn|      |
|               |ologies for supporting older adults with cognitive d|      |
|               |isorders."                                        |      |
|7              |"Opening Up Data Analysis for Medical Health Service|2016  |
|               |s: Data Integration and Analysis in Cancer Registrie|      |
|               |s with CARESS."                                   |      |
|7              |"Horizontal Business Process Model Integration."  |2015  |
|7              |"Triage Support Algorithm for Patients Classificatio|2013  |
|               |n at Urgency Care Area in a Hospital."            |      |
```

Figure 5.10: Result of query number 10

5.1.11. Query 11

Find the conferences related to publications which are written by at least one author working in a top-3 national rank institution of Milan.

```
MATCH (conf:Conference)<-[:HELD]-(pub:Publication)<-[:HAS_WRITTEN]
-(aut:Author)-[:AFFILIATING]->(inst:Institution)
WHERE inst.national_rank <= 3 AND inst.name CONTAINS 'Milan'
RETURN conf.name AS conference, inst.name AS institution,
inst.national_rank AS nr
```

"conference"	"institution"	"nr"
"NOTERE"	"University of Milan"	2
"CoopIS"	"University of Milan"	2

Figure 5.11: Result of query number 11

5.1.12. Query 12

Find the number of authors under 40 working in the top 5 universities in the world.

In this case the where clause is not used to avoid to delete the institution entry with 0 number of author under 40 years old, a check with a Boolean better fit our scope.

```
MATCH (i:Institution)
WHERE i.world_rank <= 5
WITH i
MATCH (a:Author)-[:AFFILIATING]->(i)
WITH (date().year - a.year_of_birth) < 40 AS age, i
WITH sum(toInteger(age)) AS num_authors, i
RETURN i.name AS institution, num_authors
ORDER BY num_authors DESC
```

```
|"institution"                                |"num_authors"|

|"Stanford University"                        |4            |

|"Harvard University"                         |3            |

|"University of Cambridge"                     |1            |

|"Massachusetts Institute of Technology"|0            |

|"California Institute of Technology"   |0            |
```

Figure 5.12: Result of query number 12

5.1.13. Query 13

Find the minimum path between universities with the greatest number of produced papers and the minimum path between universities with the smallest number of produced papers.

To do so, we use a collection of universities ordered by produced paper.

```
//take the relation as bidirectional, with *2 we take only the paper
//written by an author of that institution
MATCH (i:Institution)-[*1..2]-(p:Publication)
WITH I, COUNT( distinct p) AS pubnum
ORDER BY pubnum DESC
WITH collect(i) as list

MATCH (a:Institution {name: list[0].name}),
      (b:Institution {name: list[1].name}),
      p1 = shortestPath((a)-[*1..8]-(b))

MATCH (c:Institution {name: list[size(list)-2].name}),
      (d:Institution {name: list[size(list)-1].name}),
      p2 = shortestPath((c)-[*1..8]-(d))

WHERE length(p2) > 1 AND length(p1) > 1

return a.name, b.name, p1, c.name, d.name, p2
```
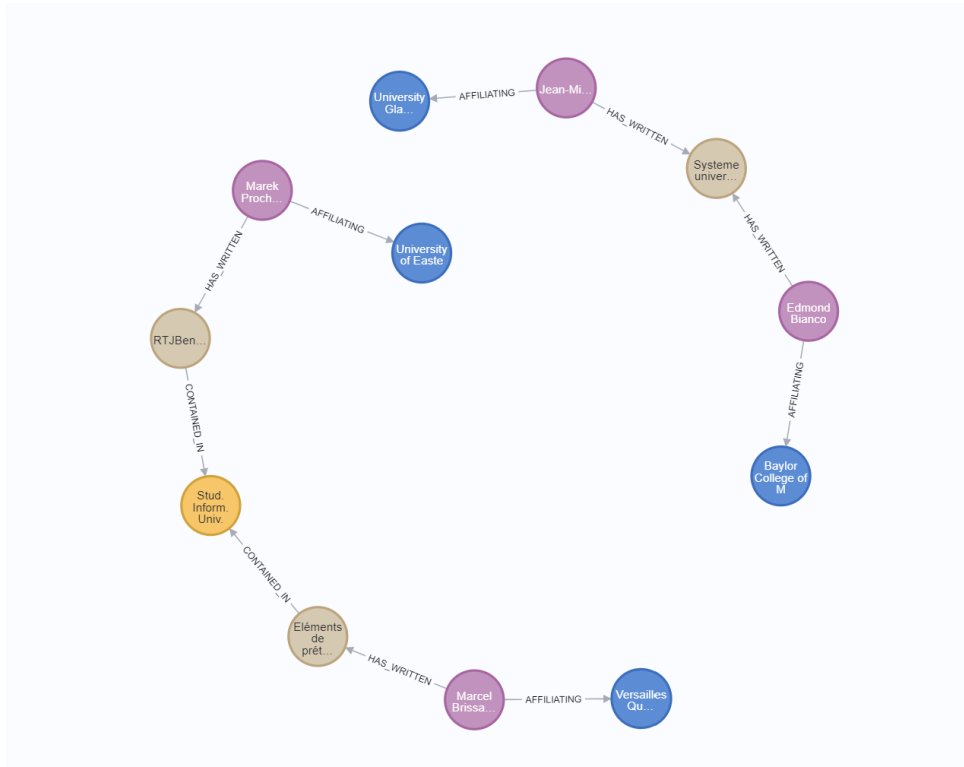
Figure 5.13: Result of query number 13

In this result we can see that the institutions with the highest number of produced papers have a shorter path with respect to the ones with the lowest number of produced papers.

5.2. Neo4J Index

To speed up the computation of our queries we set some indices on the most used attributes. In particular:

- For the queries 4 and 6 we decided to implement an index on Author.year_of_birth

- For the queries 6, 7 and 9 we decided to implement an index on Publication.doc_type

5.3.   Data creation/update commands

5.3.1.   Creation of a Publication

This command creates a new Publication node with the properties indicated by the parameters *$doi, $title, $doc_ type, $pages, $year*. It also creates a HAS_WRITTEN relationship with the author specified in the *$author*, a CITES relationship with each publication contained in the array of doi *$cites* and a HAS relationship with each keyword contained in the array of strings *$keywords*, which contains the most important keywords in the new publication.

```
MERGE (p1:Publication{doi:$doi})
ON CREATE SET
p1.title=$title,
p1.doc_type=$doc_type,
p1.pages=$pages,
p1.year=$year
WITH p1
MATCH (aut:Author{name:$author}), (p1:Publication)
MERGE (aut)-[:HAS_WRITTEN]->(p1)
WITH p1
UNWIND $cites AS cit
MATCH (c:Publication {doi: cit})
MERGE (p1)-[:CITES]->(c)
WITH p1
UNWIND $keywords AS kw
MATCH (k:Keyword {name: kw})
MERGE (p1)-[:HAS]->(k)
```

5.3.2.   Creation of an Institution

This command creates a new node Institution with the properties contained in the parameters *$name, $country, $wr* (world ranking) and *$nr* (national ranking).

```
MERGE (new_inst:Institution {name:$name})
ON CREATE SET
new_inst.country=$country,
new_inst.world_rank= $wr,
new_inst.national_rank= $nr
```

5.3.3. Creation of an Author

The following command creates an Author node, whose properties are specified by *$name*, *$orcid*, *$month_ of_ birth* and *$year_ of_ birth*. It also creates the AFFILIATING relationship with an institution defined by the parameter *$institution*.

```
MERGE (a:Author{name:$name})
ON CREATE SET
a.orcid=$orcid,
a.month_of_birth=$month_of_birth,
a.year_of_birth=$year_of_birth
WITH a
MATCH (i:Institution {name:$institution})
CREATE (a)-[w:AFFILIATING]->(i)
```

5.3.4. Update of the title of a Publication

To change the title of a Publication node we can use the following command, which specifies the previous name in *$old_ name* and the new one in *$new_ name*.

```
MATCH (pub:Publication{title:$old_name})
SET pub.title = $new_name
RETURN pub
```

5.3.5. Update of the name of an Author

To change the name of an Author node we can use the following command, which specifies the previous name in *$old_ name* and the new one in *$new_ name*.

```
MATCH (aut:Author{name:$old_name})
SET aut.name = $new_name
RETURN aut
```

5.3.6.   Update of the articles of a Journal

We can add new articles associated with a journal, whose name is specified by *$journal*, by passing a list of DOI as parameters.

```
UNWIND $articles AS article
MATCH (j:Journal {name:$journal}), (p:Publication {doi:article})
MERGE (p)-[:CONTAINED_IN]->(j)
```

6 | Document Diagram

6.1. Introduction

The purpose of this section of the project is to design and implement a "Document database" which stores information on the major computer science publications, inspired by already existing systems, such as DBLP.

The main focus of the following work is to structure a document properly, so that it can represent all kinds of publications, by exploiting the schema-less feature of NoSQL databases.

7 | Document Database structure

7.1. Design choices and assumptions

The document database has only one collection named "Publication", which stores all the document structured defined as follows.

The document structure have been defined by taking some inspiration from the diagram defined in chapter 3. Obviously, some fields have been added and removed to the final structure in order to respect the requirements given and the fact that the technological context is different (Document DB and not a Graph DB).

The following changes have been applied:

Publication:

- It contains an array of Section (as embedded document).

- It contains a bibliography, that is an array of publications' DOI (as references).

- The array of Authors and the array of Editors have been aggregated to Publication as embedded documents.

- Publisher, Conference, Journal and the array of Keywords have been aggregated to Publication as embedded document.

- It also has in addition an abstract and the number of the volume.

Section:

- It has a title, a body, an array of figures (as embedded document) and it can also have some sub-sections (as embedded document).

- A sub-section is structured exactly like a Section, so it can contain other sub-sections.

- A figure has an URL and a caption.

Author/Editor:

- Unlike Neo4J schema, in MongoDB we consider Author and Editor as two separate entities.

- Editor and Author have name, ORCID, month of birth (mob), year of birth (yob) and mail.

- The Author also contains a bio and an affiliation.

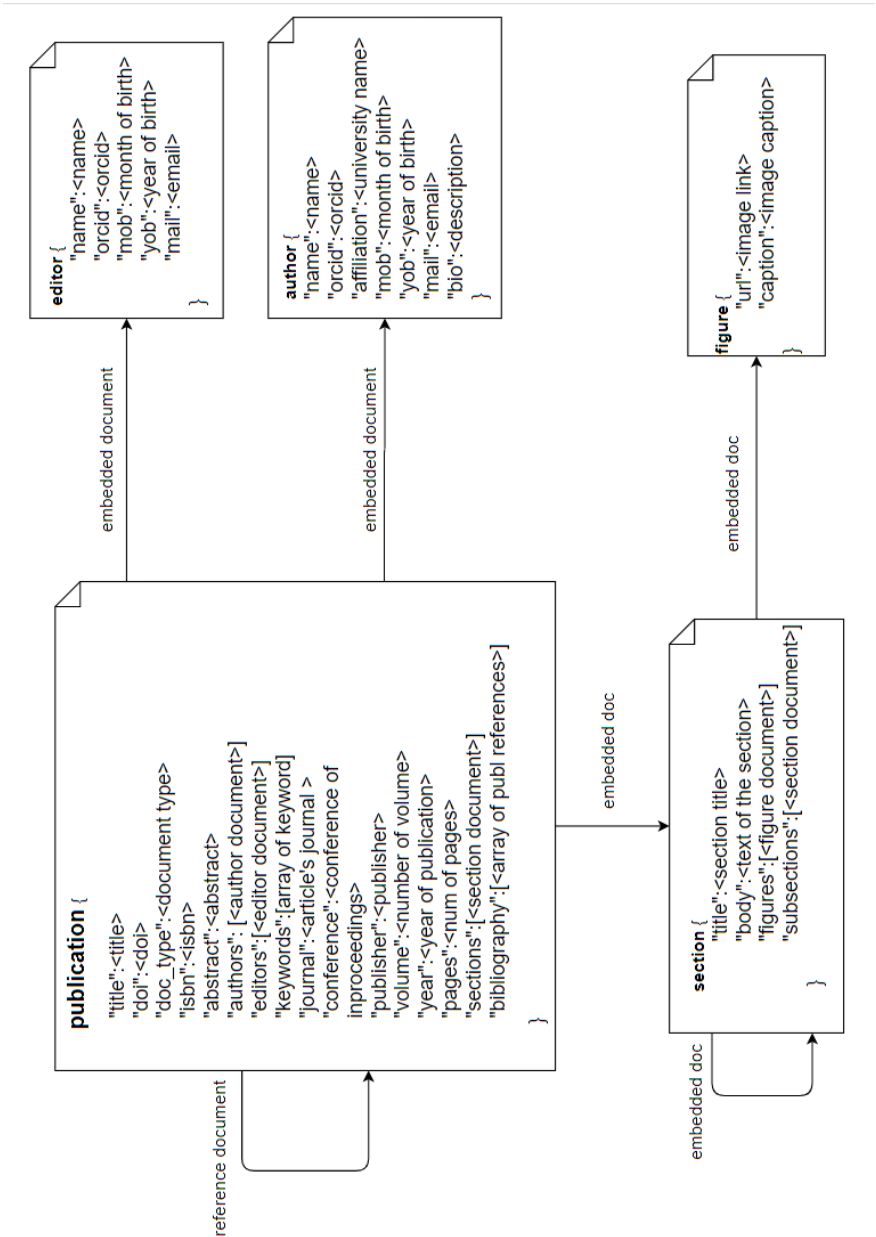This is the schema that we think in order to structure the document:

Figure 7.1: Document Diagram

An example of publication in JSON format is:

```json
{
    "doi": "https://d-nb.info/891654135",
    "title": "Planspieltechnik und ...",
    "year": 1989.0,
    "pages": 370,
    "isbn": "978-3-89012-177-2",
    "publisher": "Eul, Germany",
    "doc_type": "book",
    "authors": [
      {
        "name": "Michael A. Curth",
        "orcid": "6827-5510-6606-7585",
        "affiliation": "Harvard University",
        "mob": 12,
        "yob": 1974,
        "mail": "uwboDG@outlook.it",
        "bio": "..."
      }
    ],
    "sections": [
      {
        "title": "...",
        "figures": [
          {
            "url": "https://picsum.photos/seed/tcfsjmrfhw/200/300",
            "caption": "..."
          },
          {
            "url": "https://picsum.photos/seed/aogvuydquh/200/300",
            "caption": "..."
          }
        ],
        "subsections": [
          {
            "title": "...",
            "body": "...",
            "subsections": [
              {
                "title": "...",
                "body": "..."
              }
            ]}]
    }
```

8 | MongoDB Database upload

The dataset used is mostly based on data extracted from a subset of the *dblp* dataset[1] (same file used for data upload into a Graph database reported into chapter 4). No further analysis have been made on the XML, thanks to previous study made on the dataset.

It is crucial that data uploaded into the database respects the document design defined (in chapter 7). Moreover, in order to perform a data upload as much efficient as possible, older generated files have been used.

The process of translation from a bunch of CSV files into a single JSON which contains all the publications one can be explained into the following points:

1. CSV files used in data upload for graph database have been collected and used as input for *pandas*, a python library which allows data manipulation;

2. some join operations have been performed on CSVs' data via pandas in order to obtain a single *pandas dataframe*[2];

3. export of manipulated data into a (first version of) JSON document, containing all publications including respective fields such as title, authors, the editors, the publisher, the keywords, the cited papers, the journal and the conference, etc. (according to document type);

4. some data is still missing: emails, months and years of birthday (of authors and editors) have been randomly generated using. In order to develop a realistic dataset, some fields, such as abstracts and the text sections, have been populated using a Natural Language Processing dataset. For the sake of simplicity, at most two nested subsections within a section of a publication have been generated. Moreover, random images *URLs* and respective captions have taken from another dataset. All the operations described in this point have been performed by a custom python script;

---

[1]https://dblp.org/xml/
[2]https://pandas.pydata.org

5. the final output is a single file containing a collection of JSON documents, one per publication and structured as designed.

Next step is now the effective data upload: all JSON documents contained into the output file can be uploaded as single documents of a collection by using the *Upload Data* function, available in the MongoDB GUI client, called MongoDB Compass.

9 | MongoDB Queries and Create-Update

9.1. Queries

9.1.1. Query 1

Average number of Sub-Sub-Sections per section in papers written between the 1990 and 2010.

We tried to use an index on "year" but it does not influence the performance of the query in our database setting.

```
db.publications.aggregate([
{
    $match: {
        year: {
            $lte: 2010,
            $gte: 1990
        }
    }
}, {
    $unwind: {
        path: '$sections'
    }
}, {
    $project: {
        doi: 1,
        title: 1,

//this projection is used to set the value of SubSubSection,
//count cannot be used because it would delete 0 value element
        NumOfSubSubSec: {
```

```
            $cond: {
                'if': {
                    $isArray: '$sections.subsections.subsections'
                },
                then: {
                    $size: '$sections.subsections.subsections'
                },
                'else': 0
            }
        }
    }
}, {
//group by doi is used as intermediate step for checking
    $group: {
        _id: '$doi',
        AverageNumOfSubSubSecPerSec: {
            $avg: '$NumOfSubSubSec'
        }
    }
}, {
    $group: {
        _id: true,
        AverageNumOfSubSubSecPerSec: {
            $avg: '$AverageNumOfSubSubSecPerSec'
        }
    }
}])
```


Figure 9.1: Result of query number 1

9.1.2. Query 2

Retrieve all the publications (title, doi, year) cited in the *'World Wide Web'* journal papers.

We tried to use an index on "doi" to boost the performance of the query, this index reduces the time of the query from 88ms to 40ms.

```
db.publications.aggregate([
{
    $match: {
        $and: [
            {journal: 'World Wide Web'},
            {doc_type: 'article'}
        ]
    }
},
{
    $lookup: {
        from: 'publications',
        localField: 'bibliography',
        foreignField: 'doi',
        as: 'citation'
    }
}, {
    $unwind: {
        path: '$citation',
        preserveNullAndEmptyArrays: false
    }
}, {
    $project: {
        'citation.id': 1,
        'citation.doi': 1,
        'citation.title': 1,
        'citation.year': 1
    }
}, {
    $limit: 2
}])
```

Figure 9.2: Result of query number 2

9.1.3.   Query 3

Find all the publications written by at least one author from 'Boston University' and with more than one keyword. It returns the result ordered by number of keywords (descendent).
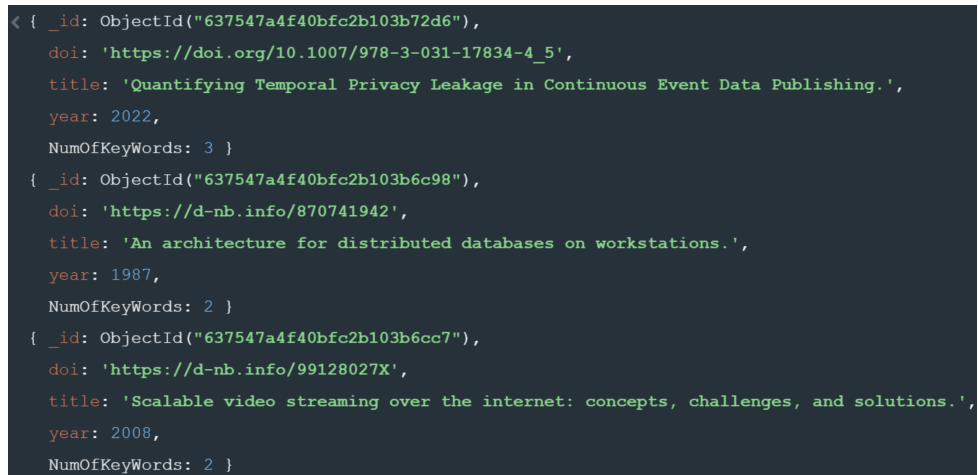
We tried to use an index on "authors.affiliation" to boost the performance of the query. This index reduces the time of the query from 35ms to 1ms.

```
db.publications.aggregate([
{
    $match: {
        'authors.affiliation': 'Boston University'
    }
}, {
    $project: {
        doi: 1,
        title: 1,
        year: 1,
        NumOfKeyWords: {
            $cond: {
                'if': {
                    $isArray: '$keywords'
                },
                then: {
                    $size: '$keywords'
                },
                'else': 0
            }
        }
```

```
    }
}, {
    $match: {
        NumOfKeyWords: { $gt: 1 }
    }
}, {
    $sort: { NumOfKeyWords: −1 }
}
])
```


Figure 9.3: Result of query number 3

9.1.4.   Query 4

Load the first three years which have a publication with the highest number of pages and where the author's mail is @polimi.

```
db.publications.aggregate([
{
    $match: {
        'authors.mail': {
            $regex: '.*@mail.polimi.it$'
        }
    }
}, {
    $group: {
        _id: '$year',
        maxPagesInYear: { $max: '$pages' }
    }
```

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

```
< { _id: 2003, maxPagesInYear: 1179 }
  { _id: 1999, maxPagesInYear: 823 }
  { _id: 2014, maxPagesInYear: 817 }
```

Figure 9.4: Result of query number 4

9.1.5. Query 5

Find the publications written by authors who have the word "Milan" in their affiliation
or in their bio. We use a regex to select the word with the Capital or Lowercase Initial
letter.

```
db.publications.find({
    "$or": [{"authors.affiliation": {"$regex": "(?i)m(?-i)ilan"}},
    {"authors.bio": {"$regex": "(?i)m(?-i)ilan"}}]
},{"authors.affiliation":1, "authors.bio":1}
).limit(3)
```

```
< { _id: ObjectId("637547a4f40bfc2b103b6d87"),
    authors:
    [ { affiliation: 'University of Milan',
        bio: 'This product is made in China' },
      { affiliation: 'University of Cologne',
        bio: 'Actually pretty good' } ] }
  { _id: ObjectId("637547a4f40bfc2b103b6e01"),
    authors:
    [ { affiliation: 'Julius Maximilian\'s University of Würzburg',
        bio: 'Just like the real thing!!!' },
      { affiliation: 'University of Milan',
        bio: 'Item Covered in Walnuts.  Needs to have allergens listed.' },
      { affiliation: 'University of Tokyo',
        bio: 'Great for after lunch' } ] }
```

Figure 9.5: Result of query number 5

9.1.6. Query 6

Top 5 authors born before 1960 that have written more pages.

The performances of this query do not change if we create an index on *authors* field.

```
db.publications.aggregate([
{
    $unwind: { path: "$authors" }
},{
    $match: {"authors.yob": { "$lte" : 1960 } }
},{
    $group: {
            _id: { name: "$authors.name", yob: "$authors.yob"},
            totNumPages: { $sum: "$pages"}
    }
},{
    $sort: { "totNumPages": -1}
},{
    $limit: 5
}
])
```



```
{ _id: { name: 'Christos G. Cassandras', yob: 1942 },
  totNumPages: 1627 }
{ _id: { name: 'Edmond Bianco', yob: 1945 }, totNumPages: 1535 }
{ _id: { name: 'David Zhang 0001', yob: 1943 },
  totNumPages: 1501 }
{ _id: { name: 'Luc Devroye', yob: 1958 }, totNumPages: 1481 }
{ _id: { name: 'John A. Vince', yob: 1942 }, totNumPages: 1430 }
```

Figure 9.6: Result of query number 6

9.1.7.   Query 7

Publications that cite papers written in 2010 (limit to 3).

```
db.publications.aggregate([
{
    $lookup: {
        from: "publications",
        localField: "bibliography",
        foreignField: "doi",
        pipeline: [
            {
                $match: { year: 2010 }
            },
            {
                $project: {
                    _id: 0,
                    ref:{
                        doi: "$doi",
                        title: "$title",
                        year: "$year"
                    }
                }
            },
            {
                $replaceRoot: { newRoot: "$ref" }
            }
        ],
        as: "citations"
    }
},
{
    $unwind : {
        path: "$citations",
        preserveNullAndEmptyArrays: false
    }
},
{
    $limit: 3
}])
```

This query is performed in circa 1600 ms without any index. If we create an index on the *doi* field, the performances improve: the computing time decreases to circa 120 ms.

```
{ _id: ObjectId("637547a4f40bfc2b103b6d0d"),
    doi: 'https://doi.org/10.1007/978-1-4939-0790-8_27',
    title: 'Neighborhood Filters and the Recovery of 3D Information.',
    year: 2015 }
{ _id: ObjectId("637547a4f40bfc2b103b6d40"),
    doi: 'https://doi.org/10.1007/978-1-4939-2092-1_31',
    title: 'Modeling and Simulation of Data Center Networks.',
    year: 2015 }
{ _id: ObjectId("637547a4f40bfc2b103b6d42"),
    doi: 'https://doi.org/10.1007/978-1-4899-7637-6_16',
    title: 'People-to-People Reciprocal Recommenders.',
    year: 2015 }
```

Figure 9.7: Result of query number 7

9.1.8.   Query 8

Load the average number of pages of all the publications written after 2015 that contain "machine learning" in the title or in one section. We tried to use an index on "title" but it does not influence the performance of the query in our database setting. We use a regex to select the word with the Capital or Lowercase Initial letter.

```
db.publications.aggregate([
{
    $match: {
        $or: [
            { title: { $regex: '(?i)m(?-i)achine (?i)l(?-i)earning'} },
            { 'sections.title': {$regex: '(?i)m(?-i)achine (?i)l(?-i)earning'}
            { 'sections.body': {$regex: '(?i)m(?-i)achine (?i)l(?-i)earning'}
        ]
    }
},
{
    $match: {
        year: {$gte: 2015}
    }
},
{
    $group: {
        _id: '$year',
```

```
        avgPages: { $avg: '$pages' }
    }
},
{
    $sort: { avgPages: −1 }
},
{
    $limit: 3
}
])
```


Figure 9.8: Result of query number 8

9.1.9. Query 9

Find the publications of type *book* with at least one author born NOT in January.

```
db.publications.find({
    "authors.mob":{ "$ne":1 },
    "doc_type":{ "$eq":"book" }
},
{ "doi":1,"authors.name":1, "authors.mob":1, "doc_type":1 }
).limit(2)
```


Figure 9.9: Result of query number 9

9.1.10. Query 10

Find books with "Software" in the title, written after 1990 by authors with average age between 60 and 70.
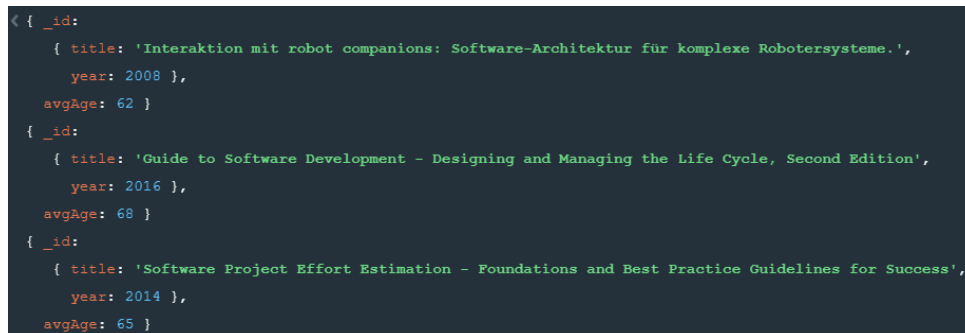
This query requires a text index on the title of the publication.

```
db.publications.aggregate([
{
    $match: {
        $and: [
            { doc_type: 'book' },
            { year: { $gt: 1990 } },
            { $text: { $search: 'Software' } }
        ]
    }
},
{
    $unwind: { path: '$authors' }
},
{
    $group: {
        _id: {
            title: '$title',
            year: '$year'
        },
        avgAge: {
            $avg: {
                $subtract: [
                    { $year: new Date() },
                    '$authors.yob'
                ]
            }
        }
    }
},
{
    $match: {
        $and: [
            { avgAge: { $gt: 60 } },
            { avgAge: { $lt: 70 } }
```

```
        ]
    }
},
{
    $limit: 3
}
])
```


Figure 9.10: Result of query number 10

9.1.11.   Query 11

Finds the top 3 most cited inproceedings (title, doi, year) written by an author(s) affiliated
to University of Milan.

Execution time with no indexes: 50ms.
Execution time with with indexes on *doi* and *authors.affiliation*: 25ms.

```
db.publications.aggregate([
  {
    $match: {
      "authors.affiliation": "University of Milan",
      doc_type: "inproceedings"
    }
  },
  {
    $lookup: {
      from: "publications",
      localField: "doi",
      foreignField: "bibliography",
      as: "ref_by"
    }
```

```
    },
    {
      $addFields: {
        ref_by_count: {$size: "$ref_by"}
      },
    },
    {
      $project: {
        title: 1,
        doi: 1,
        year: 1,
        ref_by_count: 1
      }
    {
      $sort: {ref_by_count: -1}
    },
    {
      $limit: 3
    }
])
```


Figure 9.11: Result of query number 11

9.1.12. Query 12

Find top 3 most active editors from PoliMi (with more contributions).

Execution time with no indexes: 15ms.
Execution time with with indexes *editors.mail*: 15ms.

Query has not been executed using indexes due to the fact that, by documentation, case

insensitive regex cannot use indexes in an effective way[1].

```
db.publications.aggregate([
  {
    $unwind: {path: "$editors"}
  },
  {
    $match: {
      "editors.mail": {$regex: ".*@mail.polimi.it$"}
    }
  },
  {
    $group: {
      _id: {
        orcid: "$editors.orcid",
        name: "$editors.name",
        mail: "$editors.mail"
      },
      count: {"$sum": 1}
    }
  },
  {
    $sort: {count: -1, orcid: 1}
  },
  {
    $limit: 3
  }
])
```

Note: All the queries are indented by hand for a better reading of the report, to run them in mongoDB the original indentation is needed

---

[1]https://www.mongodb.com/docs/manual/reference/operator/query/regex/index-use

Figure 9.12: Result of query number 12

9.2. Data creation/update commands

9.2.1. Delete an author from a publication

This command deletes an author, identified by his orcid (*ORCID*), from the vector "authors" of a specific publication, identified by its doi (*DOI*).

```
db.publications.updateOne(
    { "doi": "https://d-nb.info/891654135" },
    { $pull:{"authors":{orcid:"ORCID"}} },
)
```

9.2.2. Update a publication inserting new cited papers

This command updates a publication document, identified by $doi, appending to the *bibliography* field the cited papers' DOIs ($doi_cit1 and $doi_cit2). If the field does not exist, the command creates it filling it with the specified parameters.

```
db.publications.updateOne(
    { doi: $doi },
    { $push: { bibliography: { $each : [$doi_cit1, $doi_cit2] } } } }
)
```

9.2.3.   Insert a new publication

```
db.publications.insertOne({
    "doi": "http://12345",
    "title": "Publication Title",
    "year": 2022,
    "pages": 300,
    "isbn": "abcdef",
    "publisher": "Hanser",
    "doc_type": "book",
    "authors": [{
        "name": "Mario Rossi",
        "orcid": "1111-2222-3333-4444",
        "affiliation": "Politecnico di Milano",
        "mob": 11,
        "yob": 1985,
        "mail": "m.rossi@polimi.it",
        "bio": "Something..."
    }],
    "sections": [{
        "title": "Section",
        "body": "Something...",
        "figures": [{
            "url": "https://picsum.photos/seed/zxdwvzsjvg/200/300",
            "caption": "Something..."
        }]
    }],
    "abstract": "Abstract of document..."
})
```

9.2.4.   Insert a subsection into a section

This command updates a publication document, identified by its doi, by adding a new subsection into the first section.

```
db.publications.updateOne(
    { "doi": "http://12345"},
    { $push: { "sections.0.subsections": {
                "title": "Subsection",
                "body": "Something...",
```

```
            "figures": [{
                "url": "https://picsum.photos/seed/zxdwvzljvg/200/300",
                "caption": "Something..."
            }]
        }}
    }
)
```

9.2.5.   Delete a figure from a section

This command removes a figure, identified by its url, from a section of a document identified by its doi.

```
db.publications.update(
    { "doi": "http://12345"},
    { $pull: { "sections.0.figures": {
                url: "https://picsum.photos/seed/zxdwvzsjvg/200/300" }
            }
    }
)
```

10 | Spark Data Processing Framework

10.1. Introduction

The purpose of this section of the project is to design and implement a spark data processing framework to manage information on the major computer science publications, inspired by already existing systems, such as DBLP.

Starting from the data-set used for Graph Diagram, we exploit Spark computational power to access data with a different paradigm in respect to the others Database Systems.

11 | Spark Database upload

The dataset used is mostly based on data extracted from a subset of the *dblp* dataset[1] (same file used for data upload into a Graph database reported into chapter 4). No further analysis has been made on the XML, thanks to previous studies made on the dataset.

Notice that there is one additional field 'mail' in the author's file which is not present in the initial ER diagram; this field was generated to comply with the MongoDB project constraints and it has been kept in this new project for completeness and consistency reasons.

No further changes have been made with respect to the *csv files* used before (except for file names).

The following python script has been used to load all the data in the *Spark Context*. We used a python dictionary to keep *Resilient Distributed Dataset* (RDD) ordered, indexed with csv file name.

```python
path_of_the_directory= 'Directory\Dataset'
dataset = {}
for filename in os.listdir(path_of_the_directory):
    f = os.path.join(path_of_the_directory,filename)
    if os.path.isfile(f):

        # Load a DataFrame
        df = spark.read.option("header", True)\
        .option("delimiter", "|").option("inferSchema",True).csv(f)
        #split is used to get only the filename without extension
        dataset[filename.split(".")[0]]=df
        #print is used to check RDDs type and data correctness
        print("NAME OF THE KEY:" + str(filename.split(".")[0]))
        df.printSchema()
        df.show(2)
```

---

[1]https://dblp.org/xml/

For the sake of clarity, here is the list of all RDDs schemas:

NAME OF THE KEY: authors
```
root
 |-- author_name: string (nullable = true)
 |-- orcid: string (nullable = true)
 |-- month_of_birth: integer (nullable = true)
 |-- year_of_birth: integer (nullable = true)
 |-- mail: string (nullable = true)
```

NAME OF THE KEY: citations
```
root
 |-- document: string (nullable = true)
 |-- cite: string (nullable = true)
```

NAME OF THE KEY: conferences
```
root
 |-- conference_name: string (nullable = true)
 |-- year: integer (nullable = true)
```

NAME OF THE KEY: conferences_relationship
```
root
 |-- id: string (nullable = true)
 |-- conference_name: string (nullable = true)
```

NAME OF THE KEY: editor_authors
```
root
 |-- editor_name: string (nullable = true)
 |-- orcid: string (nullable = true)
 |-- month_of_birth: integer (nullable = true)
 |-- year_of_birth: integer (nullable = true)
 |-- mail: string (nullable = true)
```

NAME OF THE KEY: editor_authors_relationship
```
root
 |-- editor_name: string (nullable = true)
 |-- doi: string (nullable = true)
```

NAME OF THE KEY: institutions
```
root
```

```
 |−− world_rank: integer (nullable = true)
 |−− institution: string (nullable = true)
 |−− country: string (nullable = true)
 |−− national_rank: integer (nullable = true)
```

NAME OF THE KEY: journals
```
root
 |−− journal_name: string (nullable = true)
```

NAME OF THE KEY: journals_relationship
```
root
 |−− id: string (nullable = true)
 |−− journal_name: string (nullable = true)
```

NAME OF THE KEY: keywords
```
root
 |−− keyword: string (nullable = true)
 |−− pubID: string (nullable = true)
```

NAME OF THE KEY: publications
```
root
 |−− id: string (nullable = true)
 |−− title: string (nullable = true)
 |−− year: integer (nullable = true)
 |−− pages: integer (nullable = true)
 |−− isbn: string (nullable = true)
 |−− doc_type: string (nullable = true)
```

NAME OF THE KEY: publishers
```
root
 |−− publisher_name: string (nullable = true)
```

NAME OF THE KEY: publisher_relationship
```
root
 |−− id: string (nullable = true)
 |−− publisher_name: string (nullable = true)
```

NAME OF THE KEY: work_relationship
```
root
 |-- author_name: string (nullable = true)
 |-- university: string (nullable = true)
```

NAME OF THE KEY: write_relationship
```
root
 |-- author_name: string (nullable = true)
 |-- pub_id: string (nullable = true)
 |-- author_order: string (nullable = true)
```

12 | Spark Query and Create-Update

12.1. Queries

12.1.1. Query 1

Compute the average age of the publication's authors with the following limitations:

- Exclude the authors born in December(12) from the computation.

- Exclude the publications with authors (not born in December) whose sum of age is out of the range [275,500].

```python
dataset["write_relationship"].join(dataset["publications"], \
    dataset["write_relationship"].pub_id == dataset["publications"].id,"inner") \
.drop("pub_id","author_order","isbn","year","pages","publisher","doc_type") \
.join(dataset["authors"],dataset["write_relationship"].author_name \
    == dataset["authors"].author_name,"inner") \
.filter(dataset["authors"].month_of_birth != 12) \
.withColumn("age", year(current_date()) - dataset["authors"].year_of_birth) \
.groupBy("id", "title") \
.agg(avg("age").alias("average_age"), sum("age").alias("sum_of_age")) \
.filter((col("sum_of_age") > 275) & (col("sum_of_age") < 500)) \
.sort(col("sum_of_age").asc(), col("average_age").asc()) \
.show(5, truncate=True) #show(5) works as limit, truncate used for clarity
```

```
+------------------+--------------------+-----------+----------+
|                id|               title|average_age|sum_of_age|
+------------------+--------------------+-----------+----------+
|https://doi.org/1...|Managing Data Qua...|       55.4|       277|
|https://doi.org/1...|GHio-Ca: An Andro...|      69.25|       277|
|https://doi.org/1...|DABS-Storm: A Dat...|       55.6|       278|
|http://studia.com...|Document Classifi...|       69.5|       278|
|https://doi.org/1...|Comparing Approac...|       55.8|       279|
+------------------+--------------------+-----------+----------+
```

Figure 12.1: Result of query number 1

12.1.2.   Query 2

Count the number of publications that contain in the title the word "machine"

```
dataset["publications"].filter(col("title").contains("machine")) \
.groupBy() \
.agg(count("id").alias("Number Of Publications"),) \
.show(truncate=False)
```

```
+----------------------+
|Number Of Publications|
+----------------------+
|20                    |
+----------------------+
```

Figure 12.2: Result of query number 2

12.1.3. Query 3

Authors ordered by number of written publications

```
dataset["authors"].join(dataset["write_relationship"],["author_name"],"left") \
.groupBy("author_name") \
.agg(count("pub_id").alias("Number Of Publications")) \
.sort(col("Number Of Publications").desc()) \
.show(5)
```

```
+-------------------+----------------------+
|        author_name|Number Of Publications|
+-------------------+----------------------+
|      Edmond Bianco|                   168|
| Jean-Michel Knippel|                   50|
| Christian S. Jensen|                   21|
|Richard T. Snodgrass|                   20|
|       Eric Olivier|                   17|
+-------------------+----------------------+
```

Figure 12.3: Result of query number 3

12.1.4. Query 4

Top-7 books ordered by number of keywords

```
df_group_count = dataset["publications"].filter(dataset["publications"]. \
    doc_type == "book") \
.join(dataset["keywords"], dataset["publications"].id == \
    dataset["keywords"].pubID, "inner") \
.groupBy("id", "title") \
.agg(count("keyword").alias("Number of keywords")) \
.sort(col("Number of keywords").desc()) \
.limit(7) \
.show(truncate=True)
```

```
+--------------------+--------------------+------------------+
|                  id|               title|Number of keywords|
+--------------------+--------------------+------------------+
|https://doi.org/1...|A Hybrid Delibera...|                 3|
|https://d-nb.info...|Design and evalua...|                 3|
|https://doi.org/1...|Testing and Valid...|                 3|
|https://doi.org/1...|Formal SQL Tuning...|                 3|
|https://doi.org/1...|Fast and Scalable...|                 3|
|https://doi.org/1...|Real-Time C++ - E...|                 3|
|https://doi.org/1...|Data Center Netwo...|                 3|
+--------------------+--------------------+------------------+
```

Figure 12.4: Result of query number 4

12.1.5.   Query 5

The number of publications written by authors with a Polimi email grouped by year,
starting from 2010.

```
# Collect into an array all the authors with a Polimi email
polimi_authors = dataset["authors"].filter(col("mail").rlike("polimi")) \
.select(collect_set("author_name")).collect()[0][0]

dataset["publications"].join(dataset["write_relationship"], \
    dataset["publications"].id == dataset["write_relationship"].pub_id) \
.filter(col("author_name").isin(polimi_authors) & (col("year") >= "2010")) \
.groupBy("year") \
.agg(countDistinct("title").alias("Number of publications")) \
.sort(col("Number of publications").desc()) \
.show()
```

```
+----+----------------------+
|year|Number of publications|
+----+----------------------+
|2018|                   167|
|2015|                   132|
|2016|                    80|
|2020|                    78|
|2013|                    71|
+----+----------------------+
```

Figure 12.5: Result of query number 5

12.1.6. Query 6

Find the top-7 countries ordered by the average number of pages written in the 90'.
Countries with less than 10 written publications are excluded.

```
dataset['publications'].join(dataset['write_relationship'],\
    dataset['publications'].id == dataset['write_relationship'].pub_id, 'inner') \
.drop('author_order', 'isbn', 'publisher', 'doc_type') \
.join(dataset['authors'], dataset['write_relationship'].author_name == \
    dataset['authors'].author_name, 'inner') \
.drop('orcid', 'month_of_birth', 'year_of_birth', 'mail') \
.join(dataset['work_relationship'], dataset['authors'].author_name == \
    dataset['work_relationship'].author_name, 'inner') \
.join(dataset['institutions'], dataset['work_relationship'].university == \
    dataset['institutions'].institution, 'inner') \
.drop('world_rank', 'institution', 'national_rank') \
.filter( (dataset['publications'].year >= '1990') & \
    (dataset['publications'].year < '2000') ) \
.groupBy('country') \
.agg(avg('pages').alias('average number of pages'), \
    countDistinct('id').alias('different publications')) \
.filter( col('different publications') >= 10) \
.sort(col('average number of pages').desc()) \
.limit(7) \
.show(truncate=False)
```

```
+-----------+-----------------------+----------------------+
|country    |average number of pages|different publications|
+-----------+-----------------------+----------------------+
|Canada     |63.275862068965516     |29                    |
|Japan      |62.24193548387097      |59                    |
|Sweden     |60.714285714285715     |13                    |
|South Korea|58.888888888888886     |26                    |
|Spain      |47.142857142857146     |19                    |
|France     |44.375                 |29                    |
|Germany    |42.81666666666667      |57                    |
+-----------+-----------------------+----------------------+
```

Figure 12.6: Result of query number 6

12.1.7.　Query 7

Find 3 authors who have written a book when they were 50 years old.

```
dataset['publications'].join(dataset['write_relationship'],\
    dataset['publications'].id == dataset['write_relationship'].pub_id, 'left') \
.drop('title', 'id', 'pages', 'author_order', 'isbn', 'publisher') \
.join(dataset['authors'], ['author_name']) \
.drop('orcid', 'month_of_birth', 'mail', 'pub_id') \
.withColumn('years difference', (col('year') - col('year_of_birth'))) \
.filter((col('years difference') == 50) & (col('doc_type') == 'book')) \
.limit(3) \
.show()
```

```
+---------------+----+--------+-------------+----------------+
|    author_name|year|doc_type|year_of_birth|years difference|
+---------------+----+--------+-------------+----------------+
|Roman Dementiev|2007|    book|         1957|              50|
|   Oscar Cordón|2020|    book|         1970|              50|
|   Philip S. Yu|2019|    book|         1969|              50|
+---------------+----+--------+-------------+----------------+
```

Figure 12.7: Result of query number 7

12.1.8.　Query 8

PoliMi authors who have written at least 10 publications.

```
authors_df = dataset["authors"].withColumnRenamed('author_name', 'name') \
.filter(col("mail").like("%polimi.it")) \
.join(dataset['write_relationship'], dataset['write_relationship'].author_name \
    == col('name'), 'inner') \
.groupBy('name') \
.agg(count("pub_id").alias("num_publications_written")) \
.filter(col("num_publications_written") >= 10) \
.sort(col("num_publications_written").desc()) \
.show(truncate = False)
```

```
+--------------------+----------------------+
|name                |num_publications_written|
+--------------------+----------------------+
|Wil M. P. van der Aalst|11                 |
|Josep Domingo-Ferrer |10                 |
|Tiziana Catarci     |10                 |
+--------------------+----------------------+
```

Figure 12.8: Result of query number 8

12.1.9.   Query 9

Find top 5 articles edited by youngest group of editors.

```
editors_df = dataset["editor_authors"].withColumnRenamed('editor_name', 'name')

df_group_count = dataset["publications"] \
.filter(dataset["publications"].doc_type == "article") \
.join(dataset["editor_authors_relationship"], \
    dataset["editor_authors_relationship"].doi == dataset["publications"].id, "left") \
.join(editors_df, editors_df.name == \
    dataset["editor_authors_relationship"].editor_name, "left") \
.groupBy("doi") \
.agg(count("doi").alias("num_editors"), \
    round(avg(year(current_date()) - col("year_of_birth")), 1).alias("avg_age"),) \
.sort(col("avg_age").asc_nulls_last()) \
.show(5, truncate = True)
```

```
+--------------------+-----------+-------+
|                 doi|num_editors|avg_age|
+--------------------+-----------+-------+
|https://doi.org/1...|          1|   33.0|
|https://doi.org/1...|          1|   33.0|
|https://doi.org/1...|          1|   33.0|
|https://doi.org/1...|          1|   33.0|
|https://doi.org/1...|          1|   33.0|
+--------------------+-----------+-------+
```

Figure 12.9: Result of query number 9

12.1.10.   Query 10

Compound keywords (e.g. Agent-Based) associated to more than 2 publications

```
dataset['keywords'].filter(col("keyword").like("%-%")) \
.groupBy("keyword") \
.agg(count("keyword").alias("num of publications")) \
.filter(col("num of publications") > 2) \
.sort("num of publications") \
.show(truncate=False)
```

```
+--------------+-------------------+
|keyword       |num of publications|
+--------------+-------------------+
|multi-agent   |3                  |
|Agent-Based   |6                  |
|Content-Based |9                  |
|Object-Oriented|12                |
+--------------+-------------------+
```

Figure 12.10: Result of query number 10

12.1.11.   Query 11

Find authors who have written 3 books and who are also editors of 3 publications

```
# Authors who have written 3 books
list_authors = \
    dataset['write_relationship'].join(dataset['publications'], \
        dataset['write_relationship'].pub_id == \
        dataset['publications'].id, "inner") \
    .filter(col("doc_type") == "book") \
    .groupBy("author_name").count() \
    .filter(col("count") == 3) \
    .select(collect_set("author_name")).collect()[0][0]

# Authors who have written 3 books and edited 3 publications
dataset['editor_authors_relationship'].groupBy("editor_name").count() \
.filter(col("count") == 3) \
.filter(col("editor_name").isin(list_authors)) \
.select(col("editor_name").alias("author/editor")) \
.show()
```

```
+-------------+
|author/editor|
+-------------+
|Teuvo Kohonen|
+-------------+
```

Figure 12.11: Result of query number 11

12.1.12.   Query 12

Authors under 40 of USA institutions who have written at least 5 publications

```
dataset['authors'].withColumn("age", year(current_date()) - \
    dataset['authors'].year_of_birth) \
.filter(col("age") < 40) \
.join(dataset['work_relationship'], dataset['authors'].author_name == \
    dataset['work_relationship'].author_name, "inner") \
.join(dataset['institutions'], dataset['work_relationship'].university == \
    dataset['institutions'].institution, "inner") \
.filter(dataset['institutions'].country == "USA") \
.join(dataset['write_relationship'], dataset['work_relationship'].author_name == \
    dataset['write_relationship'].author_name, "inner") \
.groupBy(dataset['write_relationship'].author_name, "university", "country", \
    "age").count().withColumnRenamed("count", "num of publications") \
.filter(col("num of publications") >= 5) \
.sort(col("num of publications").desc()) \
.show(truncate=False)
```

```
+--------------------+--------------------------------------+-------+---+-------------------+
|author_name         |university                            |country|age|num of publications|
+--------------------+--------------------------------------+-------+---+-------------------+
|Jean-Philippe Lehmann|Wake Forest University               |USA    |33 |12                 |
|Kazuo Goda          |Boston University                     |USA    |38 |7                  |
|Paola Salomoni      |Washington University in St. Louis    |USA    |38 |7                  |
|Tran Khanh Dang     |University of Arkansas   Fayetteville |USA    |36 |6                  |
|Umeshwar Dayal      |Florida State University              |USA    |39 |5                  |
+--------------------+--------------------------------------+-------+---+-------------------+
```

Figure 12.12: Result of query number 12

12.2. Data creation/update commands

The following code inserts a new author in the authors dataframe. A custom dataframe has been used: this allows a more optimized approach to data manipulation.

12.2.1. Insert a new author

```python
schema = StructType([ \
    StructField("author_name", StringType(), True), \
    StructField("orcid", StringType(), True), \
    StructField("month_of_birth", IntegerType(), True), \
    StructField("year_of_birth", IntegerType(), True), \
    StructField("mail", StringType(), True) \
])

print("Before Add")
dataset["authors"].filter(col("orcid")=="3414-5303-4227-4420").show()

df_data= [("Name","3414-5303-4227-4420",3,1999,"assignment@mail.polimi.it")]
newRow = spark.createDataFrame(data = df_data, schema = schema)
dataset["authors"] = dataset["authors"].union(newRow)

print("After Add")
dataset["authors"].filter(col("orcid")=="3414-5303-4227-4420").show()
```

```
Before Add
+-----------+-----+--------------+-------------+----+
|author_name|orcid|month_of_birth|year_of_birth|mail|
+-----------+-----+--------------+-------------+----+
+-----------+-----+--------------+-------------+----+


After Add
+-----------+-------------------+--------------+-------------+-------------------+
|author_name|              orcid|month_of_birth|year_of_birth|               mail|
+-----------+-------------------+--------------+-------------+-------------------+
|       Name|3414-5303-4227-4420|             3|         1999|assignment@mail.p...|
+-----------+-------------------+--------------+-------------+-------------------+
```

Figure 12.13: Result of the addition of an author

12.2.2. Update the year of birth of an author

```
# Previous row
print("Before the update\n")
dataset["authors"].filter(dataset["authors"].author_name == "Gilles Guette") \
                  .show()

# Updated row
print("\nAfter the update\n")
dataset["authors"].withColumn("year_of_birth",
            when(dataset["authors"].author_name == "Gilles Guette", "1999") \
            .otherwise(col("year_of_birth"))) \
        .filter(dataset["authors"].author_name == "Gilles Guette") \
        .show()
```

```
Before the update

+------------+-------------------+-------------+-------------+---------------+
| author_name|              orcid|month_of_birth|year_of_birth|           mail|
+------------+-------------------+-------------+-------------+---------------+
|Gilles Guette|9404-5450-4767-7840|            6|         1985|iwAaxsL@duck.com|
+------------+-------------------+-------------+-------------+---------------+

After the update

+------------+-------------------+-------------+-------------+---------------+
| author_name|              orcid|month_of_birth|year_of_birth|           mail|
+------------+-------------------+-------------+-------------+---------------+
|Gilles Guette|9404-5450-4767-7840|            6|         1999|iwAaxsL@duck.com|
+------------+-------------------+-------------+-------------+---------------+
```

Figure 12.14: Result of the update of an author's year of birth

12.2.3.   Insert a new publication

The following code inserts a new publication in the publications dataframe. A custom dataframe has been used.

```
schema = StructType([ \
    StructField("id", StringType(), True), \
    StructField("title", StringType(), True), \
    StructField("year", IntegerType(), True), \
    StructField("pages", IntegerType(), True), \
    StructField("isbn", StringType(), True), \
    StructField("doc_type", StringType(), True) \
])

df_data = [("1111-2222-3333-4444", "Publication title", 2022, 300, "abcdef", "book")]
new_pub = spark.createDataFrame(data = df_data, schema = schema)

print("Before insert")
dataset['publications'].filter(col("title") == "Publication title").show(truncate=False)

# Insert a new publication
dataset['publications'] = dataset['publications'].union(new_pub)

print("After insert")
dataset['publications'].filter(col("title") == "Publication title").show(truncate=False)
```

```
Before insert
+---+-----+----+-----+----+--------+
|id |title|year|pages|isbn|doc_type|
+---+-----+----+-----+----+--------+
+---+-----+----+-----+----+--------+

After insert
+-------------------+-----------------+----+-----+------+--------+
|id                 |title            |year|pages|isbn  |doc_type|
+-------------------+-----------------+----+-----+------+--------+
|1111-2222-3333-4444|Publication title|2022|300  |abcdef|book    |
+-------------------+-----------------+----+-----+------+--------+
```

Figure 12.15: Result of the addition of a publication

12.2.4. Update PoliMi emails from "@mail.polimi.it" to "@polimi.it"

```python
print("Before update")
dataset["authors"].filter(col("mail").like("%polimi.it")).show(2)

dataset["authors"] = dataset["authors"] \
    .withColumn('mail', when(col("mail").like("%@mail.polimi.it"),
        concat(split(col("mail"), '@')[0], lit('@polimi.it'))) \
    .otherwise(col('mail')))

print("After update")
dataset["authors"].filter(col("mail").like("%polimi.it")).show(2)
```

```
Before update
+-------------------+-------------------+--------------+-------------+-------------------+
|        author_name|              orcid|month_of_birth|year_of_birth|               mail|
+-------------------+-------------------+--------------+-------------+-------------------+
|Willem-Jan van de...|7290-9549-8348-3983|            4|         1960|ApUoi@mail.polimi.it|
|      Sebastian Görg|7498-1590-5157-2600|            1|         1966|zfBvpEX@mail.poli...|
+-------------------+-------------------+--------------+-------------+-------------------+
only showing top 2 rows

After update
+-------------------+-------------------+--------------+-------------+----------------+
|        author_name|              orcid|month_of_birth|year_of_birth|            mail|
+-------------------+-------------------+--------------+-------------+----------------+
|Willem-Jan van de...|7290-9549-8348-3983|            4|         1960|  ApUoi@polimi.it|
|      Sebastian Görg|7498-1590-5157-2600|            1|         1966|zfBvpEX@polimi.it|
+-------------------+-------------------+--------------+-------------+----------------+
```

Figure 12.16: Result of the update of polimi authors' emails

12.2.5. Remove a publication

```python
print("Before the remove\n")
dataset["publications"] \
.filter(dataset["publications"].id == "https://d-nb.info/960448683").show()

# Removing the row
dataset["publications"] = dataset["publications"] \
.where(dataset["publications"].id != "https://d-nb.info/960448683")

print("\nAfter the remove\n")
dataset["publications"] \
.filter(dataset["publications"].id == "https://d-nb.info/960448683").show()
```

```
Before the remove


+-------------------+-------------------+----+-----+----------------+--------+
|                 id|              title|year|pages|            isbn|doc_type|
+-------------------+-------------------+----+-----+----------------+--------+
|https://d-nb.info...|Zwischen Organism...|2001|  239|978-3-8244-4433-5|    book|
+-------------------+-------------------+----+-----+----------------+--------+


After the remove

+---+-----+----+-----+----+--------+
| id|title|year|pages|isbn|doc_type|
+---+-----+----+-----+----+--------+
+---+-----+----+-----+----+--------+
```

Figure 12.17: Result of the elimination of a publication