POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Alessandro Bianco**
Andrea Bosisio
Salvatore Buono
Danilo Castiglia
Luca Muscarnera

Group Number: **17**

Academic Year: 2022-2023

Contents

Contents **i**

1 Introduction **1**
1.1 Abstract 1
1.2 Assumptions 1

2 ER Diagram **3**
2.1 Entities 3
2.2 Relationships 4

3 About the implemented data-set **7**
3.1 Data-set pre-processing 7
3.1.1 Data-set sampling 7
3.2 Differences with respect to implemented databases 11

4 Neo4J Delivery **13**
4.1 Abstract 13
4.2 Database description 13
4.3 Data-set upload and database creation 14
4.3.1 Author nodes loading 14
4.3.2 Publication nodes loading 15
4.3.3 Context nodes loading 15
4.3.4 Association nodes loading 15
4.3.5 Produce links loading 15
4.3.6 References links loading 15
4.3.7 Presented links loading 15
4.3.8 Affiliated links loading 16
4.4 Nodes 16
4.5 Links 16

4.6 Creation/Update Commands . . . 17
4.6.1 New author registration . . . 17
4.6.2 New publication of an existing author . . . 18
4.6.3 Update of an author's web page url . . . 19
4.6.4 Add a reference to a publication . . . 20
4.6.5 Delete an author profile . . . 20
4.7 Queries . . . 21
4.7.1 All co-authors starting from a name . . . 21
4.7.2 Get author by name . . . 21
4.7.3 Most contributing author of an association . . . 22
4.7.4 Number of publications year by year . . . 23
4.7.5 Degrees of separation between two authors . . . 24
4.7.6 Most referenced paper of the year . . . 26
4.7.7 Shortest path of references from two publications . . . 27
4.7.8 Couples with the highest number of publications in a Context . . . 27
4.7.9 Number of publication for each type . . . 29
4.7.10 Authors that have written at least one inproceeding and one article 30
4.7.11 Top associations for number of "interesting" publication . . . 31
4.7.12 Number of publications produced by the authors affiliated to a given association . . . 32
4.7.13 Ranking Algorithm . . . 33

5 MongoDB Delivery **37**
5.1 Abstract . . . 37
5.2 Database description . . . 37
5.2.1 Document Structure . . . 37
5.2.2 Adjustments from the ER model . . . 40
5.2.3 JSON generation . . . 42
5.3 Data-set upload and database creation . . . 42
5.3.1 Publications collection loading . . . 42
5.4 Creation/Update Commands . . . 42
5.4.1 New Scientific Paper insertion . . . 42
5.4.2 Adding the affiliation for the authors of the latter inserted publication 44
5.4.3 Update citations of an existing Scientific Paper . . . 45
5.4.4 Update a document with new page numbers . . . 46
5.4.5 Adding the ORCID and the website URL to an author given his ID 46
5.5 Queries . . . 47

5.5.1 Titles of inproceedings presented at a certain conference by a given publisher . . . 47
5.5.2 Scientific Paper with the highest number of authors . . . 48
5.5.3 Scientific Papers with at least k-citations . . . 48
5.5.4 Top-k conferences of a certain publisher held before a certain year by number of presented papers . . . 49
5.5.5 Documents of a certain author contextualized after a certain year . 50
5.5.6 First conferences organized after a certain year . . . 51
5.5.7 Documents of a certain author with at least k-sections . . . 51
5.5.8 Most cited Scientific Papers of a certain year . . . 52
5.5.9 Preview of a Scientific Paper . . . 53
5.5.10 First 10 images in a Scientific Paper . . . 54
5.5.11 Fetch all the referenced documents of a given Scientific Paper . . . 55
5.5.12 Find Scientific Papers written by authors whose ORCID and website URL are available . . . 56
5.5.13 Find the average number of images per Scientific Paper . . . 58

6 Spark Delivery 61

6.1 Abstract . . . 61
6.2 Database description . . . 61
6.2.1 Schema (s) definition . . . 61
6.3 Data-set upload and database creation . . . 62
6.3.1 Publication schema loading . . . 63
6.3.2 Author schema loading . . . 63
6.3.3 Context schema loading . . . 64
6.3.4 Association schema loading . . . 65
6.3.5 WrittenBy schema loading . . . 65
6.3.6 Cites schema loading . . . 66
6.4 Creation/Update Commands . . . 66
6.4.1 Add new Publication . . . 66
6.4.2 Add a new URL to the Author's URLs list . . . 68
6.4.3 Add a new association . . . 69
6.4.4 Remove a publication that has not been cited and written by a certain author . . . 70
6.4.5 Add new Context . . . 72
6.5 Queries . . . 73

6.5.1 Authors that have written a publication in the same year of a certain author 73
6.5.2 Number of publications that begins with a certain string per year. . 74
6.5.3 Top 10 most referenced publications written by at least an author of a certain association 75
6.5.4 Association with the highest number of affiliated authors 78
6.5.5 Times per year in which authors of a certain association were cited at most k-times 78
6.5.6 Context with at least k-publications presented 79
6.5.7 K-authors that have written a publication that was cited at least one time 80
6.5.8 Average number of author's urls available in each association 81

7 References **83**

List of Figures **85**

1 | Introduction

1.1. Abstract

The purpose of this project is to design a database for supporting the search of computer science related publications, using different NoSQL technologies. In each delivery we used a different technology which manage a different point of view of the general problem. The data-set used to populate the different databases is based on an existing one in order to avoid to deal with fully unreal data.

1.2. Assumptions

The assumptions taken into account are the following:

- **A1:** A Publication can mention only previously dated publication;
- **A2:** A Publication cannot mention itself;
- **A3:** When an Author produces a Publication which is a Thesis, he must be affiliated with at least one Association which is an university;
- **A4:** A Publication which is a Standalone Publication has not a context of presentation;
- **A5:** An Author must have at least one Publication;
- **A6:** A Journal is always and only composed of Publications which are Articles or Incollections;
- **A7:** A Conference is always and only composed of Publications which are Inproceedings;
- **A8:** A Publication must have at least one reference to its digital copy (URL or/and DOI);
- **A9:** An Author has at most one website URL;

- **A10:** All the data loaded in our database are consistent.

2 | ER Diagram

The model we developed to fit with the project's purpose is composed by the following listed Entities and Relationships.

2.1. Entities

- AUTHOR: it is one of the main concept in this project. It represents the entity of whom published something related to computer science world.
  An author is identified by the ORCID (reference) and its attributes are the name, the current profession, the personal web-page's url and the date of birth.

- PUBLICATION: it represents the concept of a publication and, in order to distinguish between a STANDALONE PUBLICATION (that can be either a THESIS or a BOOK) and a COLLECTABLE PUBLICATION, we decided to design an ISA for this entity (also to insert different attributes depending on the type of publication).
  A publication is identified by the DOI and its attributes are the title, an url reference to the publication, the publication date, the volume. As showed below the ISA entities contains sometimes many more attributes in order to generalize as possible the represented "kind" of publication (e.g ISBN, edition and publisher are only for the Book and the number of pages is only related to a COLLECTABLE PUBLICATION).

- CONTEXT: this entity represent the context where a STANDALONE PUBLICATION is presented: also in this entity we design an ISA that divide the CONFERENCE and JOURNAL sub-entity. More precisely the CONFERENCE is a "container" of INPROCEEDINGs and INCOLLECTIONs and a JOURNAL is "container" for ARTICLEs.
  A CONTEXT is identified by the ID and its attributes are the title and the date. The only difference in the ISA sub-entities attributes are the location which is only related to the CONFERENCE.

- ASSOCIATION: it is the entity that represents the concept of an association of researchers.

An association is identified by the `ID` and its attributes are the `name` of the association, the `country`, the `address`, the `url` of its official web-page, the `foundation year` and the type of `funding` (public or private funding).

2.2. Relationships

- `PRODUCE`: is the relation between author and publication, it is identified by the keys of those entities, the author's ORCID and the DOI of the publication.
- `AFFILIATED`: is the relation that links an author whit the affiliated association. The relationship is represented by the ID of the association and the author's ORCID.
- `REFERENCES`: is the relation between two or more publication and it represent the concept of citation/mention of a publication in another one. The relationship is identified by the DOIs of the linked publications.
- `PRESENTED`: is the relation between a publication and its presentation `CONTEXT`. This relationship is identified by the DOI of a publication and the ID of its `CONTEXT`.

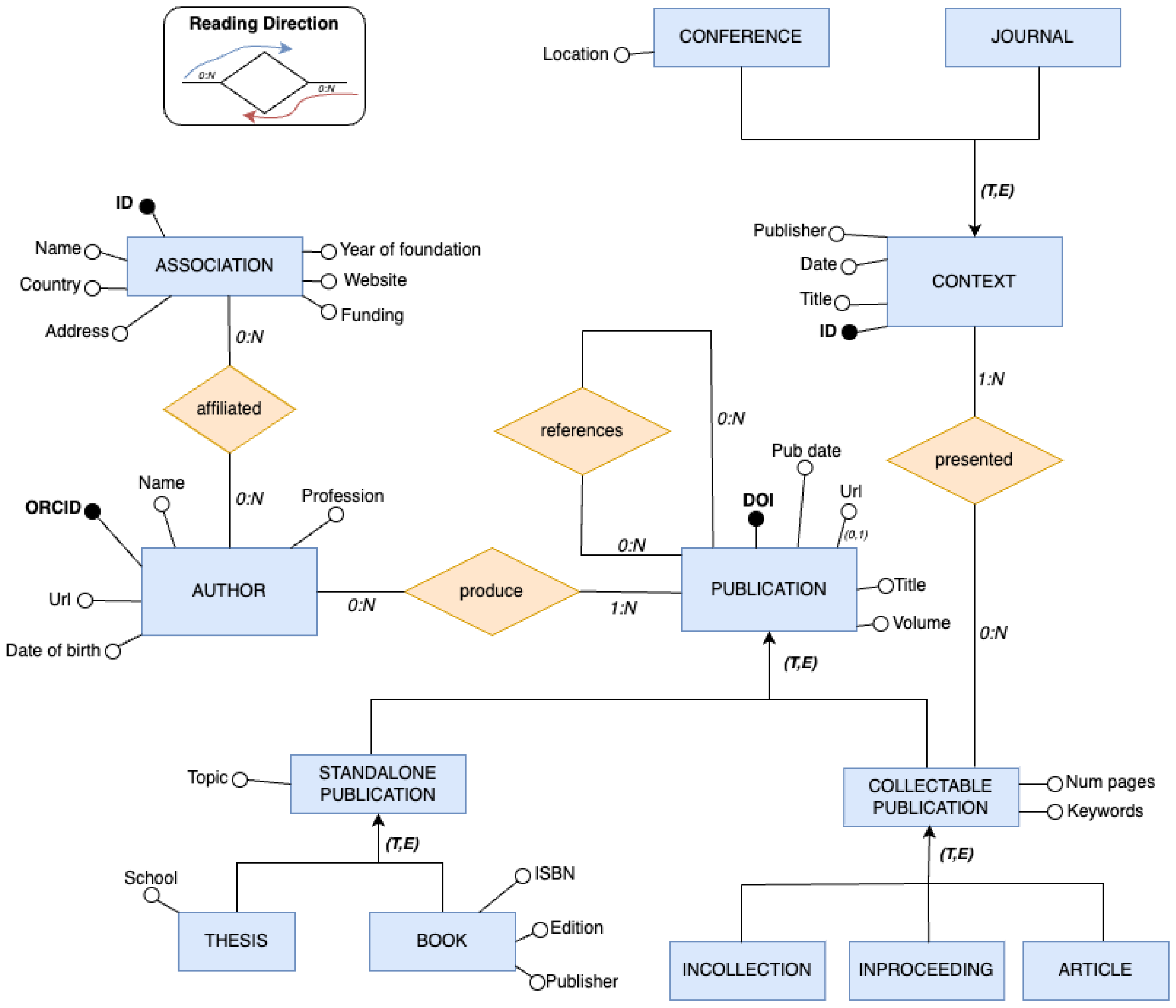


Figure 2.1: ER model

3 | About the implemented data-set

In this chapter is explained how we have obtained our data-set and how we generate the corresponding CSV(s).

3.1. Data-set pre-processing

Our data-set is built based on the DBLP database, whose entire data-set can be downloaded as an XML file from the following link. In particular, we decided to use the release `dblp-2015-03-02.xml` since it is the smallest version available.
Since it is still a big and heavy data-set, we decided to use the public Java dblp-parser (that DBLP made accessible on this webpage) and to exploit its `mmdb` library in order to explore the existent database in a meaningful way and to retrieve a smaller part of it. The parser reads the XML file and populate a model of the database which can be queried using the methods provided by the library.
We implemented in Java a CSV generator that, using the methods of `mmdb`, generates the CSVs of selected entities, with the meaningful attributes: in fact not all the attributes presented in the ER model were founded in the dblp database. In order to deal with this problem, section 4.2 will explain precisely all the differences between our data-set and the ER model.

3.1.1. Data-set sampling

In this section we explain how we have explored the dblp data-set in order to obtain only a meaningful one from it.
The visit starts from a subset of authors of size `INIT_NUM_AUTHORS` and for each of these authors:

1. it generates a CSV entry containing all the author attributes in the `author.csv` file (`AUTHOR` entity) and a CSV entry in the file `author_association_relation.csv`,

which keeps track of the affiliation of an author with an association (AFFILIATED relation),

2. for each of its related publications:

   (a) it adds it in a set of publications (util_pubs) or in a set of contexts (util_context) based on its type,

   (b) it adds all the coauthors of that publication in the authors set, only if the size of the latter set has not reached the threshold MAX_NUM_VISITING_AUTHORS.

This is the code of the beforehead mentioned procedure:

```java
dblp = loadXML(args);
dblp.getPersons().stream().map(Author::new).filter(Author::hasOrcid)
    .limit(INIT_NUM_AUTHORS).forEach(authors::add);
boolean stopAddingAuthors = false;
for (int count = 0; count < authors.size(); count++){
    Author author = authors.get(count);
    if (!author.getPublications().isEmpty()) {
        // author.csv
        author_entries.add(author.generateCSVEntry());
        //author_association_relation.csv
        author_association_entries.add(Arrays.asList(author.getPid(),
            AssociationUtils.getRandomAssociation().getId()));
        for(Publication pub : author.getPublications()) {
            MyPublication publication = new MyPublication(pub);
            // visiting coauthors
            if (!stopAddingAuthors) {
                for (String coauthorName : author.getCoauthorNamesIn(publication)) {
                    Author coauthor = new Author(dblp.getPersonByName(coauthorName));
                    if (authors.size() > MAX_NUM_VISITING_AUTHORS)
                        stopAddingAuthors = true;
                    if (!authors.contains(coauthor))
                        authors.add(coauthor);
                }
            }
            // filtering publications by type
            distributePublication(publication, true);
        }
    }
}
```

Listing 3.1: First authors visiting loop

The visit continues by cycling over all the publications previously added in util_pubs, and for each of them, if it has information about its context in which it was presented or if it is a standalone publication:

1. it creates an entry in the `publications.csv` file (`PUBLICATION` entity),
2. for each of its authors, it creates an entry in the `authors.csv` file and an entry in the `author_pubs_relation.csv` file, which keeps track all the publications written by the author (`PRODUCE` relation),
3. it adds its context in `util_context`.

It does this three steps also for each of the publications that can be cited by that publication and, for each of these possible citation, it also creates an entry in the `pub_pubs_relation.csv`, which keeps track of the citations of a publication (`REFERENCES` relation).

This is the code of the beforehead mentioned procedure:

```java
for (MyPublication publication : util_pubs) {
    addPublicationAndItsRelationEntries(publication, true);

    List<String> citations = publication.getCitations().stream().limit(
        MAX_NUM_CITATIONS_PER_PUB).toList();
    if (!citations.isEmpty()) {
        citations.forEach(cit -> {
            // pub_pubs_relation.csv (citations of a publication)
            citation_entries.add(Arrays.asList(publication.getKey(), cit));
            addPublicationAndItsRelationEntries(dblp.getPublication(cit), true);
        });
    }
}
```

Listing 3.2: Publications visiting loop

The final loop consists in cycling over all the contexts previously added in `util_contexts`, and for each of them:

1. it creates an entry in the `contexts.csv` file (`CONTEXT` entity),
2. for each of the publication presented in that context:
    (a) it creates an entry in the `context_pubs_relation.csv` (`PRESENTED` relation),
    (b) it creates an entry in the `publications.csv` file,
    (c) for each of its authors, it creates an entry in the `authors.csv` file and an entry in the `author_pubs_relation.csv` file.

We would note that in order to avoid inserting multiple times the same entries, we used the Java implementation of Set to store them.

This is the code of the procedure cited above:

```java
for (Context context : util_contexts){
    // contexts.csv
    context_entries.add(context.generateCSVEntry());
    // context_pubs_relation.csv
    List<String> pubs_in_proceedings = context.getRelatedPublications().stream().
        limit(MAX_NUM_PUBS_PER_CONTEXT).collect(Collectors.toList());
    if(!pubs_in_proceedings.isEmpty()){
        pubs_in_proceedings.forEach(pub -> {
            context_pubs_entries.add(context.generateCSVEntry(pub));
            addPublicationAndItsRelationEntries(dblp.getPublication(pub), false);
        });
    }
}
```

Listing 3.3: Contexts visiting loop

The following code explains the method addPublicationAndItsRelationEntries, which is used in the previous codes and whose job is to generate entries in the files publications.csv, authors.csv and author_pubs_relation.csv for a given publication. It also finds the context of that publication adding it to util_pubs exploiting the distributePublication, whose code is omitted for simplicity, if needed.

```java
private static void addPublicationAndItsRelationEntries(Publication publicationToAdd,
    boolean addAlsoItsPossibleContext) {
    MyPublication publication = new MyPublication(publicationToAdd);

    // add publicationToAdd's info (if we have info on its context)
    if (publication.hasContextInfo()) {
        // publications.csv
        publication_entries.add(publication.generateCSVEntry());

        // add all the authors of publicationToAdd (both in authors.csv and
            author_pubs_relation.csv)
        publication.getNames().forEach(authorName -> {
            Author author = new Author(authorName.getPerson());
            author_entries.add(author.generateCSVEntry());
            // author_pubs_relation.csv
            author_pub_entries.add(Arrays.asList(author.getPid(), publication.getKey())
                );
        });
    }

    if (addAlsoItsPossibleContext) {
        if (publication.hasCrossRef()) {
            distributePublication(new MyPublication(dblp.getPublication(publication.
                getCrossRef())), false);
        } else if (publication.getTag().equals("article")) {
                distributePublication(publication, false);
        }
    }
}
```

Listing 3.4: `addPublicationAndItsRelationEntries` method

The explanation of the other methods can be omitted due to the self-explanatory nature of their names. In particular, the method `generateCSVEntry` simply returns all the attributes of the object on which it is called. Those attributes and their difference with respect to the ER model are explained in the following section.

It is also worth mentioning that, prior to the visit, we needed also to add to the set `util_pubs` some publications that has some citations by filtering them from the set of all the publications, since we realized that the cardinality of the relations `REFERENCES` was a lot smaller with respect to the one of the other type of relations (i.e., it was very hard to find a publication that was citing another publication).

3.2. Differences with respect to implemented databases

As mentioned before our data-set is derived from DBLP and for this reasons not all of the attributes that are present in the designed ER model are also present in our implemented databases. Each variation from the ER model is well explained in the dedicated *Database description section* of each delivery's chapter.

It is also important to underline that, if not specified, the key of the entities are the ones we found in the DBLP: this makes the creation of the relationships more and more easier, even if it introduces other differences with respect to the ER model. In some cases this may seem redundant: for example, in the publication entity there are both the ID and the DOI, that conceptually have the same function (they are both identifiers), but since in the DBLP data-set not all the publications have a DOI, we also added the DBLP identifier.

4 | Neo4J Delivery

4.1. Abstract

The objective of this delivery is to design a *graph database* using **Neo4J** technology. In particular this delivery's focus is on implementing a bibliographic database which is an organized digital collection of references to published literature (i.e., the content of a publication is not stored in the database, but it is referenced with an URL).

Therefore, the database to be implemented must be able to store all the relevant metadata of all the considered publications, its authors and other relations and information that are better explained in next sections.

4.2. Database description

The following table presents for each entity of the ER model, the entity we decided to implement in our database (and the selected attributes based on the availability of the information).

ER Model Entity	Implemented Entity
AUTHOR	The only attributes we retrieved for this entity are the name and the personal web page's url.
ASSOCIATION	The associations of our database are all created manually by us; precisely they have the name, the address, the country, the url and the funding (private/public). *[The ID is generated with an hash code from the name]*
PUBLICATION	This entity is very similar to the designed one due to the availability of all its attributes. The only difference is that the Topic and Keywords attributes, respectively related to Standalone Pub. and Collectable Pub., were not implemented due to the unavailability of those data in the XML file.
CONTEXT	The context entity contains all the designed attributes except for the journal's publisher, which were not available from database, and the same for the conference's location.

It is important to denote that we chose to not separate entities that in the ER diagram were defined through an ISA structure, both for the idea of preserving the notion of "Structured Attribute" that an ISA construct may implement, both to allow the queries to be coherent with the usage of an article indexing system such the one that we projected (for example we imagined that an hypothetical user may be more interested in the publication itself rather than the distinction between the various types of context where it may appear). Moreover, this approach was particularly useful in keeping the structure of our database as simple as possible, allowing us to focus more on more important aspects in terms of the structural design of the database.

4.3. Data-set upload and database creation

In order to create our starting database, the following commands must be run using the CSV files generated with the code described before.

4.3.1. Author nodes loading

```
LOAD CSV FROM "file:///authors.csv" as row
CREATE(:Author{id:row[0], name:row[1], url:row[2]})
```

4.3.2. Publication nodes loading

```
LOAD CSV FROM "file:///publications.csv" as row
CREATE(:Publication{id:row[0], type:row[1], title:row[2], doi:row[3],
    year:toInteger(row[4]), volume:row[5], pages:row[6], publisher:row[7],
    url:row[8], isbn:row[9], school:row[10]})
```

4.3.3. Context nodes loading

```
LOAD CSV FROM "file:///contexts.csv" as row
CREATE(:Context{id:row[0], title:row[1], confName:row[2],
    year:toInteger(row[3]), volume:row[4], publisher:row[5], url:row[6]})
```

4.3.4. Association nodes loading

```
LOAD CSV FROM "file:///associations.csv" as row
CREATE(:Association {id:row[0], name:row[1], country:row[2],
    address:row[3], website:row[4], funding:row[5]})
```

4.3.5. Produce links loading

```
LOAD CSV FROM "file:///author_pubs_relation.csv" as row
MATCH (author:Author{id:row[0]})
MATCH (pub:Publication{id:row[1]})
MERGE (author)-[:PRODUCE]->(pub)
```

4.3.6. References links loading

```
LOAD CSV FROM "file:///pub_pubs_relation.csv" as row
MATCH (pub1:Publication{id:row[0]})
MATCH (pub2:Publication{id:row[1]})
MERGE (pub1)-[:REFERENCES]->(pub2)
```

4.3.7. Presented links loading

```
LOAD CSV FROM "file:///context_pubs_relation.csv" as row
MATCH (context:Context{id:row[0]})
MATCH (pub:Publication{id:row[1]})
MERGE (pub)-[:PRESENTED]->(context)
```

4.3.8. Affiliated links loading

```
LOAD CSV FROM "file:///author_association_relation.csv" as row
MATCH (author:Author{id:row[0]})
MATCH (assoc:Association{id:row[1]})
MERGE (author)-[:AFFILIATED]->(assoc)
```

4.4. Nodes

Our database, as explained in section 4.2, contains 4 kinds of nodes with the following attributes :

- Author : id, name, url. It is linked with the affiliated association(s) and all his published publications;
- Association : id, name, country, address, website, funding. It is linked with the affiliated authors;
- Context : id, title, confName, year (INT), volume, publisher, url. It is linked with all the publications presented in this context;
- Publication : id, type, title, doi, year (INT), volume, pages, publisher, url, isbn, school. It is linked with the authors and with the presentation context.

All the node's attributes are string, except for the ones explicitely marked as (INT) that are of type integer. The total number of nodes inserted in the database is 72622 nodes.

4.5. Links

All the relationships designed in the ER model were implemented in the graph database, precisely there are:

- PRODUCE : links the author with all its publications;
- PRESENTED : links the publication with its context of presentation;
- AFFILIATED: links the author with its association;
- REFERENCES : links a publication with all the referenced publications.

The total number of relationships is 103708 relationships.

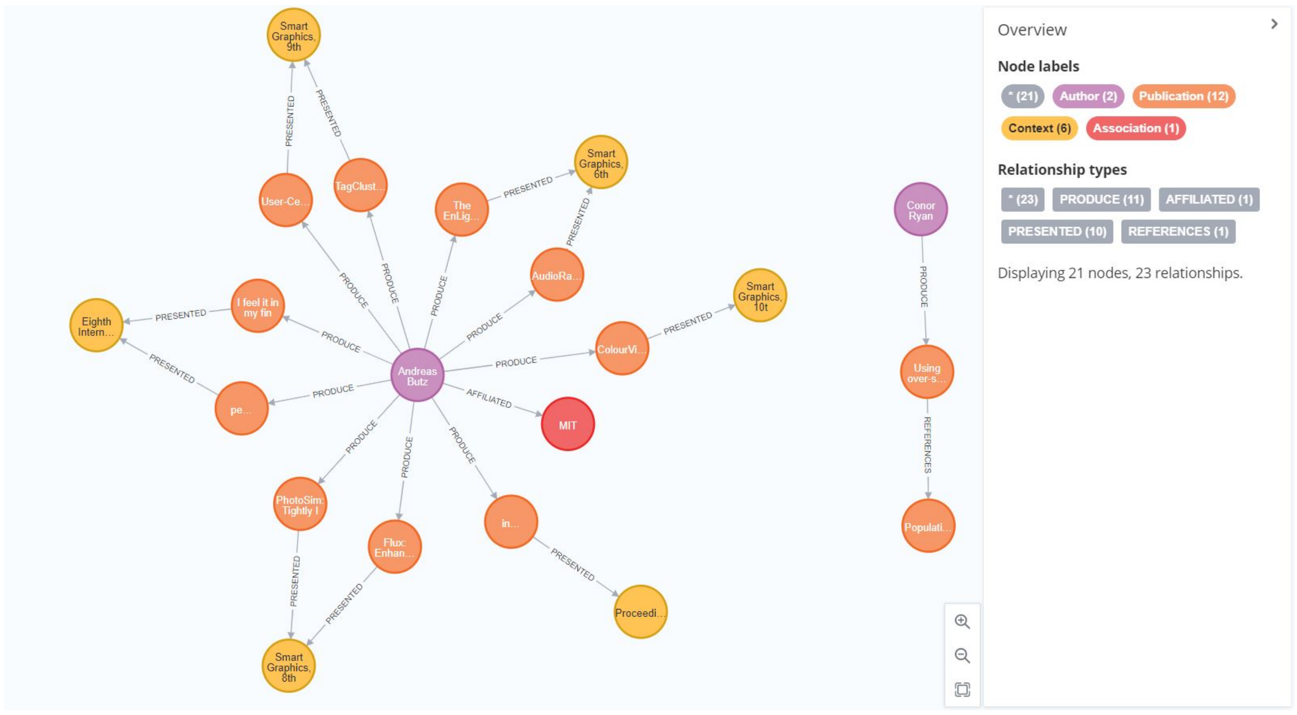


Figure 4.1: Graph Database snapshot

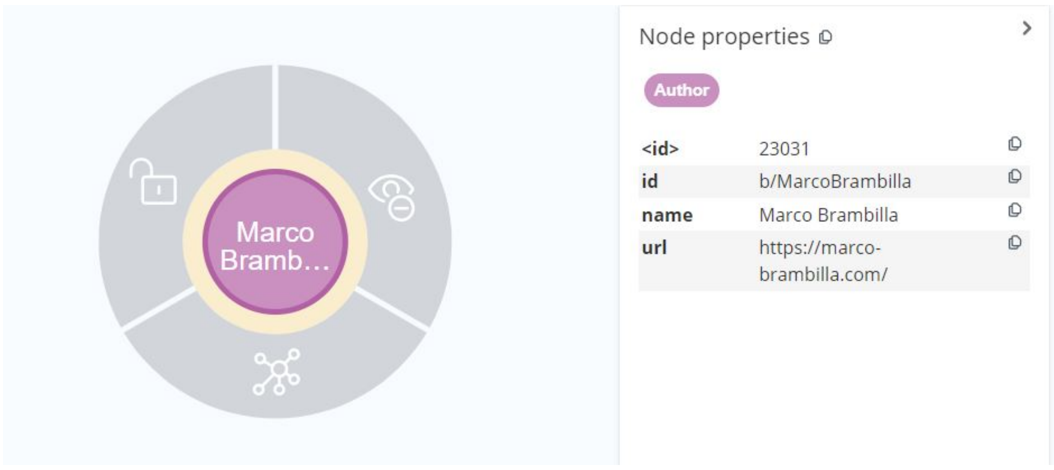
4.6. Creation/Update Commands

4.6.1. New author registration

An author registers to the platform.

Execution time : 2ms

```
CREATE (a:Author {
        id: "b/MarcoBrambilla",
        name: "Marco Brambilla",
        url: "https://marco-brambilla.com/"})
RETURN a
```



Node properties

Author

<id>	23031
id	b/MarcoBrambilla
name	Marco Brambilla
url	https://marco-brambilla.com/

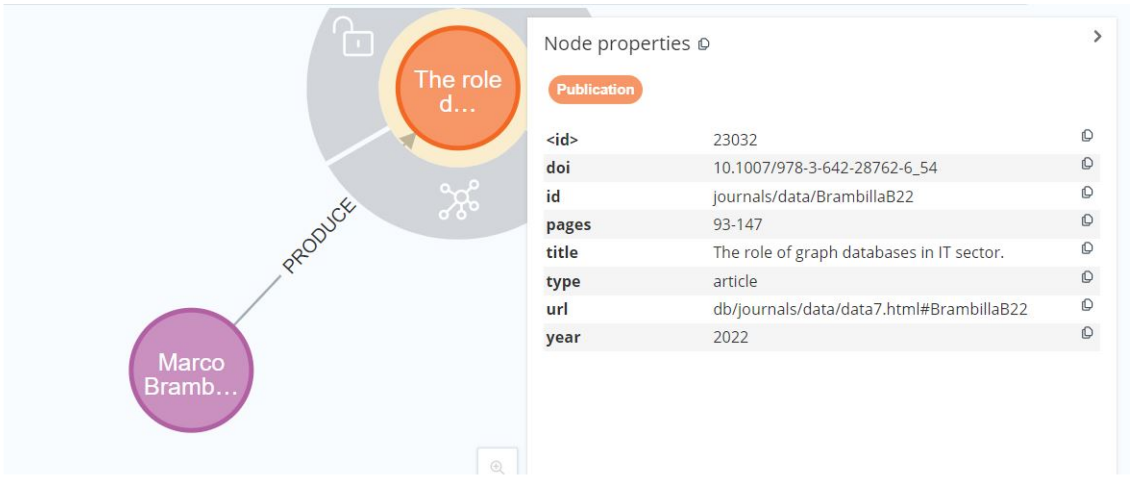
Figure 4.2: New author registration

4.6.2. New publication of an existing author

An author adds a publication: the Publication node is added and connected to the author.

Execution time : 3ms

```
MATCH (a:Author)
WHERE a.id = "b/MarcoBrambilla"
CREATE (p:Publication {
       id: "journals/data/BrambillaB22",
       title: "The role of graph databases in IT sector.",
       doi: "10.1007/978-3-642-28762-6_54",
       pages: "93-147",
       type: "article",
       url: "db/journals/data/data7.html#BrambillaB22",
       year: 2022})
CREATE (a)-[r:PRODUCE]->(p)
RETURN a,r,p
```



Node properties

Publication

<id>	23032
doi	10.1007/978-3-642-28762-6_54
id	journals/data/BrambillaB22
pages	93-147
title	The role of graph databases in IT sector.
type	article
url	db/journals/data/data7.html#BrambillaB22
year	2022

Figure 4.3: New publication

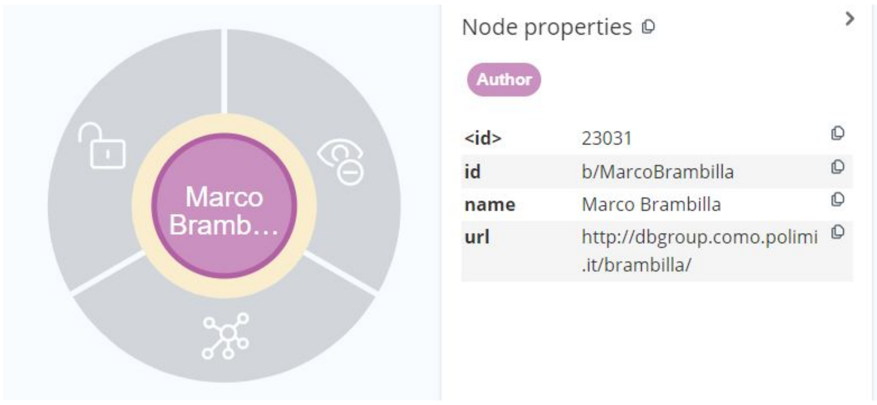
4.6.3. Update of an author's web page url

The author changes his website url: Edit the url field of the Author.

Note: this command can be used to add the website url to an existing author that doesn't already have one or to update the url field in case it already exists.

Execution time : 11ms

```
MATCH (a:Author)
WHERE a.id = "b/MarcoBrambilla"
SET a.url = "http://dbgroup.como.polimi.it/brambilla/"
RETURN a
```



Node properties

Author

<id>	23031
id	b/MarcoBrambilla
name	Marco Brambilla
url	http://dbgroup.como.polimi.it/brambilla/

Figure 4.4: Update author's web page url

4.6.4. Add a reference to a publication

Add the references of a Publication (using DOI).

Execution time : 55ms

```
MATCH (publication:Publication {
       doi: "10.1109/ISITA.2010.5649582"}), (
    referencedPublication:Publication {
       doi: "10.1007/978-3-319-08494-7_11"})
CREATE (publication)-[relation:REFERENCES]->(referencedPublication)
RETURN publication, relation, referencedPublication
```

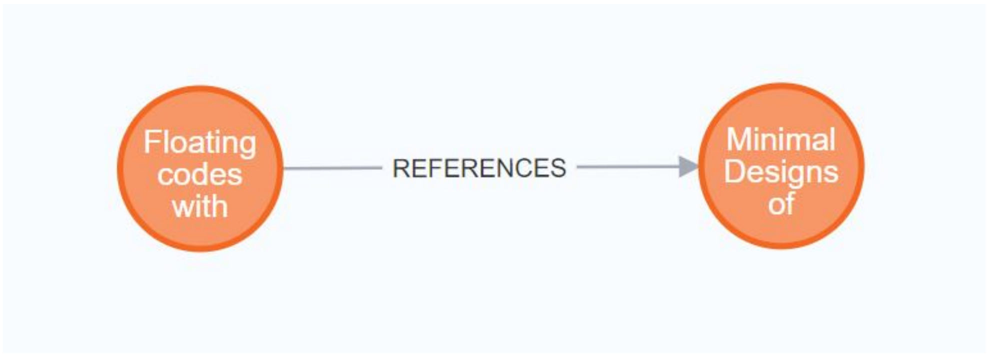


Figure 4.5: Add new reference

4.6.5. Delete an author profile

An author deletes his profile and all the relationships with the publications that he wrote.

Execution time : 4ms

```
MATCH (a:Author)
WHERE a.id = "s/AmitSinghal"
DETACH DELETE a
```

4.7. Queries

4.7.1. All co-authors starting from a name

Print all authors with whom a particular author has collaborated. *Execution time : 9ms*

```
MATCH (a:Author)-[r1:PRODUCE]->(p:Publication)<-[r2:PRODUCE]-(coauthor:Author)
WHERE a.id = "42/7075" and a <> coauthor
RETURN a, coauthor, p
```

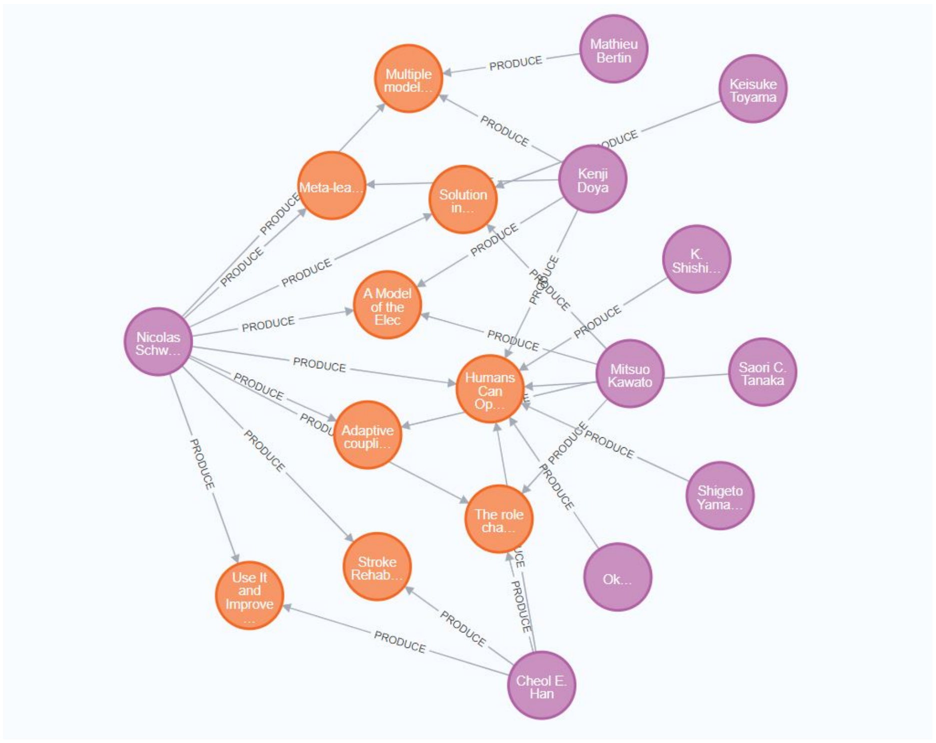


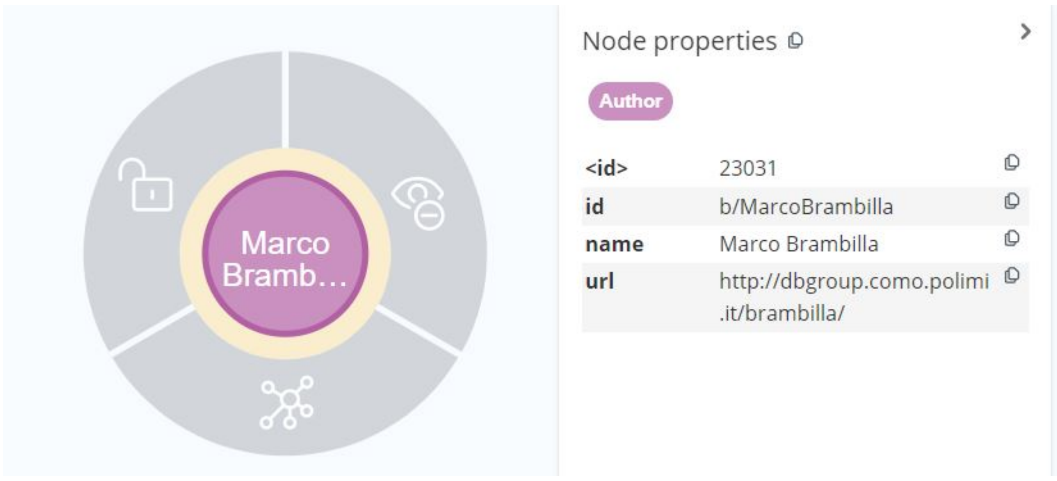
Figure 4.6: Co-authors from an author’s name

4.7.2. Get author by name

Search authors by name: all the homonyms will be printed.

Execution time : 7ms

```
MATCH (a:Author {name: "Marco Brambilla"})
RETURN a
```



Node properties

Author

<id>	23031
id	b/MarcoBrambilla
name	Marco Brambilla
url	http://dbgroup.como.polimi.it/brambilla/

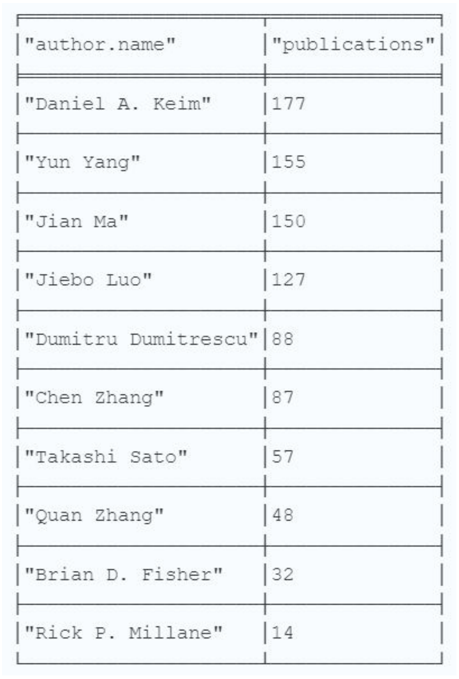
Figure 4.7: Get author by name

4.7.3. Most contributing author of an association

Get the top 10 of the most contributing authors affiliated with a given association, together with the number of publications that they wrote.

Execution time : 2ms

```
MATCH (author:Author)-[:PRODUCE]->(:Publication),(author)-[:AFFILIATED]->(association:
    Association)
WHERE association.id = "association/7"
WITH author, count(*) AS publications
ORDER BY publications DESC
LIMIT 10
RETURN author, publications
```



"author.name"	"publications"
"Daniel A. Keim"	177
"Yun Yang"	155
"Jian Ma"	150
"Jiebo Luo"	127
"Dumitru Dumitrescu"	88
"Chen Zhang"	87
"Takashi Sato"	57
"Quan Zhang"	48
"Brian D. Fisher"	32
"Rick P. Millane"	14

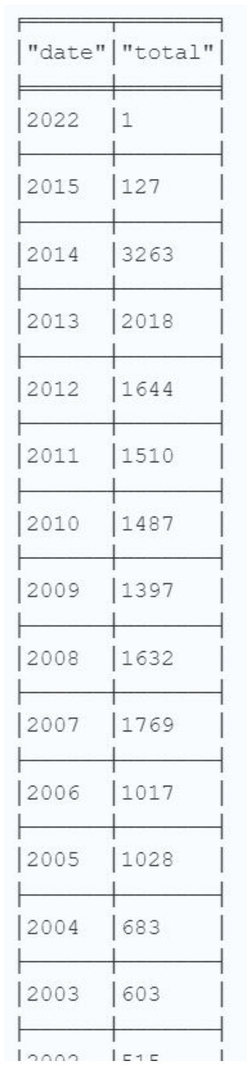
Figure 4.8: Most contributing authors of an association

4.7.4. Number of publications year by year

Get the total number of publications year by year, in descending order.

Execution time : 19ms

```
MATCH (p:Publication)
WITH p.year AS date, count(*) AS total
ORDER BY p.year DESC
RETURN date, total
```



```
┌───────┬───────┐
│"date" │"total"│
╞═══════╪═══════╡
│2022   │1      │
├───────┼───────┤
│2015   │127    │
├───────┼───────┤
│2014   │3263   │
├───────┼───────┤
│2013   │2018   │
├───────┼───────┤
│2012   │1644   │
├───────┼───────┤
│2011   │1510   │
├───────┼───────┤
│2010   │1487   │
├───────┼───────┤
│2009   │1397   │
├───────┼───────┤
│2008   │1632   │
├───────┼───────┤
│2007   │1769   │
├───────┼───────┤
│2006   │1017   │
├───────┼───────┤
│2005   │1028   │
├───────┼───────┤
│2004   │683    │
├───────┼───────┤
│2003   │603    │
├───────┼───────┤
```

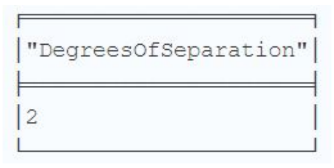
Figure 4.9: Number of publications year by year

4.7.5. Degrees of separation between two authors

Degrees of separation between two authors: let's call coauthors of a publication two author that have written together the same publication. Given two authors A and B, this query calculates the minimum number of intermediate coauthors to link A and B.

Execution time : 127ms

```
MATCH (a:Author {id:"25/4412"}), (b:Author {id:"73/2665"}),
     p = shortestPath((a)-[:PRODUCE*]-(b))
RETURN SIZE([n IN nodes(p) WHERE n:Author]) - 2 AS DegreesOfSeparation
```



```
"DegreesOfSeparation"
2
```

Figure 4.10: Degrees of separation between authors

Variant: Print all the authors that connects A to B, in this way if the author A wants to reach the author B, he can know who the intermediate authors are in order to contact them.

Execution time : 16ms

```
MATCH (a:Author {id:"25/4412"}), (b:Author {id:"73/2665"}),p = shortestPath((a)-[:
    PRODUCE*]-(b))
RETURN p
```

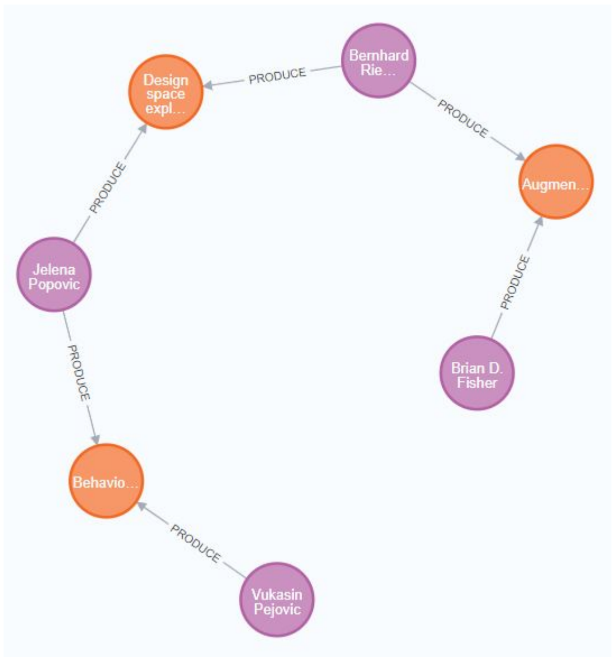


Figure 4.11: Variant : degrees of separation

4.7.6. Most referenced paper of the year

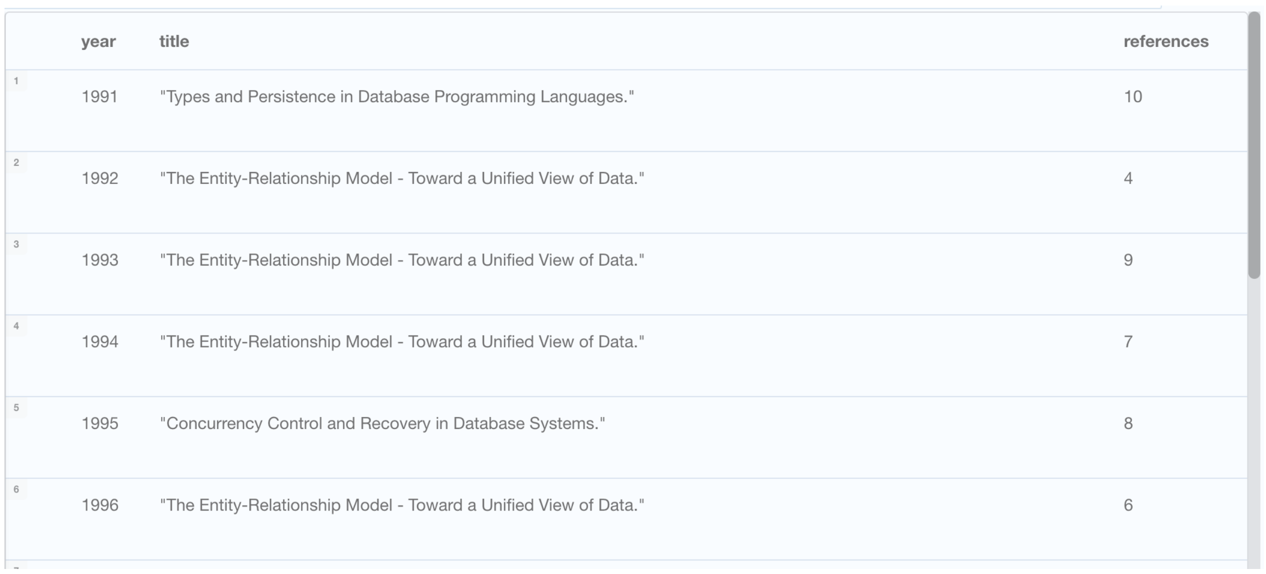
Get, for every year after 1990, the most referenced paper of the year.

Execution time : 43ms

```
MATCH (publication:Publication)-[:REFERENCES]->(referenced:Publication)
WHERE publication.year > 1990
WITH publication.year AS year, referenced AS referenced, count(*) AS numRef
WITH year AS year, max(numRef) AS maxNumRef

MATCH (pub:Publication)-[:REFERENCES]->(ref:Publication)
WHERE pub.year = year
WITH year AS year, ref AS referenced, count(*) AS numRef, maxNumRef AS maxNumRef
WHERE numRef = maxNumRef

RETURN year, head(collect(referenced.title)) AS title, maxNumRef AS references
ORDER BY year ASC
```



	year	title	references
1	1991	"Types and Persistence in Database Programming Languages."	10
2	1992	"The Entity-Relationship Model - Toward a Unified View of Data."	4
3	1993	"The Entity-Relationship Model - Toward a Unified View of Data."	9
4	1994	"The Entity-Relationship Model - Toward a Unified View of Data."	7
5	1995	"Concurrency Control and Recovery in Database Systems."	8
6	1996	"The Entity-Relationship Model - Toward a Unified View of Data."	6

Figure 4.12: Most referenced paper of the year

4.7.7. Shortest path of references from two publications

Given 2 publications find the shortest path between them in terms of references.

Execution time : 28ms

```
MATCH (a:Publication {id:"conf/ssdbm/ThomasH83"}),
     (b:Publication {id:"conf/vldb/TurnerHC79"}),
      p = shortestPath((a)-[:REFERENCES*]-(b))
RETURN p
```

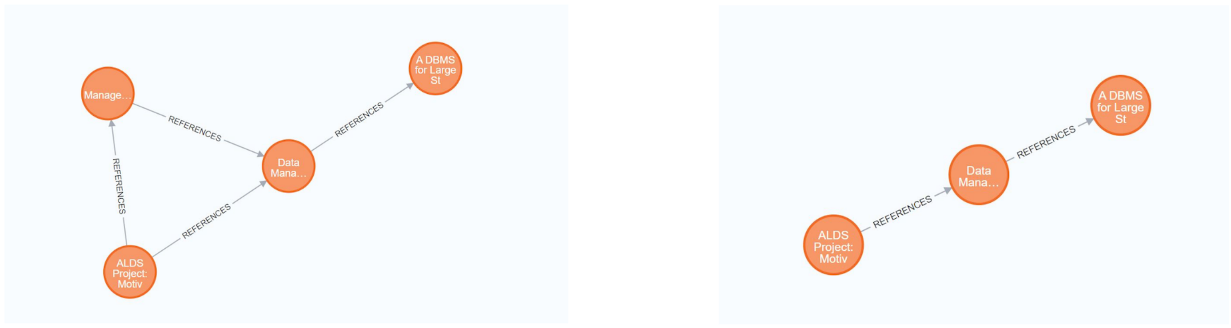


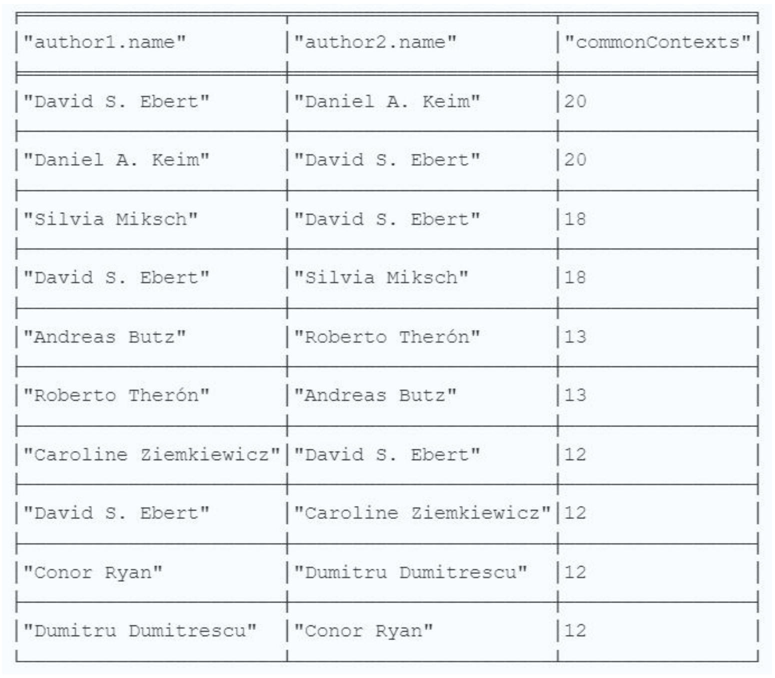
Figure 4.13: Shortest path of references from two publications

4.7.8. Couples with the highest number of publications in a Context

Find the 5 couples of authors with the highest number of publications in the same context of presentation (`JOURNAL` or `CONFERENCE`).

Execution time : 91ms

```
MATCH (author1:Author)-[:PRODUCE]->(:Publication)-[:PRESENTED]->(context:Context),
     (author2:Author)-[:PRODUCE]->(:Publication)-[:PRESENTED]->(context)
WHERE author1 <> author2
WITH author1 AS author1, author2 AS author2, count(context) AS commonContexts
RETURN author1.name, author2.name, commonContexts
ORDER BY commonContexts DESC
LIMIT 10
```

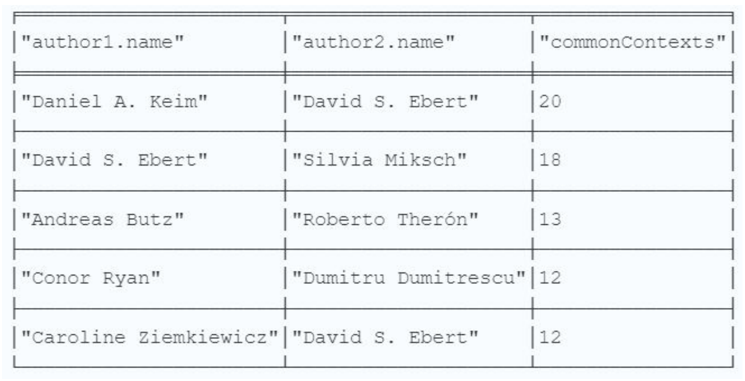


```
"author1.name"         "author2.name"         "commonContexts"
"David S. Ebert"       "Daniel A. Keim"       20
"Daniel A. Keim"       "David S. Ebert"       20
"Silvia Miksch"        "David S. Ebert"       18
"David S. Ebert"       "Silvia Miksch"        18
"Andreas Butz"         "Roberto Therón"       13
"Roberto Therón"       "Andreas Butz"         13
"Caroline Ziemkiewicz" "David S. Ebert"       12
"David S. Ebert"       "Caroline Ziemkiewicz" 12
"Conor Ryan"           "Dumitru Dumitrescu"   12
"Dumitru Dumitrescu"   "Conor Ryan"           12
```

Figure 4.14: Couples with the highest number of publications in an association

Variant: As we can see in the previous Figure, the query returns each couple two times, one with authors A and B, and one with swapped authors, B and A. To solve this issue a `DISTINCT` clause is not enough, because (A, B) is not equal to (B, A).
We solved this duplicates problem adding a condition `author1.name < author2.name` that ensures an ordering relation between all the authors.
Execution time : 47ms

```
MATCH (author1:Author)-[:PRODUCE]->(:Publication)-[:PRESENTED]->(context:Context),
     (author2:Author)-[:PRODUCE]->(:Publication)-[:PRESENTED]->(context)
WHERE author1 <> author2 and author1.name < author2.name
WITH author1 AS author1, author2 AS author2, count(context) AS commonContexts
RETURN author1.name, author2.name, commonContexts
ORDER BY commonContexts DESC
LIMIT 5
```



```
|"author1.name"        |"author2.name"       |"commonContexts"|
|"Daniel A. Keim"      |"David S. Ebert"     |20              |
|"David S. Ebert"      |"Silvia Miksch"      |18              |
|"Andreas Butz"        |"Roberto Therón"     |13              |
|"Conor Ryan"          |"Dumitru Dumitrescu"|12              |
|"Caroline Ziemkiewicz"|"David S. Ebert"     |12              |
```

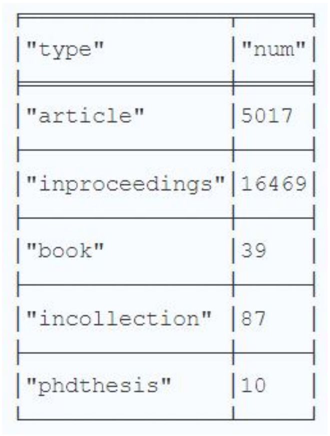
Figure 4.15: Variant : Couples with the highest number of publications in an association

4.7.9. Number of publication for each type

Get the number of publications divided for each type.

Execution time : 13ms

```
MATCH (a:Publication)
WITH a.type AS type, count(*) AS num
RETURN a, num
```



```
|"type"         |"num"|
|"article"      |5017 |
|"inproceedings"|16469|
|"book"         |39   |
|"incollection" |87   |
|"phdthesis"    |10   |
```

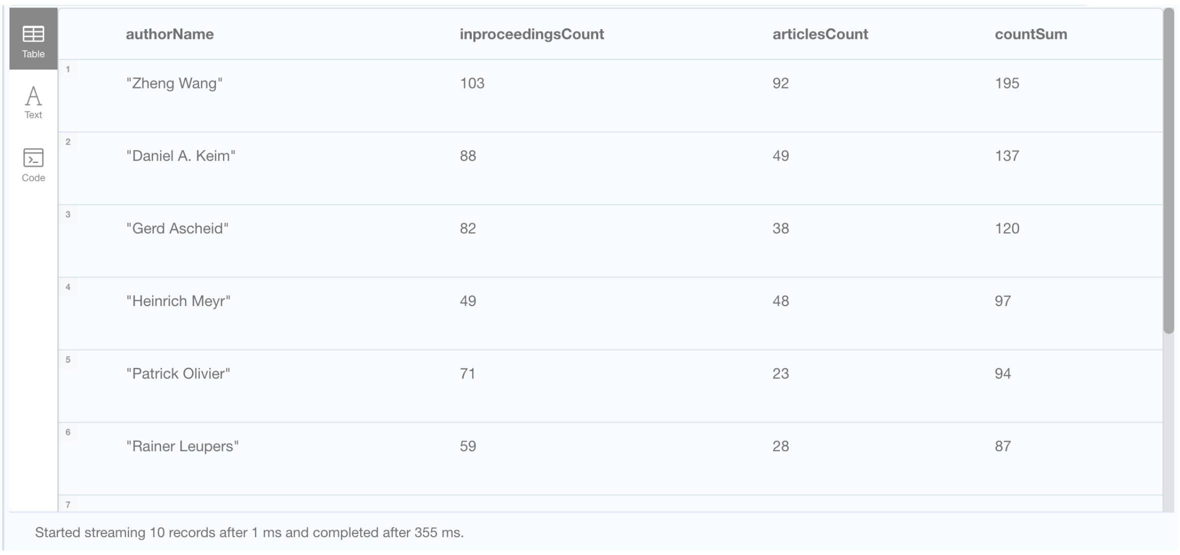
Figure 4.16: Number of publication for each type

4.7.10. Authors that have written at least one inproceeding and one article

Get all the authors that have written at least one inproceedings and one article, showing the respective count. Results are ordered by the sum of the two counts.

Execution time : 385ms

```
MATCH (author:Author)-[:PRODUCE]->(inproceedings:Publication),
     (author)-[:PRODUCE]->(article:Publication)
WHERE inproceedings.type = "inproceedings" AND article.type = "article"
WITH author.name AS authorName, count(distinct inproceedings.id) AS inproceedingsCount
    , count(distinct article.id) AS articlesCount
WITH authorName AS authorName, inproceedingsCount AS inproceedingsCount, articlesCount
     AS articlesCount,(inproceedingsCount + articlesCount) AS countSum
ORDER BY countSum DESC
RETURN authorName, inproceedingsCount, articlesCount, countSum
LIMIT 10
```



authorName	inproceedingsCount	articlesCount	countSum
"Zheng Wang"	103	92	195
"Daniel A. Keim"	88	49	137
"Gerd Ascheid"	82	38	120
"Heinrich Meyr"	49	48	97
"Patrick Olivier"	71	23	94
"Rainer Leupers"	59	28	87

Started streaming 10 records after 1 ms and completed after 355 ms.

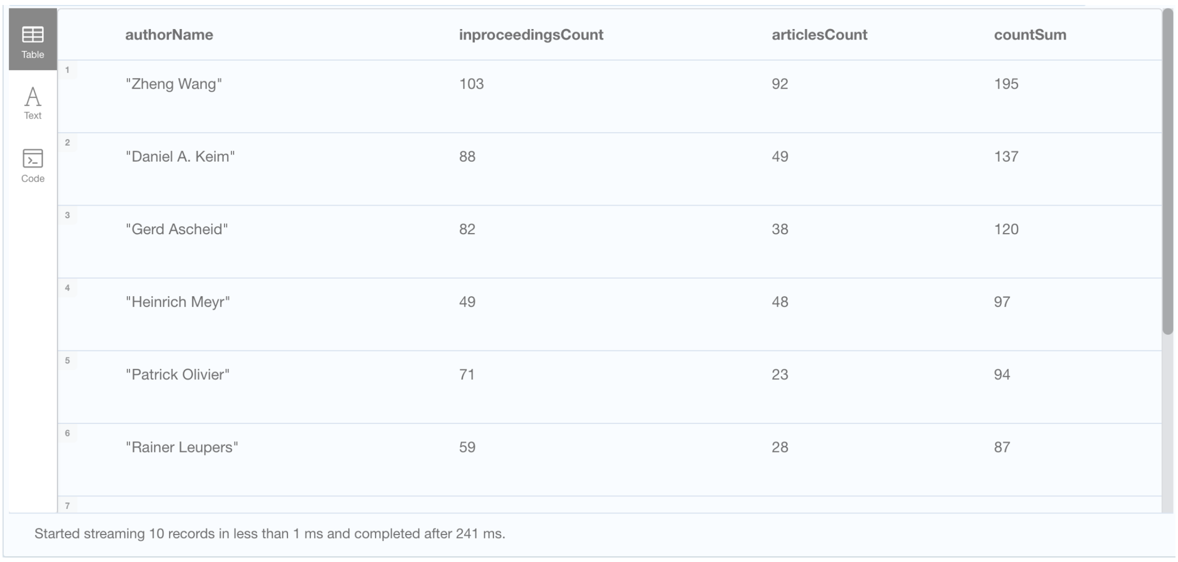
Figure 4.17: Authors that have written at least one incollection and one article, with respective counts, using Neo4j functions

Variant: the same result can be obtained using aggregations instead of functions, but with a shorter execution time.

Execution time : 241ms

```
MATCH (author:Author)-[:PRODUCE]->(inproceedings:Publication)
WHERE inproceedings.type = "inproceedings"
```

```
WITH author AS author, count(inproceedings) AS inproceedingsCount
MATCH (author)-[:PRODUCE]->(article:Publication{type:"article"})
WITH author.name AS authorName, count(article) as articlesCount,inproceedingsCount as
    inproceedingsCount
WITH (articlesCount + inproceedingsCount) AS countSum, authorName AS authorName,
    inproceedingsCount as inproceedingsCount, articlesCount AS articlesCount
ORDER BY countSum DESC
RETURN authorName, inproceedingsCount, articlesCount, countSum
LIMIT 10
```



	authorName	inproceedingsCount	articlesCount	countSum
1	"Zheng Wang"	103	92	195
2	"Daniel A. Keim"	88	49	137
3	"Gerd Ascheid"	82	38	120
4	"Heinrich Meyr"	49	48	97
5	"Patrick Olivier"	71	23	94
6	"Rainer Leupers"	59	28	87
7				

Started streaming 10 records in less than 1 ms and completed after 241 ms.

Figure 4.18: Authors that have written at least one incollection and one article, with respective counts, using aggregations

4.7.11. Top associations for number of "interesting" publication

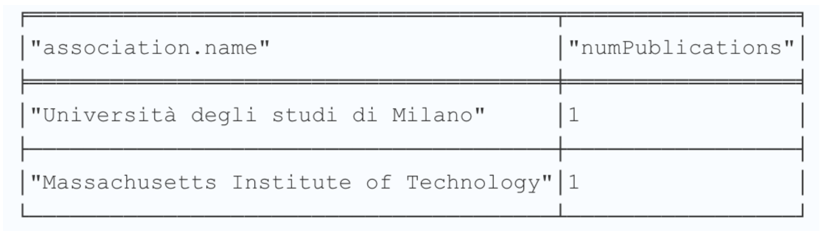
Get the top 3 associations for number of "interesting" publications that are presented in all the contexts (ex. conferences) that took place in a given year range (ex. 2000-2010). A publication is considered "interesting" if there are at least 3 publications referencing it.

Execution time : 13ms

```
MATCH (auth:Author)-[:AFFILIATED]->(assoc:Association),
     (auth)-[:PRODUCE]->(pub:Publication)-[:PRESENTED]->(context:Context),
     (otherPub:Publication)-[:REFERENCES]->(pub)
WHERE context.year > 2000 and context.year < 2010
WITH assoc AS assoc, pub AS pub, count(otherPub) AS numReferences
WHERE numReferences > 3
```

```
WITH assoc AS association, count(pub) AS numPublications
ORDER BY numPublications DESC
RETURN association.name, numPublications
LIMIT 3
```

Note: due to the dimension of the data-set this query returned only 2 results. In a real context with a complete data-set this query should return the 3 associations with the highest number of "interesting" publications.



"association.name"	"numPublications"
"Università degli studi di Milano"	1
"Massachusetts Institute of Technology"	1

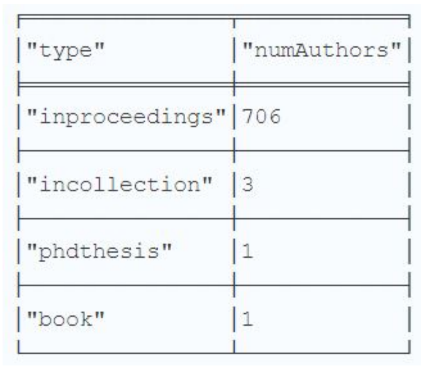
Figure 4.19: Top associations for number of "interesting" publications

4.7.12. Number of publications produced by the authors affiliated to a given association

Given an association, return the number of publications produced by the authors affiliated with that association, divided by type of publication.

Execution time : 3ms

```
MATCH (author:Author)-[:AFFILIATED]->(association:Association), (author)-[:PRODUCE]->(
    publication:Publication)
WHERE association.name = "Politecnico di Bari"
RETURN publication.type AS type, count(author) AS numAuthors
```



"type"	"numAuthors"
"inproceedings"	706
"incollection"	3
"phdthesis"	1
"book"	1

Figure 4.20: Number of publications produced by the authors associated to an association

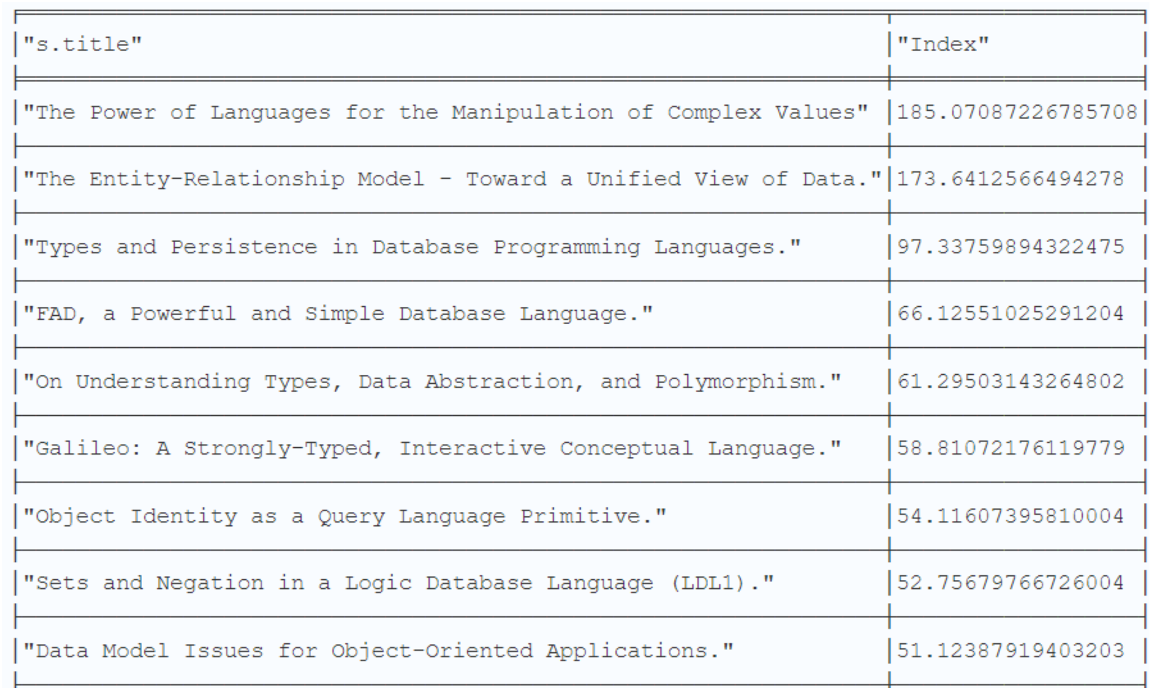
4.7.13. Ranking Algorithm

It returns a ranking of the Publications according to a certain measure of relevance that has been defined ad hoc for this purpose.
In particular the starting idea was to quantify the relevance of a certain publications counting the children of the relationship `REFERENCES*` , which is the transitive closure of the relationship `REFERENCES`. Starting from this point of view we implemented this measure by treating the counting operation as a weighted sum which penalized `REFERENCES*` relationship that are far in terms of year of publication. The heuristic behind this assumption is simple; if a lot of publications tends to transitively reference a certain publication in a short amount of time, then it is reasonable to assume that the influence of that publication has been important to determinate the results in the next papers, on the opposite when the `REFERENCES*` is between two publications that are far in terms of time then it is more likely that the relationship has not a very strong significance. In order to penalize distance in time a coefficient has been chosen (0.9) in order to describe the degradation of influence through time.

Execution time : 79 ms

```
MATCH (s:Publication)
MATCH (p:Publication)-[r:REFERENCES *]->(s:Publication)
RETURN   s, sum( 0.9^(-s.year + p.year) ) as Index
ORDER BY Index DESC
LIMIT 10
```

Showing only the title of the publication is shown for readability reason



"s.title"	"Index"
"The Power of Languages for the Manipulation of Complex Values"	185.07087226785708
"The Entity-Relationship Model - Toward a Unified View of Data."	173.6412566494278
"Types and Persistence in Database Programming Languages."	97.33759894322475
"FAD, a Powerful and Simple Database Language."	66.12551025291204
"On Understanding Types, Data Abstraction, and Polymorphism."	61.29503143264802
"Galileo: A Strongly-Typed, Interactive Conceptual Language."	58.81072176119779
"Object Identity as a Query Language Primitive."	54.11607395810004
"Sets and Negation in a Logic Database Language (LDL1)."	52.75679766726004
"Data Model Issues for Object-Oriented Applications."	51.12387919403203

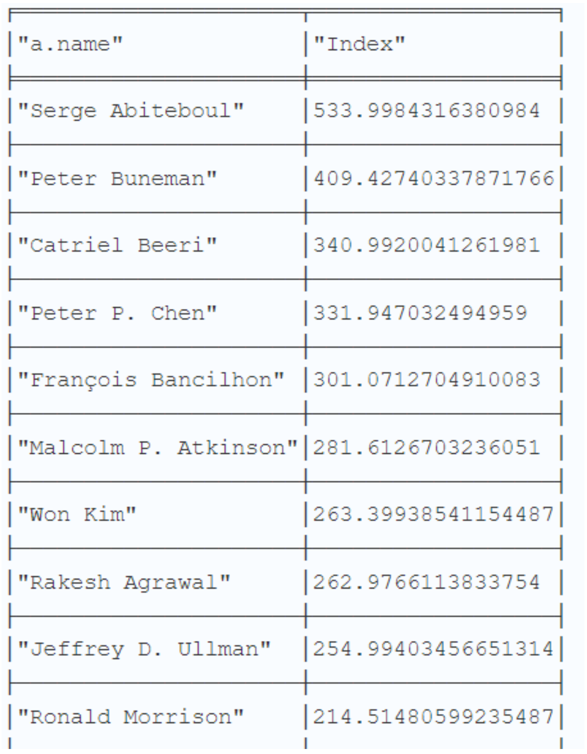
Figure 4.21: Ranking Algorithm

Variant: The same algorithm is proposed in order to rank author by relevance of their publications, summing along them.

Execution time : 181 ms

```
MATCH (p:Publication)-[r:REFERENCES *]->(s:Publication)<-[:PRODUCE]-(a:Author)
RETURN   a , sum( 0.9^(-s.year + p.year) ) as Index
ORDER BY Index DESC
LIMIT 10
```

Showing only the name of the author is shown for readability reason.



```
╒══════════════════════╤═════════════════╕
│"a.name"              │"Index"          │
╞══════════════════════╪═════════════════╡
│"Serge Abiteboul"     │533.9984316380984 │
├──────────────────────┼─────────────────┤
│"Peter Buneman"       │409.42740337871766│
├──────────────────────┼─────────────────┤
│"Catriel Beeri"       │340.9920041261981 │
├──────────────────────┼─────────────────┤
│"Peter P. Chen"       │331.947032494959  │
├──────────────────────┼─────────────────┤
│"François Bancilhon"  │301.0712704910083 │
├──────────────────────┼─────────────────┤
│"Malcolm P. Atkinson" │281.6126703236051 │
├──────────────────────┼─────────────────┤
│"Won Kim"             │263.39938541154487│
├──────────────────────┼─────────────────┤
│"Rakesh Agrawal"      │262.9766113833754 │
├──────────────────────┼─────────────────┤
│"Jeffrey D. Ullman"   │254.99403456651314│
├──────────────────────┼─────────────────┤
│"Ronald Morrison"     │214.51480599235487│
```

Figure 4.22: Ranking Algorithm for Authors

5 | MongoDB Delivery

5.1. Abstract

The objective of this delivery is to design a *documental database* using the **MongoDB** technology for storing Scientific Papers. In particular this technology easily allow us to store all the information of the previously called `PUBLICATION`, including the entire content of the document and related metadata (its sections, subsections, figures and so on).

5.2. Database description

The focus of this delivery is on the entity `PUBLICATION`, that in this chapter will be called `SCIENTIFIC PAPER` (for coherence with respect to the delivery's specifications). For this reason we needed to add some other attributes to the initially designed entity of the ER model.

The following sections explain the document structure of a `SCIENTIFIC PAPER` and the differences/improvements we had to do in order to exploit most of the data-set used in the first delivery.

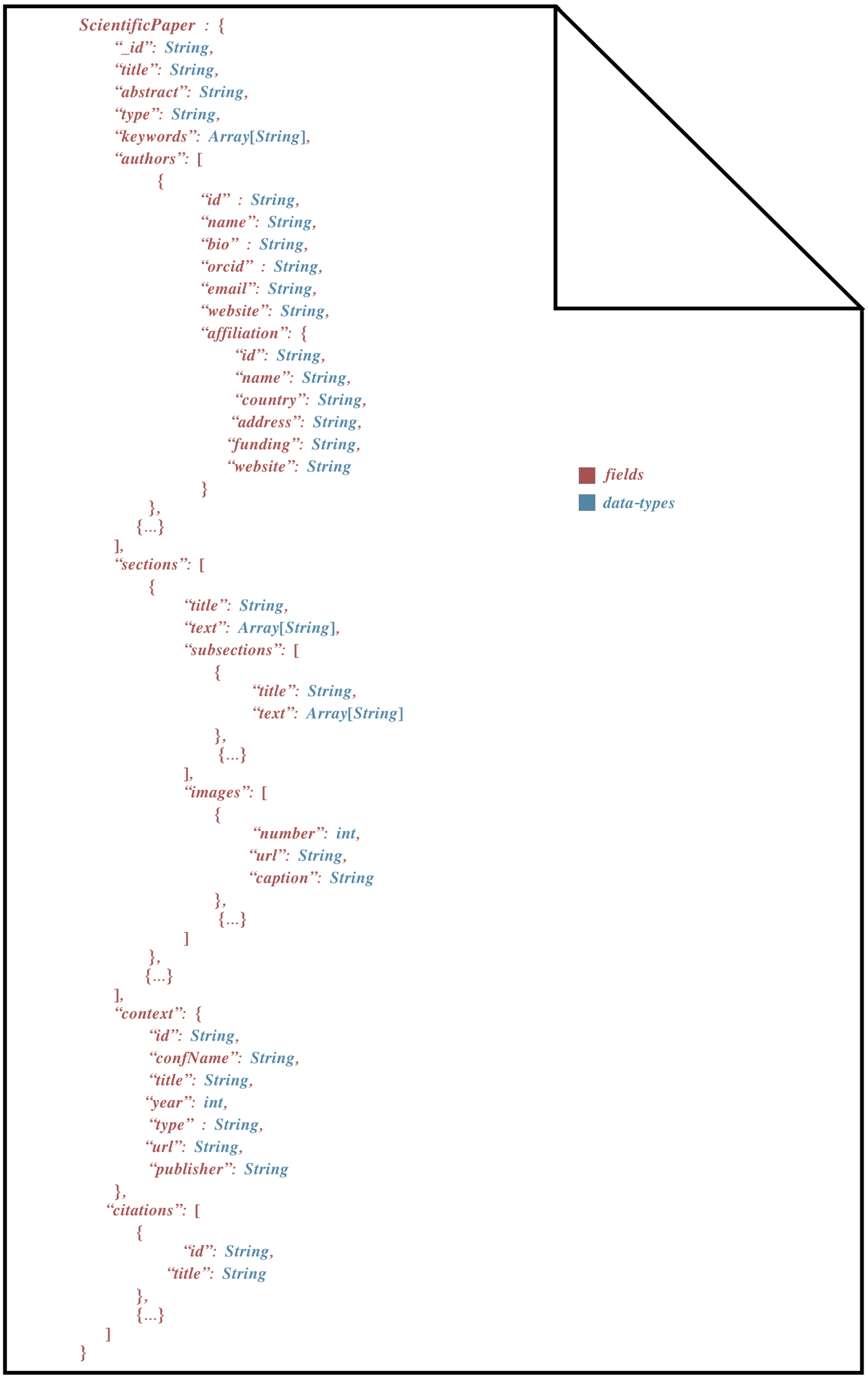
5.2.1. Document Structure

This sections explain the Scientific Paper document's structure and the structure of all the sub-documents contained in it.
We want to underline that since this delivery want to put the emphasis on *documental databases*, we implement only the collection **Scientific Papers**.

The idea behind the structure that we gave to our JSON document is based on the objective of optimizing the per-query complexity; having only one collection builds an optimal situation in terms of the amount of necessary look-up operations. In fact, the look-up operation would be inherently necessary whenever the user requests information regarding different collections; reducing the number of collections, the complexity of every query decreases in favor of an increment of redundancy in the data.

A deep analysis was performed in order to choose the optimal compromise between the two options, in particular our focus was manly on the possibility of dividing the set of authors from the publication collection: by one hand separating authors from publications would make the structure of the data-set simpler, but at the same time it would increase the complexity required for a large set of basic operations (moreover, keeping these two entities in the same collection is also a way to highlight the deep relationship between these two concepts, which may be useful in further developments).

In this trade-off framework we decided to choose for an extreme design strategy and we keep only one collection: this solution is based on the nature of the data, that tends to accumulate the most of its information (in terms of amount of content) inside the sections, keeping a shallow structure in the entities that represent other objects. This choice implies that all the other entities connected to the Scientific Paper are embedded in the document: apart from increasing the efficiency of the queries, this choice helped us to really understand the differences between documental technologies and relational-oriented technologies, and how these different ways of storing data can be most suitable for design a situation/model instead of another one.
The only reference we used is related to the `citations` that, as can be seen in the figure below, contains the primary-key of other Scientific Paper documents in order to retrieve them by reference. We introduced the field `title` in order to avoid the look-up in case we only need to retrieve the title of the cited paper.



```
ScientificPaper : {
    "_id": String,
    "title": String,
    "abstract": String,
    "type": String,
    "keywords": Array[String],
    "authors": [
        {
            "id" : String,
            "name": String,
            "bio" : String,
            "orcid" : String,
            "email": String,
            "website": String,
            "affiliation": {
                "id": String,
                "name": String,
                "country": String,
                "address": String,
                "funding": String,
                "website": String
            }
        },
        {...}
    ],
    "sections": [
        {
            "title": String,
            "text": Array[String],
            "subsections": [
                {
                    "title": String,
                    "text": Array[String]
                },
                {...}
            ],
            "images": [
                {
                    "number": int,
                    "url": String,
                    "caption": String
                },
                {...}
            ]
        },
        {...}
    ],
    "context": {
        "id": String,
        "confName": String,
        "title": String,
        "year": int,
        "type" : String,
        "url": String,
        "publisher": String
    },
    "citations": [
        {
            "id": String,
            "title": String
        },
        {...}
    ]
}
```

Figure 5.1: Scientific Paper Document Structure

5.2.2. Adjustments from the ER model

As explained by the Document Structure above, we inserted new attributes to the entity `SCIENTIFIC PAPER` (i.e., ex `PUBLICATION`) (e.g section, subsection, figures, abstract, etc..) and we implemented them just for this delivery because they are strictly related to the document itself. The other entities implemented are easily recognizable in the Document Structure (same field/entity-name) and the relationships between them are embedded in the Scientific Paper's fields, except for the citations which are directly referenced by the id of the cited Scientific Paper documents.
The following table lists all the new attributes inserted for this delivery:

ER Model Entity	**Implemented Entity**
AUTHOR	All the attributes are implemented in the sub-documents embedded in the field authors
ASSOCIATION	All the attributes are implemented in the sub-sub-document embedded in the field association of the sub-documents in authors
CONTEXT	All the attributes are implemented in the sub-document embedded in the field context
PUBLICATION / SCIENTIFIC PAPER	The designed attributes for this entity in the original ER model do not cover all the specifications for this delivery, so we inserted: • _id : is the unique identifier of the document SCIENTIFIC PAPER in the collection of Scientific Papers. • title : is the title of the SCIENTIFIC PAPER. • keywords : is a list of keywords related to the title. • abstract : is the abstract of the SCIENTIFIC PAPER. • type : is used to implement the ISA of the PUBLICATION. • authors : is a list of sub-documents containing the information about the authors of the SCIENTIFIC PAPER (it represents the relationship PRODUCE). It also include the information of the authors' ASSOCIATION in the nested field affiliation. • sections : is a list of section's sub-document containing: – subsections : is a list of sub-section's sub-document containing: ∗ title : is the sub-section's title, ∗ text : is an array of the sub-section's paragraphs, – images : is a list of image's sub-document containing: ∗ number : is the progressive number of the image, ∗ url : is the image's url reference, ∗ caption : is the textual caption of the image, • context : is the sub-document containing the information about the context (it represents the relationship PRESENTED). • citations : is a list of sub-documents containing the id and the title of each paper which was cited by the SCIENTIFIC PAPER (it represents the relationship REFERENCES).

5.2.3. JSON generation

The JSON file containing the whole data-set imported in MongoDB has been generated with a Python script that parses the CSV files generated for the first delivery (as explained in section 3.1) to retrieve the relevant information concerning a publication and that generates randomly all the information about the content of that publication, such as the number of chapters, the number of sections in a specific chapter, the title and the actual text of these sections (in particular, the text has been generated selecting random terms from a pool of synonyms of the title, in order to preserve a minimal semantic coherence with the actual publication) and the number of figures in a section. At each figure we have just assigned an incremental number, an URL of the actual image of the format `images/publication_id/num_section/num_figure` and a randomly generated caption.
For each author of whom we did not have the email in the data-set we have generated one starting from the its name and its association.

5.3. Data-set upload and database creation

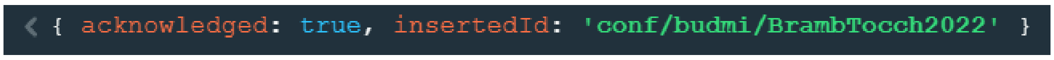
5.3.1. Publications collection loading

To upload the data-set into our MongoDB database we have simply imported the file `publications.json` generated as explained in section 5.2.3 using the GUI of MongoDB Compass.

5.4. Creation/Update Commands

5.4.1. New Scientific Paper insertion

```
db.ScientificPapers.insertOne({
    "_id": "conf/budmi/BrambTocch2022",
    "title": "Systems and Methods for Big and Unstructred Data",
    "type": "inproceedings",
    "keywords": ["Systems","Methods","Big","Unstructured","Data"],
    "authors" : [
        {
            "id": "b/MarcoBrambilla",
            "orcid": "0000-0002-8753-2434",
            "name": "Marco Brambilla",
            "bio": "Professor at Politecnico di Milano",
            "email": "marco.brambilla@polimi.it",
            "website": "http://home.dei.polimi.it/mbrambil/"
        },
        {
            "id": "281/3807",
            "name": "Andrea Tocchetti",
            "bio": "PhD candidate at Politecnico di Milano",
            "email": "andrea.tocchetti@polimi.it",
            "website": "https://www.deib.polimi.it/ita/personale/dettagli/1054414"
        }
    ],
    "sections": [
        {
            "title": "Delivery 1:  graph",
            "text": "This chapter is about Neo4j",
            "subsection": [
                {
                    "title": "The importance of graph representation",
                    "text": "We could say a lot about this topic..."
                }
            ],
            "images": []
        },
        {
            "title": "Delivery 2:  documental",
            "text": "This chapter is about MongoDB",
            "subsection": [
                {
                    "title": "The importance of documental databases",
                    "text": "The main aspect about documental database is..."
                }
            ],
            "images": []
        }
    ],
    "context": [
        {
            "id": "conf/BUDMi/2022",
            "confName": "BUDMi",
            "title": "Conference on Big and Unstructured Data, Milano, 2022",
            "type": "conf",
            "year": 2022,
            "url": "www.polimi-press.it/BUD/2022",
            "publisher": "PoliMi Press"
        }
    ],
    "citations": [
        {
            "id": "conf/icde/Bertino91" ,
            "title": "An Indexing Technique for Object-Oriented Databases."
        }
    ]
})
```



```
< { acknowledged: true, insertedId: 'conf/budmi/BrambTocch2022' }
```

Figure 5.2: New Scientific Paper insertion

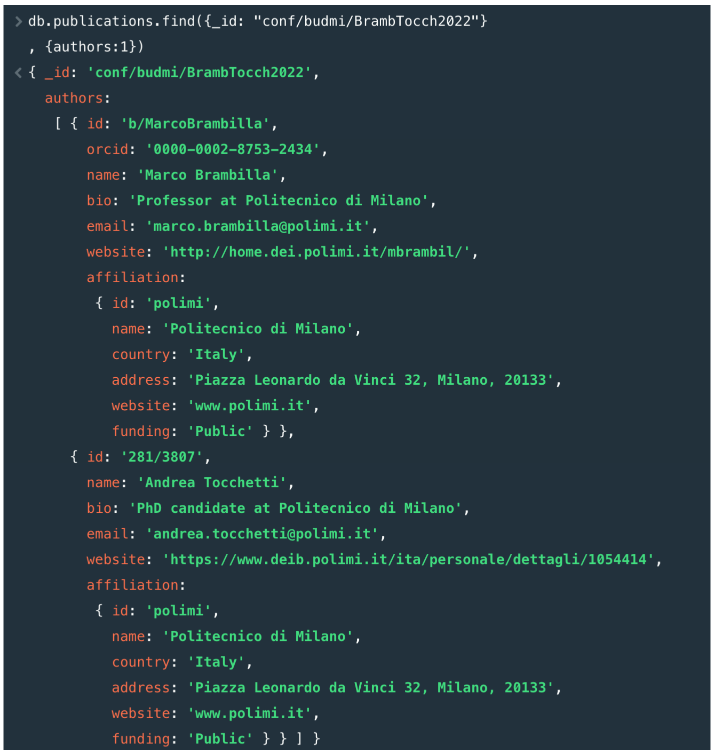
5.4.2. Adding the affiliation for the authors of the latter inserted publication

Since the the authors of the latter inserted publication are both affiliated with *"Politecnico di Milano"* we can use the following commands:

```
db.ScientificPapers.updateMany({
  "authors.id": "b/MarcoBrambilla"
},
{
  $set: {
    "authors.$.affiliation": {
      "id": "polimi",
      "name": "Politecnico di Milano",
      "country": "Italy",
      "address": "Piazza Leonardo da Vinci 32, Milano, 20133",
      "website": "www.polimi.it",
      "funding": "public"
    }
  }
})

db.ScientificPapers.updateMany({
  "authors.id": "281/3807"
},
{
  $set: {
    "authors.$.affiliation": {
        "id": "polimi",
        "name": "Politecnico di Milano",
        "country": "Italy",
        "address": "Piazza Leonardo da Vinci 32, Milano, 20133",
        "website": "www.polimi.it",
        "funding": "public"
      }
    }
})
```

The result is that in the latter inserted paper we now have information on the affiliation of the two authors. Indeed:



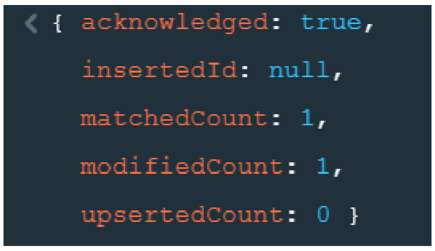
```
> db.publications.find({_id: "conf/budmi/BrambTocch2022"}
  , {authors:1})
< { _id: 'conf/budmi/BrambTocch2022',
    authors:
     [ { id: 'b/MarcoBrambilla',
         orcid: '0000-0002-8753-2434',
         name: 'Marco Brambilla',
         bio: 'Professor at Politecnico di Milano',
         email: 'marco.brambilla@polimi.it',
         website: 'http://home.dei.polimi.it/mbrambil/',
         affiliation:
          { id: 'polimi',
            name: 'Politecnico di Milano',
            country: 'Italy',
            address: 'Piazza Leonardo da Vinci 32, Milano, 20133',
            website: 'www.polimi.it',
            funding: 'Public' } },
       { id: '281/3807',
         name: 'Andrea Tocchetti',
         bio: 'PhD candidate at Politecnico di Milano',
         email: 'andrea.tocchetti@polimi.it',
         website: 'https://www.deib.polimi.it/ita/personale/dettagli/1054414',
         affiliation:
          { id: 'polimi',
            name: 'Politecnico di Milano',
            country: 'Italy',
            address: 'Piazza Leonardo da Vinci 32, Milano, 20133',
            website: 'www.polimi.it',
            funding: 'Public' } } ] }
```

Figure 5.3: Adding the affiliation field of the authors of an already existing paper

5.4.3. Update citations of an existing Scientific Paper

Update the list of citations of a Scientific Paper with a new citation.

```
db.ScientificPapers.updateOne(
   { "_id": "journals/ac/BinkleyG96" },
   { "$push" : { "citations": { "_id": "conf/MMdada/33", "title": "The best book of the
       world" }}}
)
```

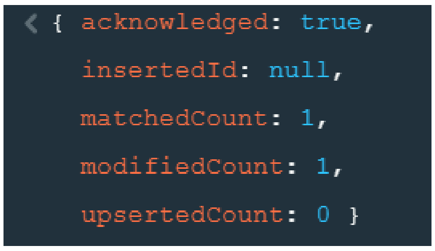


```
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
```

Figure 5.4: Update citations of an existing Scientific Paper

5.4.4. Update a document with new page numbers

```
db.ScientificPapers.updateOne(
    { "title" : "An Indexing Technique for Object-Oriented Databases." },
    { "$set" : { "num_pages" : "160-175" }}
)
```



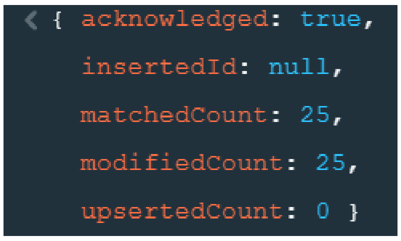
```
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
```

Figure 5.5: Update a document with new page numbers

5.4.5. Adding the ORCID and the website URL to an author given his ID

Adding the ORCID and the website URL of the author with the id "92/476".

```
db.ScientificPapers.updateMany(
  { "authors.id" : "92/476"},
  { "$set" : { "authors.$.orcid" : "0000-0002-5118-7835", "authors.$.website" : "
      www.sunder.lal.eng.edu/home"}}
)
```



```
< { acknowledged: true,
    insertedId: null,
    matchedCount: 25,
    modifiedCount: 25,
    upsertedCount: 0 }
```

Figure 5.6: Adding the ORCID and the website URL to an author given his ID

5.5. Queries

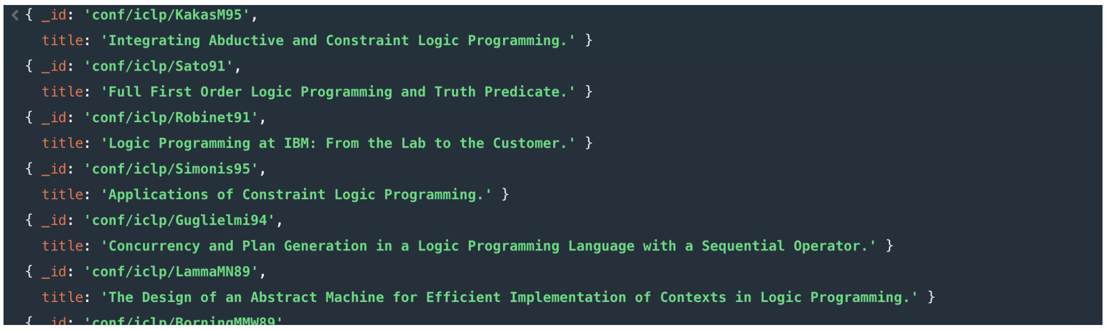
All the queries were performed using the shell `mongosh` on MongoDB Compass. The results of the queries are shown as screenshots of `mongosh`'s output. The images don't contain the entire result of the query but only the most important parts that are enough to verify that the query work as expected.

Note: *Some of the queries we wrote are parametric (e.g., in "Find all the publications of a given author" the author is the parameter). To highlight this fact and for reading purposes, the parameters of the queries are in magenta color.*

5.5.1. Titles of inproceedings presented at a certain conference by a given publisher

Find the titles of the Scientific Papers which type is 'inproceeding' and that were presented in the conference 'ICLP' hold by 'MIT Press'. To slightly increase the complexity of this query, in the title of the selected papers there will be the word 'DBMS' or 'Programming'.

```
db.ScientificPapers.find(
    { "$and" : [
        { "type" : "inproceedings"},
        { "context.confName" : "ICLP"},
        { "context.publisher" : "MIT Press"},
        { "$or" : [
            { "title" : {$regex:"DMBS"}},
            { "title" : {$regex:"Programming"}}
        ]}
    ]},
    { "title" : 1}
)
```



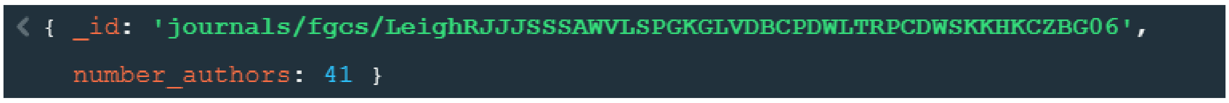
```
< { _id: 'conf/iclp/KakasM95',
    title: 'Integrating Abductive and Constraint Logic Programming.' }
  { _id: 'conf/iclp/Sato91',
    title: 'Full First Order Logic Programming and Truth Predicate.' }
  { _id: 'conf/iclp/Robinet91',
    title: 'Logic Programming at IBM: From the Lab to the Customer.' }
  { _id: 'conf/iclp/Simonis95',
    title: 'Applications of Constraint Logic Programming.' }
  { _id: 'conf/iclp/Guglielmi94',
    title: 'Concurrency and Plan Generation in a Logic Programming Language with a Sequential Operator.' }
  { _id: 'conf/iclp/LammaMN89',
    title: 'The Design of an Abstract Machine for Efficient Implementation of Contexts in Logic Programming.' }
  { _id: 'conf/iclp/BorningMMW89',
```

Figure 5.7: Titles of inproceedings presented at a certain conference by a given publisher

5.5.2. Scientific Paper with the highest number of authors

Find the Scientific Paper with the highest number of authors present in the database.

```
db.ScientificPapers.aggregate([
    { "$project": { "id": true, "number_authors" : { "$size" : "$authors"}}},
    { "$sort" : {"number_authors" : -1}},
    { "$limit" : 1}
])
```



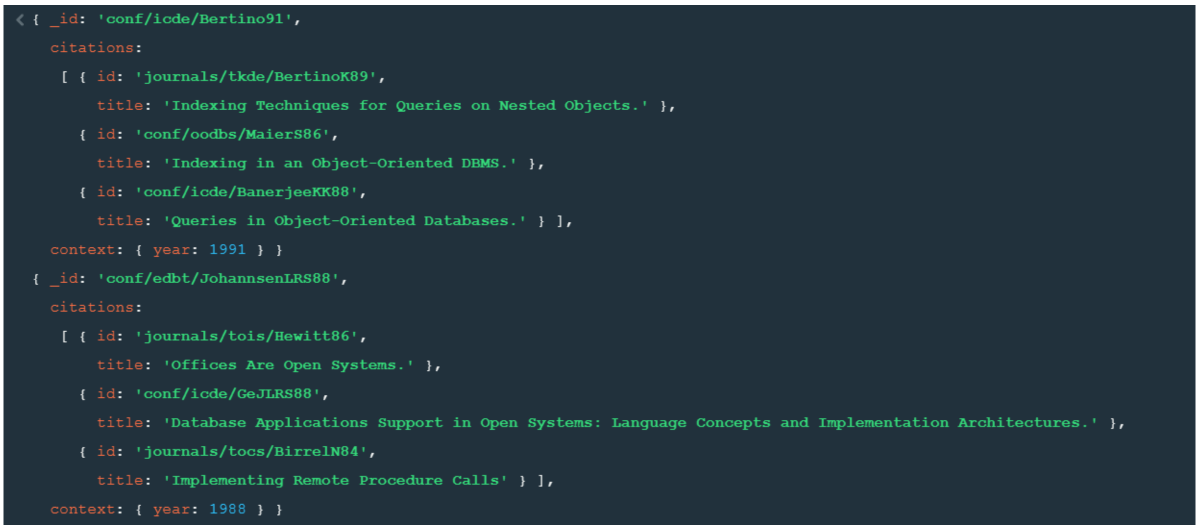
```
< { _id: 'journals/fgcs/LeighRJJJSSSAWVLSPGKGLVDBCPDWLTRPCDWSKKHKCZBG06',
    number_authors: 41 }
```

Figure 5.8: Scientific Paper with the highest number of authors

5.5.3. Scientific Papers with at least k-citations

Find the id, the title, the citations and the context year of Scientific Papers with the context year lower than 1999 and with at least 3 citations.

```
db.ScientificPapers.find(
    { "$and" : [
        { "context.year" : { "$lt" : 1999}},
        { "citations":{"$size" : 3}}
    ]},
    { "id" : 1, "title" : 1, "context.year" : 1, "citations" : 1}
)
```



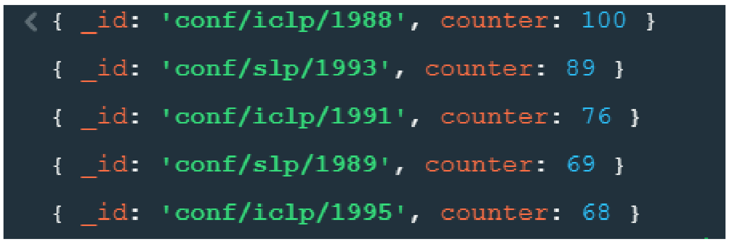
```
< { _id: 'conf/icde/Bertino91',
    citations:
     [ { id: 'journals/tkde/BertinoK89',
         title: 'Indexing Techniques for Queries on Nested Objects.' },
       { id: 'conf/oodbs/MaierS86',
         title: 'Indexing in an Object-Oriented DBMS.' },
       { id: 'conf/icde/BanerjeeKK88',
         title: 'Queries in Object-Oriented Databases.' } ],
    context: { year: 1991 } }
  { _id: 'conf/edbt/JohannsenLRS88',
    citations:
     [ { id: 'journals/tois/Hewitt86',
         title: 'Offices Are Open Systems.' },
       { id: 'conf/icde/GeJLRS88',
         title: 'Database Applications Support in Open Systems: Language Concepts and Implementation Architectures.' },
       { id: 'journals/tocs/BirrelN84',
         title: 'Implementing Remote Procedure Calls' } ],
    context: { year: 1988 } }
```

Figure 5.9: Scientific Papers with at least k-citations

5.5.4. Top-k conferences of a certain publisher held before a certain year by number of presented papers

Find the top 5 conferences held after 1990 by MIT Press ordered by the number of presented Scientific Papers.

```
db.ScientificPapers.aggregate([
    { "$match" : {
        "context.year" : { "$gt" : 1990},
        "context.publisher" : "MIT Press"
    }},
    { "$group" : {
        "_id" : "$context.id",
        counter : { $sum : 1}
    }},
    { "$sort" : { counter : -1}},
    { "$limit" : 5}
])
```



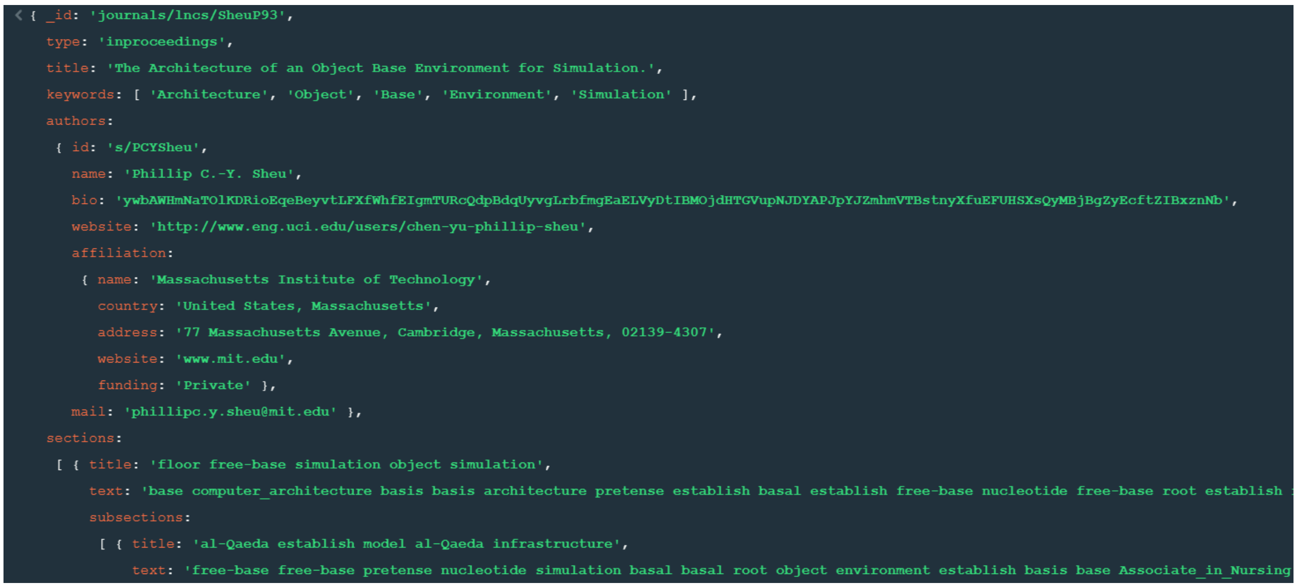
```
{ _id: 'conf/iclp/1988', counter: 100 }
{ _id: 'conf/slp/1993', counter: 89 }
{ _id: 'conf/iclp/1991', counter: 76 }
{ _id: 'conf/slp/1989', counter: 69 }
{ _id: 'conf/iclp/1995', counter: 68 }
```

Figure 5.10: Top k-conferences before a certain year

5.5.5. Documents of a certain author contextualized after a certain year

Find the Scientific Papers written by *Phillip C.-Y. Sheu* whit the context year greater than 2000.

```
db.ScientificPapers.aggregate([
    { "$unwind" : "$authors"},
    { "$match" : {
        "authors.name" : "Phillip C.-Y. Sheu",
        "context.year" : { "$gt" : 2000}
    }}
])
```



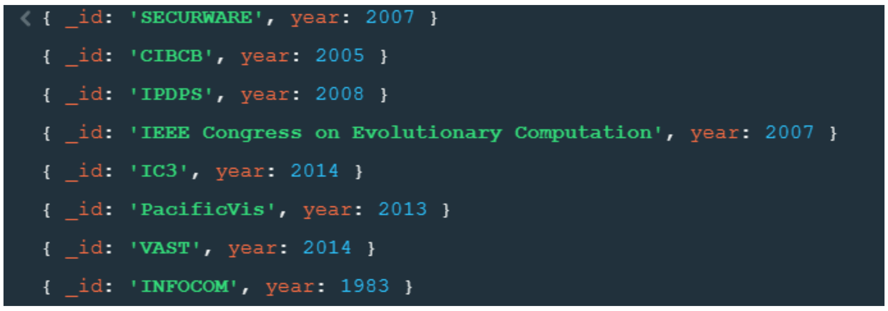
```
{ _id: 'journals/lncs/SheuP93',
  type: 'inproceedings',
  title: 'The Architecture of an Object Base Environment for Simulation.',
  keywords: [ 'Architecture', 'Object', 'Base', 'Environment', 'Simulation' ],
  authors:
   { id: 's/PCYSheu',
     name: 'Phillip C.-Y. Sheu',
     bio: 'ywbAWHmNaTOlKDRioEqeBeyvtLFXfWhfEIgmTURcQdpBdqUyvgLrbfmgEaELVyDtIBMOjdHTGVupNJDYAPJpYJZmhmVTBstnyXfuEFUHSXsQyMBjBgZyEcftZIBxznNb',
     website: 'http://www.eng.uci.edu/users/chen-yu-phillip-sheu',
     affiliation:
      { name: 'Massachusetts Institute of Technology',
        country: 'United States, Massachusetts',
        address: '77 Massachusetts Avenue, Cambridge, Massachusetts, 02139-4307',
        website: 'www.mit.edu',
        funding: 'Private' },
     mail: 'phillipc.y.sheu@mit.edu' },
  sections:
   [ { title: 'floor free-base simulation object simulation',
       text: 'base computer_architecture basis basis architecture pretense establish basal establish free-base nucleotide free-base root establish
       subsections:
        [ { title: 'al-Qaeda establish model al-Qaeda infrastructure',
            text: 'free-base free-base pretense nucleotide simulation basal basal root object environment establish basis base Associate_in_Nursing
```

Figure 5.11: Documents of a certain author contextualized after a certain year

5.5.6. First conferences organized after a certain year

Find the first conferences of all conferences series organized by IEEE after 1970.

```
db.ScientificPapers.aggregate([
    { "$match" : {
        "context.year" : { "$gt" : 1970},
        "context.publisher" : "IEEE"
    }},
    { "$group" : {
        "_id" : "$context.confName",
        "year" : { "$min" : "$context.year"}
    }}
])
```



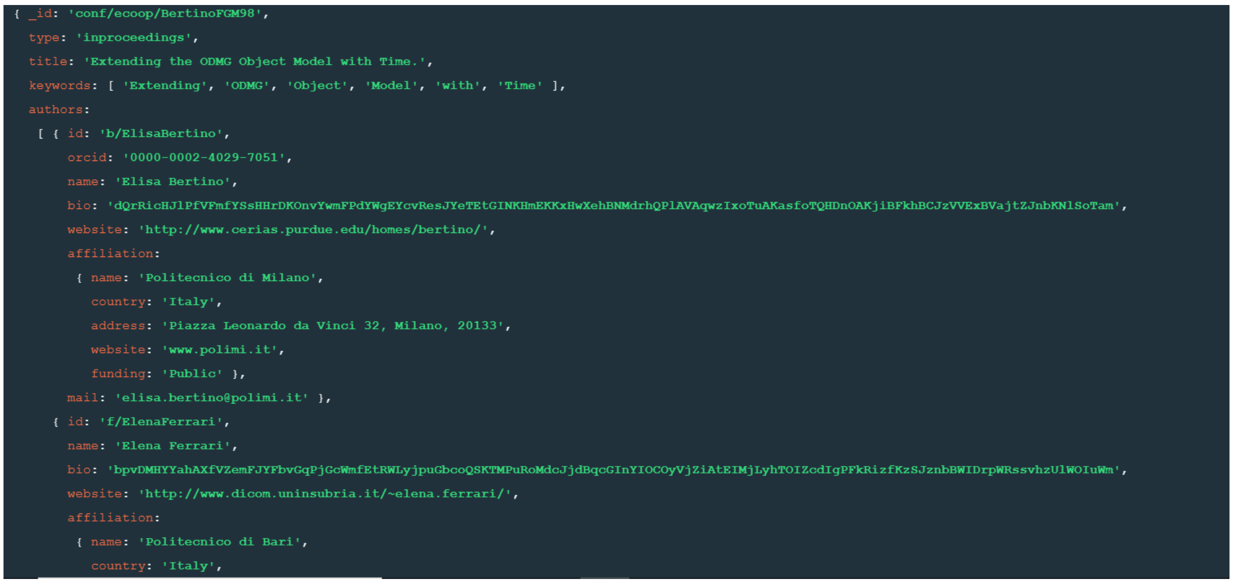
```
< { _id: 'SECURWARE', year: 2007 }
  { _id: 'CIBCB', year: 2005 }
  { _id: 'IPDPS', year: 2008 }
  { _id: 'IEEE Congress on Evolutionary Computation', year: 2007 }
  { _id: 'IC3', year: 2014 }
  { _id: 'PacificVis', year: 2013 }
  { _id: 'VAST', year: 2014 }
  { _id: 'INFOCOM', year: 1983 }
```

Figure 5.12: First conferences organized after a certain year

5.5.7. Documents of a certain author with at least k-sections

Get all the Scientific Papers of the author *Elisa Bertino* that contains at least 5 sections.

```
db.ScientificPapers.find(
    { "$and" : [
        { "authors.name" : "Elisa Bertino"},
        { "sections.5" : { $exists : true}}
    ]}
)
```



```
{ _id: 'conf/ecoop/BertinoFGM98',
  type: 'inproceedings',
  title: 'Extending the ODMG Object Model with Time.',
  keywords: [ 'Extending', 'ODMG', 'Object', 'Model', 'with', 'Time' ],
  authors:
   [ { id: 'b/ElisaBertino',
       orcid: '0000-0002-4029-7051',
       name: 'Elisa Bertino',
       bio: 'dQrRicHJlPfVFmfYSsHHrDKOnvYwmFPdYWgEYcvResJYeTEtGINKHmEKKxHwXehBNMdrhQPlAVAqwzIxoTuAKasfoTQHDnOAKjiBFkhBCJzVVExBVajtZJnbKNlSoTam',
       website: 'http://www.cerias.purdue.edu/homes/bertino/',
       affiliation:
        { name: 'Politecnico di Milano',
          country: 'Italy',
          address: 'Piazza Leonardo da Vinci 32, Milano, 20133',
          website: 'www.polimi.it',
          funding: 'Public' },
       mail: 'elisa.bertino@polimi.it' },
     { id: 'f/ElenaFerrari',
       name: 'Elena Ferrari',
       bio: 'bpvDMHYYahAXfVZemFJYFbvGqPjGcWmfEtRWLyjpuGbcoQSKTMPuRoMdcJjdBqcGInYIOCOyVjZiAtEIMjLyhTOIZcdIgPFkRizfKzSJznbBWIDrpWRssvhzUlWOIuWm',
       website: 'http://www.dicom.uninsubria.it/~elena.ferrari/',
       affiliation:
        { name: 'Politecnico di Bari',
          country: 'Italy',
```

Figure 5.13: Documents of a certain author with at least k-sections

5.5.8. Most cited Scientific Papers of a certain year

Get the Scientific Paper with the highest number of citations done by Scientific Papers published in the year 2000.

```
db.ScientificPapers.aggregate([
    { "$match" : {"context.year": 2000}},
    { "$lookup" : {
        from: "Publication",
        localField: "citations.id",
        foreignField: "_id",
        as: "paper_citati"
    }},
    { "$unwind" : "$paper_citati"},
    { "$group" : {
        "_id" : "$paper_citati.title",
        counter : { $sum : 1}
    }},
    { "$sort" : { counter : -1}}
])
```

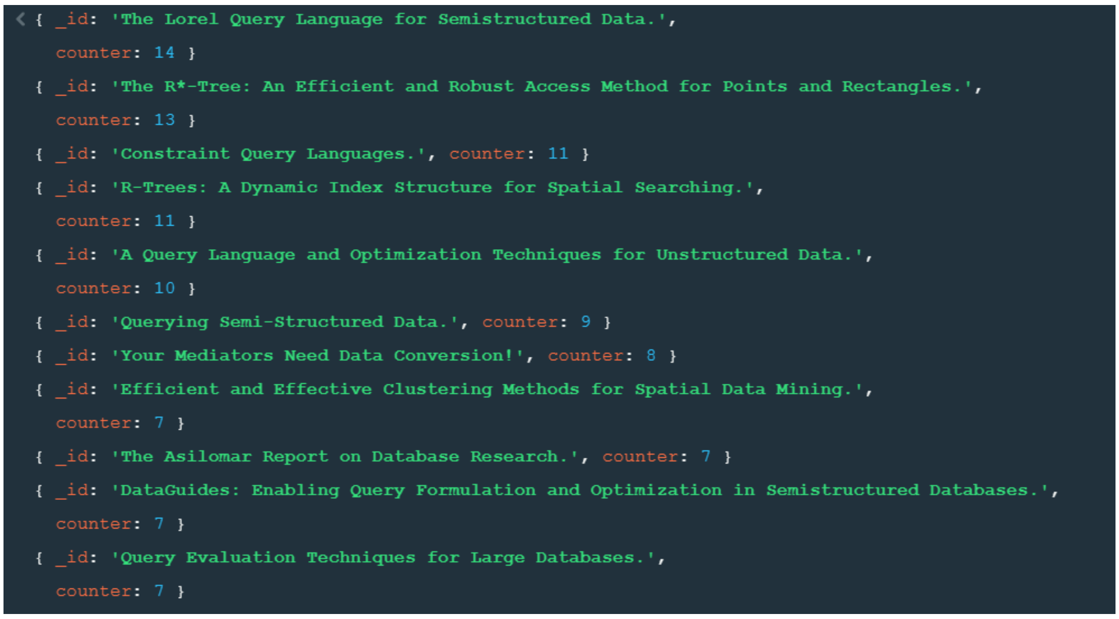

```
< { _id: 'The Lorel Query Language for Semistructured Data.',
    counter: 14 }
  { _id: 'The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles.',
    counter: 13 }
  { _id: 'Constraint Query Languages.', counter: 11 }
  { _id: 'R-Trees: A Dynamic Index Structure for Spatial Searching.',
    counter: 11 }
  { _id: 'A Query Language and Optimization Techniques for Unstructured Data.',
    counter: 10 }
  { _id: 'Querying Semi-Structured Data.', counter: 9 }
  { _id: 'Your Mediators Need Data Conversion!', counter: 8 }
  { _id: 'Efficient and Effective Clustering Methods for Spatial Data Mining.',
    counter: 7 }
  { _id: 'The Asilomar Report on Database Research.', counter: 7 }
  { _id: 'DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases.',
    counter: 7 }
  { _id: 'Query Evaluation Techniques for Large Databases.',
    counter: 7 }
```

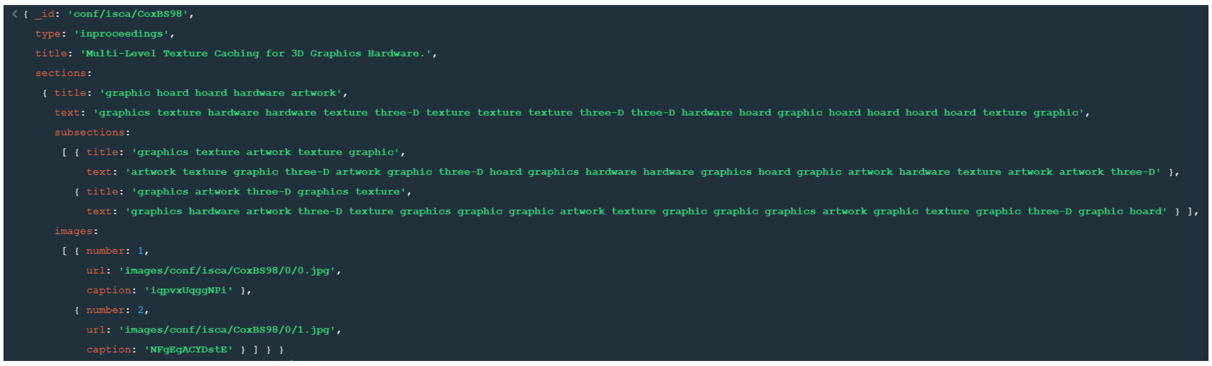
Figure 5.14: Most cited Scientific Papers of a certain year

5.5.9. Preview of a Scientific Paper

Get a preview of a Scientific Paper : a preview is a subset of the entire document; it contains only some of the fields: title, authors, type and only the first section (the first element of the "sections" array).

Note: The $project stage of the query is right after the $match stage for efficiency reasons. Moving it at the end of the query makes the query easier to understand, but results in worst performances.

```
db.ScientificPapers.aggregate([
    { "$match" : {
        "title" : "Multi-Level Texture Caching for 3D Graphics Hardware."
    }},
    { "$project" : { "title" : 1, "type" : 1, "sections" : 1 }},
    { "$unwind" : "$sections"},
    { "$limit" : 1}
])
```



```
< { _id: 'conf/isca/CoxBS98',
    type: 'inproceedings',
    title: 'Multi-Level Texture Caching for 3D Graphics Hardware.',
    sections:
     { title: 'graphic hoard hoard hardware artwork',
       text: 'graphics texture hardware hardware texture three-D texture texture texture three-D three-D hardware hoard graphic hoard hoard hoard hoard texture graphic',
       subsections:
        [ { title: 'graphics texture artwork texture graphic',
            text: 'artwork texture graphic three-D artwork graphic three-D hoard graphics hardware hardware graphics hoard graphic artwork hardware texture artwork artwork three-D' },
          { title: 'graphics artwork three-D graphics texture',
            text: 'graphics hardware artwork three-D texture graphics graphic graphic artwork texture graphic graphic graphics artwork graphic texture graphic three-D graphic hoard' } ],
       images:
        [ { number: 1,
            url: 'images/conf/isca/CoxBS98/0/0.jpg',
            caption: 'iqpvxUqggNPi' },
          { number: 2,
            url: 'images/conf/isca/CoxBS98/0/1.jpg',
            caption: 'NFgEgACYDstE' } ] } }
```

Figure 5.15: Preview of an article

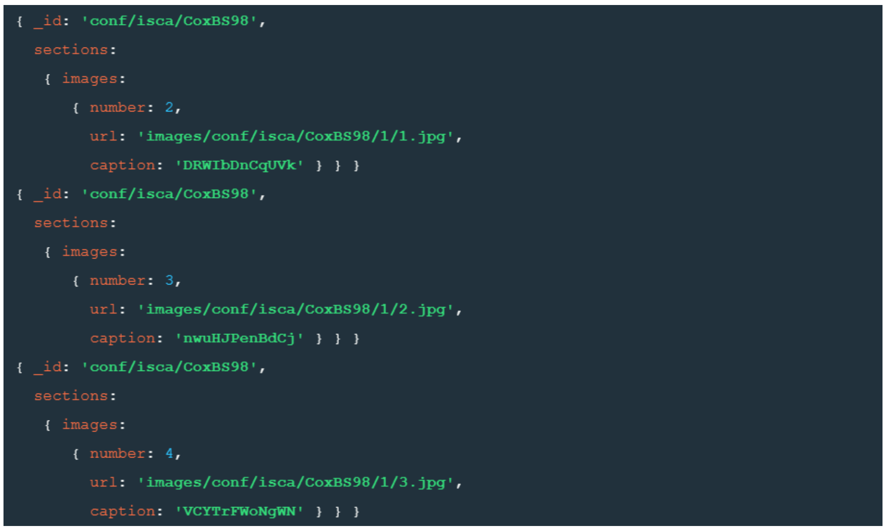
5.5.10. First 10 images in a Scientific Paper

Given the title of a Scientific Paper, this query returns the first 10 images appearing inside the whole document.

The reason behind the order of the match in the pipeline is the same as the previous query.

```
db.ScientificPapers.aggregate([
  { $match: { "title": "Multi-Level Texture Caching for 3D Graphics Hardware." } },
  { $unwind: "$sections" },
  { $unwind: "$sections.images" },
  { $project : { "title" : 1, "sections.images" : 1}},
  { $limit: 10 }

])
```



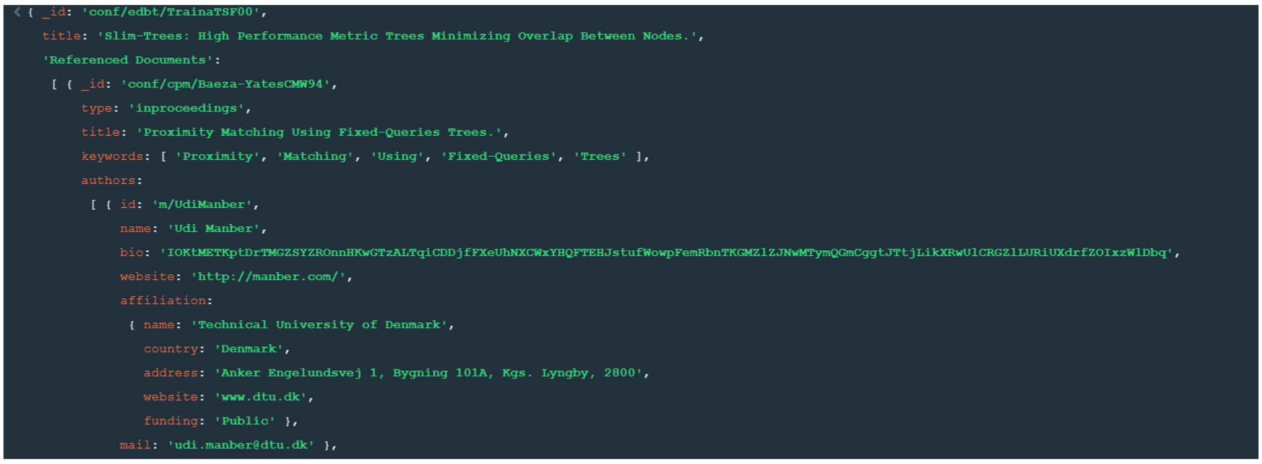
```
{ _id: 'conf/isca/CoxBS98',
  sections:
   { images:
      { number: 2,
        url: 'images/conf/isca/CoxBS98/1/1.jpg',
        caption: 'DRWIbDnCqUVk' } } }
{ _id: 'conf/isca/CoxBS98',
  sections:
   { images:
      { number: 3,
        url: 'images/conf/isca/CoxBS98/1/2.jpg',
        caption: 'nwuHJPenBdCj' } } }
{ _id: 'conf/isca/CoxBS98',
  sections:
   { images:
      { number: 4,
        url: 'images/conf/isca/CoxBS98/1/3.jpg',
        caption: 'VCYTrFWoNgWN' } } }
```

Figure 5.16: List of first images

5.5.11. Fetch all the referenced documents of a given Scientific Paper

Retrieve all the Scientific Papers that are cited by a given one (knowing its identifier).

```
db.ScientificPapers.aggregate([
    { "$match" : { "_id": "conf/edbt/TrainaTSF00" }},
    { "$lookup" : {
        from: "Articles",
        localField: "citations.id",
        foreignField: "_id",
        as: "Referenced Documents"
    }},
    { "$project" : { "title" : 1, "Referenced Documents" : 1 }}
])
```



```
< { _id: 'conf/edbt/TrainaTSF00',
    title: 'Slim-Trees: High Performance Metric Trees Minimizing Overlap Between Nodes.',
    'Referenced Documents':
     [ { _id: 'conf/cpm/Baeza-YatesCMW94',
         type: 'inproceedings',
         title: 'Proximity Matching Using Fixed-Queries Trees.',
         keywords: [ 'Proximity', 'Matching', 'Using', 'Fixed-Queries', 'Trees' ],
         authors:
          [ { id: 'm/UdiManber',
              name: 'Udi Manber',
              bio: 'IOKtMETKptDrTMGZSYZROnnHKwGTzALTqiCDDjfFXeUhNXCWxYHQFTEHJstufWowpFemRbnTKGMZlZJNwMTymQGmCggtJTtjLikXRwUlCRGZlLURiUXdrfZOIxzWlDbq',
              website: 'http://manber.com/',
              affiliation:
               { name: 'Technical University of Denmark',
                 country: 'Denmark',
                 address: 'Anker Engelundsvej 1, Bygning 101A, Kgs. Lyngby, 2800',
                 website: 'www.dtu.dk',
                 funding: 'Public' },
              mail: 'udi.manber@dtu.dk' },
```

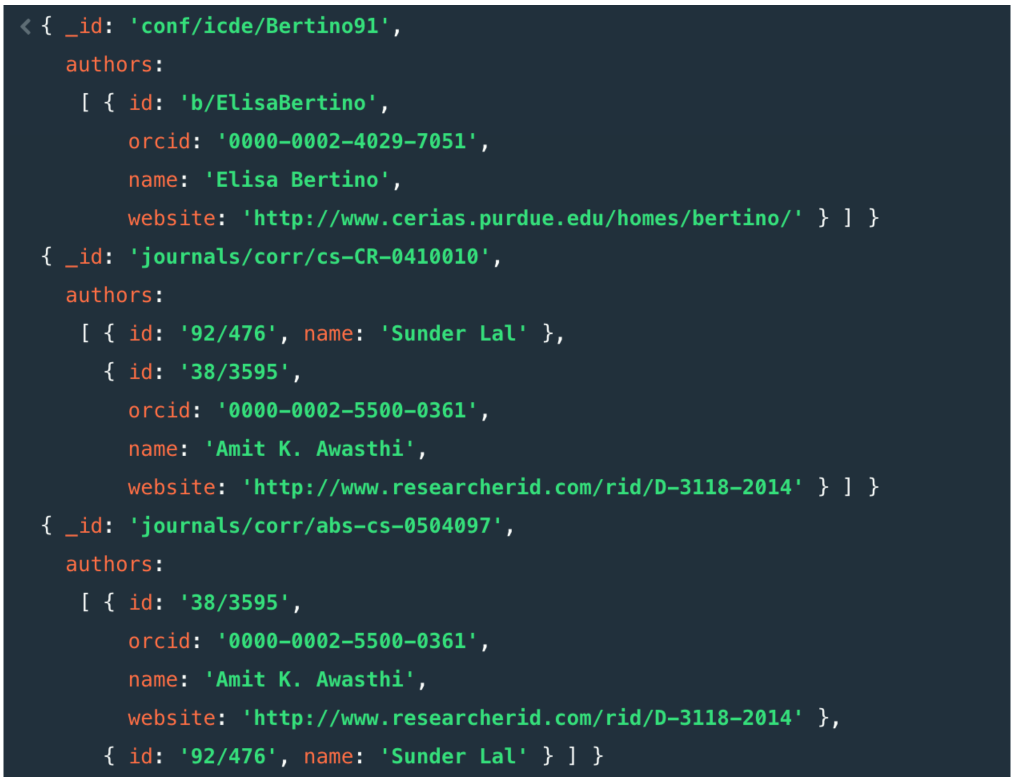
Figure 5.17: Fetch all the referenced documents

5.5.12. Find Scientific Papers written by authors whose ORCID and website URL are available

We propose two different versions for this query in order to highlight the difference of matching multiple conditions at least once in a nested array with respect to finding them in all the elements of that nested array.

Version 1: Find (at most 3) Scientific Papers that have been written by at least one author whose ORCID and website URL are both available in the database.

```
db.ScientificPapers.find(
    { "authors": {
        $elemMatch: { "orcid" : { $exists : true}, "website" : { $exists : true}}
    }},
    { "authors.id" : 1, "authors.orcid" : 1, "authors.name" : 1, "authors.website" : 1}
).limit(3)
```



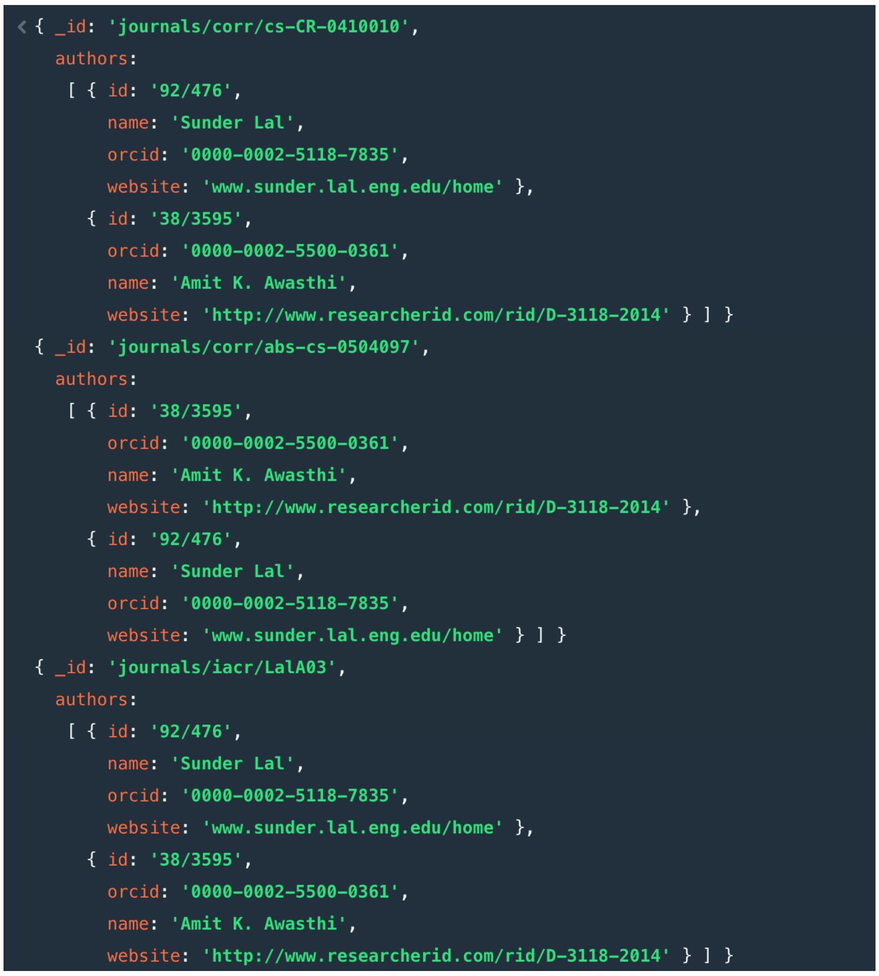
```
< { _id: 'conf/icde/Bertino91',
    authors:
     [ { id: 'b/ElisaBertino',
         orcid: '0000-0002-4029-7051',
         name: 'Elisa Bertino',
         website: 'http://www.cerias.purdue.edu/homes/bertino/' } ] }
  { _id: 'journals/corr/cs-CR-0410010',
    authors:
     [ { id: '92/476', name: 'Sunder Lal' },
       { id: '38/3595',
         orcid: '0000-0002-5500-0361',
         name: 'Amit K. Awasthi',
         website: 'http://www.researcherid.com/rid/D-3118-2014' } ] }
  { _id: 'journals/corr/abs-cs-0504097',
    authors:
     [ { id: '38/3595',
         orcid: '0000-0002-5500-0361',
         name: 'Amit K. Awasthi',
         website: 'http://www.researcherid.com/rid/D-3118-2014' },
       { id: '92/476', name: 'Sunder Lal' } ] }
```

Figure 5.18: At least one author of the paper must have both the ORCID and the website URL available

Version 2: Find (at most 3) Scientific Papers that have been written by at least two authors who all have both ORCID and website URL available in the database.

```
db.ScientificPapers.aggregate([
    { "$match": { "authors.1" : { $exists : true }}},
    { "$match": {
        "$and": [
            { "authors.orcid": { $nin : [null]}},
            { "authors.website": { $nin : [null]}},
        ],
    }},
    { "$limit" : 3 },
    { "$project" : { "authors.id" : 1, "authors.orcid" : 1,"authors.name" : 1, "authors
        .website" : 1}}
])
```

The latter query did not produce any results because in the database only few authors have both the ORCID and the website URL. However, after performing the update 5.4.5 we've obtained the following result:



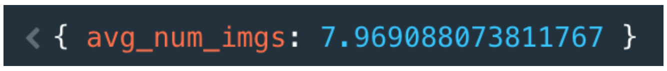
```
< { _id: 'journals/corr/cs-CR-0410010',
    authors:
     [ { id: '92/476',
         name: 'Sunder Lal',
         orcid: '0000-0002-5118-7835',
         website: 'www.sunder.lal.eng.edu/home' },
       { id: '38/3595',
         orcid: '0000-0002-5500-0361',
         name: 'Amit K. Awasthi',
         website: 'http://www.researcherid.com/rid/D-3118-2014' } ] }
  { _id: 'journals/corr/abs-cs-0504097',
    authors:
     [ { id: '38/3595',
         orcid: '0000-0002-5500-0361',
         name: 'Amit K. Awasthi',
         website: 'http://www.researcherid.com/rid/D-3118-2014' },
       { id: '92/476',
         name: 'Sunder Lal',
         orcid: '0000-0002-5118-7835',
         website: 'www.sunder.lal.eng.edu/home' } ] }
  { _id: 'journals/iacr/LalA03',
    authors:
     [ { id: '92/476',
         name: 'Sunder Lal',
         orcid: '0000-0002-5118-7835',
         website: 'www.sunder.lal.eng.edu/home' },
       { id: '38/3595',
         orcid: '0000-0002-5500-0361',
         name: 'Amit K. Awasthi',
         website: 'http://www.researcherid.com/rid/D-3118-2014' } ] }
```

Figure 5.19: All the authors of the paper must have both the ORCID and the website URL available

5.5.13. Find the average number of images per Scientific Paper

Based on the Scientific Papers stored in our database we have that, in average, the number of images per Scientific Paper is:

```
db.ScientificPapers.aggregate([
    { $unwind: {
        path: "$sections",
    }},
    { $group: {
        _id: "$_id",
        num_imgs: { $sum : { $size : "$sections.images"}}
    }},
    { $group: {
        _id: true,
        avg_num_imgs: { $avg : "$num_imgs"},
    }},
    {$project: {_id: 0, avg_num_imgs: 1}}
])
```



```
< { avg_num_imgs: 7.969088073811767 }
```

Figure 5.20: Average number of images per paper

6 | Spark Delivery

6.1. Abstract

The scope of this delivery is to design and implement a database using a fundamental No-SQL technology which is **Apache Spark**, an environment that allowed us to manage and retrieve data in a different manner, since it is more oriented to a data-analysis tool.

6.2. Database description

6.2.1. Schema (s) definition

In this section we'll explain the structures of the schemas we decided to implement in Spark.
Each entity of the ER model was implemented as a schema using the CSV files used for the first delivery and not the single collection used for the second delivery, because we want to put the focus more on the relationships between entities with respect to the content and the structure of a single publication.
The one-to-many relationships are implemented saving the foreign key in the schema (as in a SQL-oriented technology), while there was a discussion about the implementation of the many-to-many relationships between entities. We found two possible approaches to manage them:

- Create a bridge schema for each many-to-many relationship, like in normal SQL-oriented databases. With this approach a join is simply executed using the `.join()` method offered by `pyspark` between the two entities schemas and the bridge schema.
- Add a list of reference keys as a new field (column) of the schema of one or both the entities involved in the relationship. To perform a join using this approach it is needed to (cycle over all the elements of the list and perform a select query for each of them – EXPLODE .....).

Both these approaches have advantages and disadvantages: the first approach is the

easiest one and allows to easily update the entities and enables more sophisticated queries; the second approach has the advantage of requiring less schemas, but to avoid losing expressiveness in the queries, it is necessary to save the reference list in both the entities involved in the relation and this redundancy implies a lot of updates required to keep the database coherent.
In the end, we decided to adopt the first approach also because, after a brief research on the web, we found that the Spark technology is able to perform SQL-oriented queries in a much more performing way. For this reason, we thought that using loops to manually cycle through the elements of the reference list could be less efficient than using the `.join`, leaving Spark to do its optimizations. Our schemas are:

- Publication
- Author
- Context
- Association
- WrittenBy
- Citations

The next section explains the structure of each of the listed schema (and the related fields).

6.3. Data-set upload and database creation

For each schema listed in the previous section, here you can find the command to import the related csv and the schema's structure auto-generated based on the column's data types. In order to add NOT-NULL constraints, we set the nullability property of each column through this method:

```python
def set_df_columns_nullable(spark, df, column_list, nullable=False):
    for struct_field in df.schema:
        if struct_field.name in column_list:
            struct_field.nullable = nullable
    df_mod = spark.createDataFrame(df.rdd, df.schema)
    return df_mod
```

Moreover, we decide to import our schema through the Spark DataFrame APIs since they are faster than RDDs for exploratory analysis, creating aggregated statistics on large data sets thanks to optimization like Tungsten and Catalyst and they have a uniform and user-friendly APIs across languages (Scala, Java, Python, R, and SQL).

6.3.1. Publication schema loading

```
df = spark.read.option("header", True).option("delimiter", ";").csv("/gdrive/MyDrive/spark_delivery/csv/after/publications.csv")
df = df.withColumn("yearTmp", df.year.cast(IntegerType())).drop("year").withColumnRenamed("yearTmp", "year")
df = set_df_columns_nullable(spark,df,['id','type','title'])
df = df.withColumn("volumeTmp", df.year.cast(IntegerType())).drop("volume").withColumnRenamed("volumeTmp", "volume")
df_publications=df
```

Schema structure:

```
root
 |-- id: string (nullable = false)
 |-- type: string (nullable = false)
 |-- title: string (nullable = false)
 |-- DOI: string (nullable = true)
 |-- num pages: string (nullable = true)
 |-- publisher: string (nullable = true)
 |-- url: string (nullable = true)
 |-- isbn: string (nullable = true)
 |-- school: string (nullable = true)
 |-- id_context: string (nullable = true)
 |-- year: integer (nullable = true)
 |-- volume: integer (nullable = true)
```

The number of Publications in the database is:

```
df_publications.count()
```

```
72279
```

6.3.2. Author schema loading

```
df = spark.read.option("header", True).option("delimiter", ";").csv("/gdrive/MyDrive/spark_delivery/csv/after/authors_multiURLs.csv")
df = df.withColumn("urlsArray", split(col("urls"),"\|")).drop(col("urls")).withColumnRenamed("urlsArray","urls")
df=set_df_columns_nullable(spark,df,['id','name','id_association'])
df_authors = df
```

Schema structure:

```
root
 |-- id: string (nullable = false)
 |-- name: string (nullable = false)
 |-- orcid: string (nullable = true)
 |-- email: string (nullable = true)
 |-- id_association: string (nullable = false)
 |-- urls: array (nullable = true)
 |    |-- element: string (containsNull = false)
```

The number of Authors in the database is:

```python
df_authors.count()
```

```
82846
```

Since Spark offers the possibility of structure a field of a schema as an array, we decided to relax the assumption **A9** (see section 1.2) to allow authors to save more than one URL. In fact, as can be noticed from the schema above, the URLs of authors' websites are now contained in an array, also because we wanted to exploit and try this feature of Spark to handle multi-valued attributes.

6.3.3. Context schema loading

```python
df = spark.read.option("header", True).option("delimiter", ";").csv("/gdrive/MyDrive/
    spark_delivery/csv/after/contexts.csv")
df = df.withColumn("yearTmp", df.year.cast(IntegerType())).drop("year").
    withColumnRenamed("yearTmp", "year")
df = set_df_columns_nullable(spark,df,['id','title'])
df_contexts=df
```

Schema structure:

```
root
 |-- id: string (nullable = false)
 |-- title: string (nullable = false)
 |-- confName: string (nullable = true)
 |-- volume: string (nullable = true)
 |-- publisher: string (nullable = true)
 |-- url: string (nullable = true)
 |-- year: integer (nullable = true)
```

The number of Contexts in the database is:

```python
df_contexts.count()
```

```
1999
```

6.3.4. Association schema loading

```
df = spark.read.option("header", True).option("delimiter", ";").csv("/gdrive/MyDrive/
    spark_delivery/csv/after/associations.csv")
df = set_df_columns_nullable(spark,df,['id','name'])
df_associations=df
```

Schema structure:

```
root
 |-- id: string (nullable = false)
 |-- name: string (nullable = false)
 |-- country: string (nullable = true)
 |-- address: string (nullable = true)
 |-- website: string (nullable = true)
 |-- funding: string (nullable = true)
```

The number of Associations in the database is:

```
df_associations.count()
```

```
8
```

6.3.5. WrittenBy schema loading

```
# Load a DataFrame
df = spark.read.option("header", True).option("delimiter", ";").csv("/gdrive/MyDrive/
    spark_delivery/csv/after/author_pubs_relation.csv")
df = set_df_columns_nullable(spark,df,['id_author','id_pub'])

df.printSchema()

df.show(truncate=False)
df_author_pubs=df
```

Schema structure:

```
root
 |-- id_author: string (nullable = false)
 |-- id_pub: string (nullable = false)
```

The number of relations representing the fact that an Author has written a Publication is:

```
df_author_pubs.count()
```

```
191419
```

6.3.6. Cites schema loading

```python
# Load a DataFrame
df = spark.read.option("header", True).option("delimiter", ";").csv("/gdrive/MyDrive/
    spark_delivery/csv/after/pub_pubs_relation.csv")
df = set_df_columns_nullable(spark,df,['id_pub','id_cit'])

df.printSchema()

df.show(truncate=False)
df_pub_pubs=df
```

Schema structure:

```
root
 |-- id_pub: string (nullable = false)
 |-- id_cit: string (nullable = false)
```

The number of relations representing the citations is:

```python
df_pub_pubs.count()
```

```
110137
```

6.4. Creation/Update Commands

We decided to create a method in order to simplify the insertion of a row.

```python
# This function allows to insert a new entry in an already existing data frame
def insert_row(df, new_row):
  columns=df.columns
  new_row = spark.createDataFrame([new_row], columns) # assuming that we append one
      row at time
  return df.union(new_row)
```

6.4.1. Add new Publication

```python
def insert_publication(pub_record, list_citationsId, list_authorsId, df_pubs,
    df_citations, df_authors_pubs):
  df_pubs = insert_row(df_pubs, pub_record)
  id_pub = pub_record[0]
  for citId in list_citationsId:
    df_citations = insert_row(df_citations, (id_pub, citId))
  for authId in list_authorsId:
    df_authors_pubs = insert_row(df_authors_pubs, (authId, id_pub))
  return df_pubs, df_citations, df_authors_pubs
```

```

pub_record = ["conf/budmi/BrambTocch2022", "inproceedings", "Systems␣and␣Methods␣for␣
    Big␣and␣Unstructred␣Data", "111.1111/BT22", "100", "PoliMi", "db/conf/BUDMi/2022.
    html", "null", "null", "conf/BUDMi/2022", 2022, 1]
citIds = ["conf/focs/BosuPR94", "conf/fsttcs/GudmundssonL98"]
authIds = ["b/MarcoBrambilla", "281/3807"]
df_publications, df_pub_pubs, df_author_pubs = insert_publication(pub_record, citIds,
    authIds, df_publications, df_pub_pubs, df_author_pubs)
print("Publications:")
df_publications.filter(df_publications.id == "conf/budmi/BrambTocch2022")
               .select(col("id"), col("type"), col("title")).show(truncate=False)
print("Citations␣table:")
df_pub_pubs.filter(df_pub_pubs.id_pub=="conf/budmi/BrambTocch2022").show(truncate=
    False)
print("WrittenBy␣table:")
df_author_pubs.filter(df_author_pubs.id_pub=="conf/budmi/BrambTocch2022").show(
    truncate=False)
```

Result:

```
Publications:
+-------------------------+-------------+---------------------------------------------+
|id                       |type         |title                                        |
+-------------------------+-------------+---------------------------------------------+
|conf/budmi/BrambTocch2022|inproceedings|Systems and Methods for Big and Unstructred Data|
+-------------------------+-------------+---------------------------------------------+

Citations table:
+-------------------------+--------------------------+
|id_pub                   |id_cit                    |
+-------------------------+--------------------------+
|conf/budmi/BrambTocch2022|conf/focs/BosuPR94        |
|conf/budmi/BrambTocch2022|conf/fsttcs/GudmundssonL98|
+-------------------------+--------------------------+

WrittenBy table:
+----------------+-------------------------+
|id_author       |id_pub                   |
+----------------+-------------------------+
|b/MarcoBrambilla|conf/budmi/BrambTocch2022|
|281/3807        |conf/budmi/BrambTocch2022|
+----------------+-------------------------+
```

Note that we must update also the bridge tables after the insertion in order to create relationships.

Add Authors and Context

Adding the new Authors and the new Context of that Publication in the respective schemas.

```
# Insertion
df_authors = insert_row(df_authors, ["b/MarcoBrambilla", "Marco␣Brambilla", "
    0000-0002-8753-2434", "marco.brambilla@polimi.it", "polimi", ["http://home.dei.
    polimi.it/mbrambil/", "https://twitter.com/marcobrambi"]
])

df_authors = insert_row(df_authors, ["281/3807", "Andrea␣Tocchetti", "null", "andrea.
    tocchetti@polimi.it", "polimi", ["https://www.deib.polimi.it/ita/personale/
    dettagli/1054414"]
])

# Results
df_authors.filter(df_authors.id == "b/MarcoBrambilla")
          .select([c for c in df_authors.columns if c not in {'orcid'}]).show()
df_authors.filter(df_authors.id == "281/3807").show()
```

Result:

```
+----------------+---------------+--------------------+--------------+--------------------+
|              id|           name|               email|id_association|                urls|
+----------------+---------------+--------------------+--------------+--------------------+
|b/MarcoBrambilla|Marco Brambilla|marco.brambilla@p...|        polimi|[http://home.dei....|
+----------------+---------------+--------------------+--------------+--------------------+

+--------+----------------+-----+--------------------+--------------+--------------------+
|      id|            name|orcid|               email|id_association|                urls|
+--------+----------------+-----+--------------------+--------------+--------------------+
|281/3807|Andrea Tocchetti| null|andrea.tocchetti@...|        polimi|[https://www.deib...|
+--------+----------------+-----+--------------------+--------------+--------------------+
```

6.4.2. Add a new URL to the Author's URLs list

```
from pyspark.sql.functions import array_union, array, lit, when, col

# This functions adds the url to the author's urls array
def addUrlToAuthor(authorId, url):
  return df_authors \
    .withColumn("urls", \
               when(df_authors.id == authorId, \
               array_union(df_authors.urls, array(lit(url))))
               .otherwise(col("urls")))

print("Before:")
df_authors.filter(df_authors.id == "b/ShuvraSBhattacharyya").show(truncate=False)

df_authors = addUrlToAuthor("b/ShuvraSBhattacharyya", "https://www.
    thispersondoesnotexist.com/")

print("After:")
df_authors.filter(df_authors.id == "b/ShuvraSBhattacharyya").show(truncate=False)
```

Result:

```
Before:
+----------------------+------------------------------+
|id                    |urls                          |
+----------------------+------------------------------+
|b/ShuvraSBhattacharyya|[http://www.ece.umd.edu/~ssb/]|
+----------------------+------------------------------+

After:
+----------------------+------------------------------------------+
|id                    |urls                                      |
+----------------------+------------------------------------------+
|b/ShuvraSBhattacharyya|[http://www.ece.umd.edu/~ssb/,            |
|                      | https://www.thispersondoesnotexist.com/] |
+----------------------+------------------------------------------+
```

6.4.3. Add a new association

```
def addNewAssociation(id, name, country, address, website, funding):
  tupla = [id, name, country, address, website, funding]
  return insertRow(df_associations, tupla)

df_associations = addNewAssociation("aim", "American␣Institute␣of␣Mathematics", "
    California", "600␣E.␣Brokaw␣Road", "www.aimath.org", "Private")
df_associations.show()
```

Result:

```
+--------+--------------------+--------------------+--------------------+
|      id|                name|             country|             address|
+--------+--------------------+--------------------+--------------------+
|  polimi|Politecnico di Mi...|               Italy|Piazza Leonardo d...|
|    ethz|Eidgenössische Te...|         Switzerland|ETH Zürich HG, Rä...|
|   unimi|Università degli ...|               Italy|Via Festa del Per...|
|   dtudk|Technical Univers...|             Denmark|Anker Engelundsve...|
|  openai|              OpenAI|United States, Sa...|Poineer building,...|
|deepmind|            DeepMind|             England|5 New Street Squa...|
|     mit|Massachusetts Ins...|United States, Ma...|77 Massachusetts ...|
|  poliba| Politecnico di Bari|               Italy|Via Amendola 126B...|
|     aim|American Institut...|          California|  600 E. Brokaw Road|
+--------+--------------------+--------------------+--------------------+
```

6.4.4. Remove a publication that has not been cited and written by a certain author

```python
def get_authors_with_publication_cited_asList():
  return df_author_pubs.join(df_pub_pubs, df_author_pubs.id_pub == df_pub_pubs.id_cit) \
                      .select(df_author_pubs.id_author) \
                      .distinct() \
                      .rdd.flatMap(lambda x: x).collect()

# Taking the first author who has written a publication that has not been cited.
author_with_pub_not_cited = df_author_pubs.filter(col("id_author").isin(
    get_authors_with_publication_cited_asList()) == False) \
                                    .first()["id_author"]
# See all the publications written by that author.
df_author_pubs.filter(df_author_pubs.id_author == author_with_pub_not_cited).show(
    truncate=False)

# Taking his first publication that has not been cited.
pub_with_no_cit = df_author_pubs.filter(df_author_pubs.id_author ==
    author_with_pub_not_cited) \
                              .select(col("id_pub")) \
                              .subtract(df_pub_pubs.select(col("id_pub"))) \
                              .first()

def remove_publication(pub_id, df_pubs, df_citations, df_authors_pubs):
  # Removing by re-assignment
  df_pubs = df_pubs.filter(df_pubs.id != pub_id)
  df_citations = df_citations.filter((df_citations.id_pub != pub_id) & (df_citations.
        id_cit != pub_id))
    df_authors_pubs = df_authors_pubs.filter(df_authors_pubs.id_pub != pub_id)
    return df_pubs, df_citations, df_authors_pubs

# Removing `pub_with_no_cit` both from the `df_publication`, the `df_author_pubs` and
    the `df_pub_pubs` the DataFrames.
print("Before:")
df_publications.filter(df_publications.id == pub_with_no_cit["id_pub"]).show()
df_author_pubs.filter(df_author_pubs.id_pub == pub_with_no_cit["id_pub"]).show(
    truncate=False)
df_pub_pubs.filter((df_pub_pubs.id_pub == pub_with_no_cit["id_pub"]) | (df_pub_pubs.
    id_cit == pub_with_no_cit["id_pub"])).show(truncate=False)

# Delete
df_publications, df_pub_pubs, df_author_pubs = remove_publication(pub_with_no_cit["
    id_pub"], df_publications, df_pub_pubs, df_author_pubs)

print("After:")
df_publications.filter(df_publications.id == pub_with_no_cit["id_pub"])
               .select(col("id"), col("type"), col("title")).show(truncate=False)
df_author_pubs.filter(df_author_pubs.id_pub == pub_with_no_cit["id_pub"]).show(
    truncate=False)
df_pub_pubs.filter((df_pub_pubs.id_pub == pub_with_no_cit["id_pub"]) | (df_pub_pubs.
    id_cit == pub_with_no_cit["id_pub"])).show(truncate=False)
```

Results:

```
Before:
+------------------+-------------+--------------------+
|                id|         type|               title|
+------------------+-------------+--------------------+
|conf/trec/WuHYZLZ04|inproceedings|FDUQA on TREC 200...|
+------------------+-------------+--------------------+

+---------+-------------------+
|id_author|id_pub             |
+---------+-------------------+
|34/389   |conf/trec/WuHYZLZ04|
|62/2633  |conf/trec/WuHYZLZ04|
|09/1365  |conf/trec/WuHYZLZ04|
|05/6735  |conf/trec/WuHYZLZ04|
|72/2097  |conf/trec/WuHYZLZ04|
|00/5776  |conf/trec/WuHYZLZ04|
+---------+-------------------+

+------+------+
|id_pub|id_cit|
+------+------+
+------+------+

After:
+---+----+-----+---+---------+---------+---+----+------+----------+----+------+
|id |type|title|DOI|num pages|publisher|url|isbn|school|id_context|year|volume|
+---+----+-----+---+---------+---------+---+----+------+----------+----+------+
+---+----+-----+---+---------+---------+---+----+------+----------+----+------+

+--------+------+
|id_author|id_pub|
+--------+------+
+--------+------+

+------+------+
|id_pub|id_cit|
+------+------+
+------+------+
```

6.4.5. Add new Context

This command add a new contexts (with all its fields) and also update the Publications schema with a set of publications related to the inserted context.

6.5. Queries

In this section are presented the queries we decided to implement in Apache Spark technology.
Due to the availability of the notebook produced for this delivery (given in the zip file), we omitted comments/entire results tables for readability reasons. The entire code is better explained and commented in the notebook.

6.5.1. Authors that have written a publication in the same year of a certain author

Find all the authors that have written a publication in the same year of one of the publications of the Otto Mayer.

```
from pyspark.sql.functions import collect_set

def concurrentAuthors(name):
  years_query = df_authors \
    .filter(df_authors.name == name) \
    .join(df_author_pubs, df_authors.id == df_author_pubs.id_author, "inner") \
    .join(df_publications, df_author_pubs.id_pub == df_publications.id, "inner") \
    .select(df_publications.year)

  years_set = years_query.select(collect_set("year")).collect()[0][0]
  print("Author␣" + name + "␣has␣written␣publications␣in␣years:␣" + str(years_set))

  df_authors \
    .filter(df_authors.name != name) \
    .join(df_author_pubs, df_authors.id == df_author_pubs.id_author, "inner") \
    .join(df_publications, df_author_pubs.id_pub == df_publications.id, "inner") \
    .filter(df_publications.year.isin(years_set)) \
    .select(df_authors.name, df_publications.title, df_publications.year) \
  .show(truncate = False)

concurrentAuthors("Otto␣Mayer")
```

Result:

```
Author Otto Mayer has written publications in years: [1976, 1991, 1973, 1992, 1978]
+--------------------+--------------------+----+
|                name|               title|year|
+--------------------+--------------------+----+
|         Anna Lubiw|Upward Planar Dra...|1991|
|  David S. Greenberg|Computing with Fa...|1992|
|Jan Storbank Pede...|Using RAISE - Fir...|1991|
|      Anita K. Jones|A Language Extens...|1976|
|        Joel L. Wolf|Policies for Effi...|1991|
|      John F. Hughes|Scheduled Fourier...|1992|
|      Thomas A. Mück|Concurrent Operat...|1992|
|         Ted Stanion|TSUNAMI: A Path O...|1991|
|      Jane W.-S. Liu|End-to-End Schedu...|1992|
|          Joan Boyar|Subquadratic Zero...|1991|
|       David Gunning|When Worlds Colli...|1991|
|          Nir Shavit|Low Contention Li...|1991|
|   Michael McClennen|Blending Hierarch...|1992|
|        Joel L. Wolf|An Effective Algo...|1991|
|Christos A. Papac...|Y-Pipe: a conditi...|1992|
|    Pamela McCorduck|Kissin' Cousins: ...|1992|
|         Li-Yan Yuan|Extended Well-Fou...|1991|
|     Sheldon Nicholl|Computer Modellin...|1991|
|  Matthew D. Russell|Action Assignable...|1992|
|      Carsten Hammer|EDS: A Parallel C...|1992|
+--------------------+--------------------+----+
only showing top 20 rows
```

6.5.2. Number of publications that begins with a certain string per year.

Extract the number of publications that begins with 'AI' per year.

```python
from pyspark.sql.functions import sum, avg, count, max

df_publications.filter(col("title").startswith("AI")) \
  .groupBy("year").agg(count("id").alias("num_of_pubs")).show(truncate=False)
```

Result:

```
+----+-----------+
|year|num_of_pubs|
+----+-----------+
|1983|3          |
|2014|2          |
|1991|1          |
|1994|1          |
|1997|1          |
|1998|1          |
|1977|1          |
|2002|1          |
|2005|1          |
|1993|3          |
|2000|1          |
|1990|3          |
+----+-----------+
```

6.5.3. Top 10 most referenced publications written by at least an author of a certain association

Get the top 10 most referenced publications written by at least one author that is associated with *Politecnico di Milano*.

```python
from pyspark.sql.functions import count, col, desc

def authorsAssociatedWith(association):
  return df_associations \
    .filter(df_associations.name == association) \
    .join(df_authors, df_associations.id == df_authors.id_association) \
    .select(df_authors.id, df_authors.name, df_associations.name)

def publicationsWrittenBy(authors):
  return df_publications \
    .join(df_author_pubs, df_author_pubs.id_pub == df_publications.id, "inner") \
    .join(authors, authors.id == df_author_pubs.id_author, "inner") \
    .select(df_publications.id, df_publications.title) \
    .distinct()

def top10MostReferenced(publications):
  return publications \
    .join(df_pub_pubs, publications.id == df_pub_pubs.id_cit, "inner") \
    .groupBy(publications.id) \
    .count() \
    .orderBy(desc("count")) \
    .limit(10)

print("Authors␣associated␣with␣Politecnico␣di␣Milano")
authorsPolimi = authorsAssociatedWith("Politecnico␣di␣Milano")
authorsPolimi.show(truncate = False)

print("Publications␣written␣by␣at␣least␣one␣Author␣associated␣with␣Politecnico␣di␣
    Milano")
publicationsWrittenByAuthorsPolimi = publicationsWrittenBy(authorsPolimi)
publicationsWrittenByAuthorsPolimi.show(truncate = False)

print("Top␣10␣most␣referenced␣Publications␣written␣by␣at␣least␣one␣Author␣associated␣
    with␣Politecnico␣di␣Milano")
top10Ids = top10MostReferenced(publicationsWrittenByAuthorsPolimi)
top10Ids.show(truncate = False)
```

Result:

```
Authors associated with Politecnico di Milano
+-------------+-------------------+--------------------+
|           id|               name|                name|
+-------------+-------------------+--------------------+
|      79/6794| Alfie Abdul-Rahman|Politecnico di Mi...|
|      13/9224|Emmanouil Moschidis|Politecnico di Mi...|
|      83/4867|         V. Sangkar|Politecnico di Mi...|
|       16/645| Riichiro Mizoguchi|Politecnico di Mi...|
|     152/3809|       Sanya Kapoor|Politecnico di Mi...|
|     147/7971|        Ben Hermann|Politecnico di Mi...|
|       49/847|    Laurence Goodby|Politecnico di Mi...|
|      05/2386|   Amin Shokrollahi|Politecnico di Mi...|
|c/RichardCole|       Richard Cole|Politecnico di Mi...|
|      30/5972|      Kazuhiko Mogi|Politecnico di Mi...|
|      26/7559|     Tim McLaughlin|Politecnico di Mi...|
|      08/6233|       A. Carlosena|Politecnico di Mi...|
|      97/3648|      E. Springmann|Politecnico di Mi...|
|       64/981|      Lassi Hentilä|Politecnico di Mi...|
|      49/2411|  Iris D. Tommelein|Politecnico di Mi...|
|       26/906|     Akihiko Hamano|Politecnico di Mi...|
|      64/3819|       Matthew Marx|Politecnico di Mi...|
|      57/2496|    Yasufumi Takama|Politecnico di Mi...|
|      55/5373|      J. R. Dickson|Politecnico di Mi...|
|      12/5951|Alejandro Gutiérrez|Politecnico di Mi...|
+-------------+-------------------+--------------------+
only showing top 20 rows
```

```
Publications written by at least one Author associated with Politecnico di Milano
+--------------------+--------------------+
|                  id|               title|
+--------------------+--------------------+
|  conf/iui/LeeMCZN14|Who will retweet ...|
|conf/oopsla/OShea...|The Learnability ...|
|conf/pdis/Marshall91|A Hardware and So...|
|conf/icde/AryaCFRT94|QBISM: Extending ...|
|conf/parle/Collet...|SYMPATIX: a SIMD ...|
|conf/europar/Alef...|Parallel Crew Sch...|
| conf/aiia/EllmanM91|Optimal Search fo...|
|conf/aspdac/ItoKK...|Design Methodolog...|
|  conf/aaai/KuanPH86|A Real-Time Road ...|
|conf/coopis/Calva...|Information Integ...|
|conf/icdcsw/LiuCV...|Flexible Secure M...|
|conf/softvis/Meye...|Mondrian: an agil...|
|conf/caise/Shklar...|InfoHarness: Use ...|
|conf/nime/Deutsch...|           Echology.|
|    conf/lrec/Rapp04|Utilizing the One...|
|conf/icse/BriandDM97|An Investigation ...|
|conf/qshine/Duvvu...|Scheme for Assign...|
|   conf/ewsn/YangH09|A Better Choice f...|
| conf/mm/TangWSWCZ13|LDA: document cl...|
|   conf/icc/XuSYPS10|Spectrum Sensing ...|
+--------------------+--------------------+
only showing top 20 rows

Top 10 most referenced Publications written by at least one Author
associated with Politecnico di Milano
+--------------------+-----+
|                  id|count|
+--------------------+-----+
|conf/sigmod/Selin...|  366|
|journals/tods/Ast...|  244|
|conf/vldb/SellisRF87|  142|
|journals/csur/Bat...|  128|
|conf/sigmod/Abite...|  120|
|conf/sigmod/Blake...|  113|
|conf/sigmod/Agraw...|  111|
|journals/tkde/DeW...|  109|
|conf/vldb/BittonDT83|  107|
|conf/oopsla/Maier...|  103|
+--------------------+-----+
```

6.5.4. Association with the highest number of affiliated authors

Find the name, the country, and the members of the Public Association with the highest number of members affiliated

```python
from pyspark.sql.functions import max, col, count

tmp = df_associations.filter(df_associations.funding == "Public").join(df_authors,
    df_associations.id == df_authors.id_association, "inner") \
.groupBy(df_associations.name, df_associations.country) \
.agg(count(df_associations.name).alias("members"))
max_members = tmp.select(max(tmp["members"])).collect()[0][0]
tmp.filter(tmp["members"] == max_members).show(truncate=False)
```

Result:

```
+-------------------------------+-------+-------+
|name                           |country|members|
+-------------------------------+-------+-------+
|Technical University of Denmark|Denmark|10402  |
+-------------------------------+-------+-------+
```

6.5.5. Times per year in which authors of a certain association were cited at most k-times

Extract the year and the number of times in which authors associated to Politecnico di Milano are cited at most 20 times.

```python
df_authors.filter(col("id_association")=="polimi") \
    .join(df_author_pubs, df_authors.id == df_author_pubs.id_author,"inner") \
    .join(df_pub_pubs,df_pub_pubs.id_cit == df_author_pubs.id_pub,"inner") \
    .join(df_publications,df_publications.id == df_pub_pubs.id_cit,"inner") \
    .groupBy("year").agg(count("id_cit").alias("#times␣citated")) \
    .filter(col("#times␣citated")<=20).show(truncate=False)
```

Result:

```
+----+--------------+
|year|#times cited|
+----+--------------+
|1972|16            |
|2007|3             |
|1967|1             |
|2012|1             |
|1970|6             |
|2004|1             |
|2003|2             |
|2011|2             |
|1971|18            |
|2008|1             |
|2001|2             |
+----+--------------+
```

6.5.6. Context with at least k-publications presented

Find the title and the number of publications of context(s) with at least 75 publications presented in it.

```python
from pyspark.sql.functions import *

df_publications.join(df_contexts, df_publications.id_context == df_contexts.id, "inner
    ") \
.groupBy(df_contexts.title) \
.agg(count(df_publications.title).alias("publications")) \
.filter(col("publications") >= 75) \
.show(truncate=False)
```

Result:

```
+--------------------+------------+
|               title|publications|
+--------------------+------------+
|Database and Expe...|          78|
|Genetic and Evolu...|         100|
|Proceedings of th...|         100|
|Proceedings of th...|          81|
|Fundamentals of C...|         100|
|2000 Design, Auto...|         100|
|Proceedings of th...|         100|
|ACM SIGMOD Digita...|          99|
|IEEE Trans. Knowl...|          75|
|FPL 2007, Interna...|         100|
|High-Performance ...|          82|
|Leveraging Knowle...|          84|
|Proceedings of th...|         100|
|Proceedings of th...|         100|
|New Interfaces fo...|          77|
|Proceedings of th...|         100|
|Proceedings of IP...|         100|
|IEEE Internationa...|          99|
|Human Factors in ...|          76|
|Eighth Internatio...|         100|
+--------------------+------------+
only showing top 20 rows
```

6.5.7. K-authors that have written a publication that was cited at least one time

Find 5 authors that have written at least a publication that have been cited and whose full name starts with "Ma" and ends with "la".

```
df_authors.where(col("id").isin(get_authors_with_publication_cited_asList())) \
         .filter(col("name").like("Ma%la")) \
         .limit(5).show()
```

Result:

```
+---------------+---------------+-----+--------------------+--------------+--------------------+
|             id|           name|orcid|               email|id_association|                urls|
+---------------+---------------+-----+--------------------+--------------+--------------------+
|        15/2141|  Marek Missala| null|marek.missala@pol...|        polimi|                null|
| c/MarkCrovella|  Mark Crovella| null|mark.crovella@mit...|           mit|[http://www.cs.bu...|
|g/MartinGogolla| Martin Gogolla| null|martin.gogolla@un...|         unimi|[http://www.db.in...|
|         16/285|Matthias Nicola| null|matthias.nicola@d...|      deepmind|                null|
|     p/MPatella|  Marco Patella| null|marco.patella@pol...|        poliba|[http://www-db.de...|
+---------------+---------------+-----+--------------------+--------------+--------------------+
```

6.5.8. Average number of author's urls available in each association

Find the average number of authors' website URLs available in each association.

```python
from pyspark.sql.functions import udf
# Defining a custom function that return the number of URLs.
array_len_function = udf(lambda a: 0 if a == None else len(a), IntegerType())

from pyspark.sql.functions import avg

         # Adding a column that contains the count of the number of URLs
df_authors.withColumn("urls_count", array_len_function(df_authors.urls)) \
         .groupBy("id_association") \
         .agg(avg("urls_count").alias("avg_num_urls_per_author")) \
         .show(truncate=False)
```

Result:

```
+--------------+-----------------------+
|id_association|avg_num_urls_per_author|
+--------------+-----------------------+
|deepmind      |0.1464968152866242     |
|dtudk         |0.14526052682176505    |
|poliba        |0.14425783904475295    |
|polimi        |0.14593068535825546    |
|openai        |0.1401243424198948     |
|unimi         |0.14441233140655105    |
|mit           |0.1441630165682727     |
|ethz          |0.13507668108533227    |
+--------------+-----------------------+
```

Variant: retrieve the total number of authors' website URLs available in each association.

```python
from pyspark.sql.functions import sum

```

```
df_authors.withColumn("urls_count", array_len_function(df_authors.urls)) \
         .groupBy("id_association") \
         .agg(sum("urls_count").alias("tot_num_urls_authors")) \
         .show(truncate=False)
```

Result:

```
+--------------+--------------------+
|id_association|tot_num_urls_authors|
+--------------+--------------------+
|deepmind      |1518                |
|dtudk         |1511                |
|poliba        |1486                |
|polimi        |1499                |
|openai        |1465                |
|unimi         |1499                |
|mit           |1514                |
|ethz          |1374                |
+--------------+--------------------+
```

7 | References

- DBLP : https://dblp.org/
- "How to parse DBLP" : https://dblp.org/faq/How+to+parse+dblp+xml.html
- ORCID : https://orcid.org/
- DOI : https://www.doi.org/
- Neo4J Documentation : https://neo4j.com/docs/
- MongoDB Documentation : https://www.mongodb.com/docs/
- MongoDB Compass Documentation : https://www.mongodb.com/docs/compass/current/
- Apache Spark Documentation : https://spark.apache.org/docs/3.3.1/

List of Figures

2.1 ER model 5

4.1 Graph Database snapshot 17
4.2 New author registration 18
4.3 New publication 19
4.4 Update author's web page url 19
4.5 Add new reference 20
4.6 Co-authors from an author's name 21
4.7 Get author by name 22
4.8 Most contributing authors of an association 23
4.9 Number of publications year by year 24
4.10 Degrees of separation between authors 25
4.11 Variant : degrees of separation 25
4.12 Most referenced paper of the year 26
4.13 Shortest path of references from two publications 27
4.14 Couples with the highest number of publications in an association 28
4.15 Variant : Couples with the highest number of publications in an association 29
4.16 Number of publication for each type 29
4.17 Authors that have written at least one incollection and one article, with respective counts, using Neo4j functions 30
4.18 Authors that have written at least one incollection and one article, with respective counts, using aggregations 31
4.19 Top associations for number of "interesting" publications 32
4.20 Number of publications produced by the authors associated to an association 32
4.21 Ranking Algorithm 34
4.22 Ranking Algorithm for Authors 35

5.1 Scientific Paper Document Structure 39
5.2 New Scientific Paper insertion 44
5.3 Adding the affiliation field of the authors of an already existing paper . . . 45

5.4 Update citations of an existing Scientific Paper 46
5.5 Update a document with new page numbers 46
5.6 Adding the ORCID and the website URL to an author given his ID 47
5.7 Titles of inproceedings presented at a certain conference by a given publisher 48
5.8 Scientific Paper with the highest number of authors 48
5.9 Scientific Papers with at least k-citations 49
5.10 Top k-conferences before a certain year 50
5.11 Documents of a certain author contextualized after a certain year 50
5.12 First conferences organized after a certain year 51
5.13 Documents of a certain author with at least k-sections 52
5.14 Most cited Scientific Papers of a certain year 53
5.15 Preview of an article 54
5.16 List of first images 55
5.17 Fetch all the referenced documents 56
5.18 At least one author of the paper must have both the ORCID and the website URL available 57
5.19 All the authors of the paper must have both the ORCID and the website URL available 58
5.20 Average number of images per paper 59