POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Emmanuël Caputo**

Carlo Sgaravatti

Fateme Hajizadekiakalaye

Leonardo Giusti

Alireza Yahyanejad

Group Number: **5**

Academic Year: 2022-2023

Contents

1 | ER Diagram

1.1. Description of the problem

Digital Bibliography & Library Project is a database that publishes a catalogue of bibliographies in computer science. DBLP users have to be affiliated, at least, to an organization (characterized by a city and a country of location) to subscribe. This organization can be considered an agency (containing an id and a name). Each user is identified by an ORCID, the first name, the last name, the country of birth, a scopus id, a loop profile, a personal URL, and a researcher id. Each user can have different roles in the workload of publications (for example author, contributor, co-author, researcher) and can write comments on them. Publications are identified by an id and have a DOI code, an ISBN code, a title, a year of publication, a supplemental material, an abstract, and an index term. Every publication can cite and/or refer to other publications and can have a few external resources based on URLs. Publications belong to one or more categories and can also be searched thanks to keywords (both are defined by a name). Every publication can be part of a conference, published in a journal (specifying volume, issue, starting and ending pages of the publication), and/or a series. Conferences have a date, and a location and can be the successor of another conference. Journals and series are characterized by an ISSN code. Each of these three concepts (journal, conferences, and series) is qualified as a venue. A venue is identified by a DOI code, a title, the status, and the type of document access (public or private). Finally, each venue can be managed by an agency which can publish or fund a publication.

1.2. Assumptions

1. Each Publication could contain a reference to an External resource (for example Wikipedia web page)
2. Each person can comment on a publication, no matter his role (contributor/author).
3. Contributors cannot publish a publication, only agencies can.
4. Each Publication can be part of at most one Series, at most one Conference, and at most one Journal
5. Each Publication can be funded and/or published by a different / the same Agency
6. An Agency does not have to be an Organization (an Agency does not need to have affiliations) but an Organization is an Agency (also an Organization can publish/-fund publications)
7. Each Agency can publish also Venues
8. A person has to be affiliated to an organization in order to contribute to a publication
9. Each viewer can access the text (pdf) of the publication by clicking on the supplemental material (URL of the pdf), but the abstract is still accessible without the need to access the pdf file

1.3. ER description summary

1.3.1. Entities

Entity name	Description
Person	A person is a DBLP user that can contribute to a publication, writes comments, and be affiliated with an organization.
Organization	An organization is an agency that could have multiple affiliations (members).
Agency	An agency is a company that can fund and/or publish a publication. It also can publish a venue.
Publication	A publication is a document (article, PhD thesis, book, ...) written by users and published in the DBLP database.
External resource	An external resource is a link related to a file that is not a publication (for example a Wikipedia link).
Comment	A comment is a text written by a person as a reaction to a publication.
Category	A category is a class of publications that have a common field of study.
Keyword	A keyword is a word that acts as a search key for a publication.
Venue	A venue is a space where publications are shown, for example, a conference, a journal and a series.
Conference	A conference is a venue which represents a formal meeting of people with a shared interest.
Journal	A journal is a venue which represents a collection of multiple publications that deal with a particular subject or professional activity.
Series	A series is a collection of multiple publications that satisfy specific criteria.

Table 1.1: Entities description

1.3.2. Relationships

Relationship name	Attributes	Entities connected	Description
contribute to	**role**: specific role for each person in a publication. For example, author or coauthor	Person & Publication	A person can contribute to 1 or n publication(s) and each publication can be written by 1 or n person
affiliate to	**no attributes**	Person & Organization	A person can be part of 1 or n organization(s) and each organization can have 1 or n member(s)
write	**no attributes**	Person & Comment	A person can write 0,1 or n comments and each comment is related to only 1 person
include in	**no attributes**	Publication & Comment	A publication can include 0,1 or n comments and each comment is related to only 1 publication
fund	**no attributes**	Agency & Publication	An agency can fund 1 or n publication(s) and a publication can be funded by 0 or 1 agency
publish	**no attributes**	Agency & Publication	An agency can publish 1 or n publication(s) and a publication can be published by 0 or 1 agency
refer to	**no attributes**	Publication & Publication	Each publication can refer to or be referred by 0,1 or n publication(s)
cite	**no attributes**	Publication & Publication	Each publication can cite or be cited by 0,1 or n publication(s)

Table 1.2: Relationships description part 1

Relationship name	Attributes	Entities connected	Description
belong to	**no attributes**	Category & Publication	Each publication can belong to 1 or n category(s) and each category can have 1 or n publication(s)
contain of	**no attributes**	Keyword & Publication	Each publication can contain of 0,1 or n keyword(s) and each keyword can be in 1 or n publication(s)
have	**no attributes**	External Resource & Publication	Each publication can have 0,1 or n external resource(s) and each external resource can be in 1 or n publication(s)
publish by	**no attributes**	Agency & Venue	Each agency can publish 1 or n venue(s) and each venue can be published by 1 and only 1 agency
part of	**no attributes**	Conference & Publication	Each publication can be part of 0 or 1 conference and each conference can discuss 1 or n publication(s)

Table 1.3: Relationships description part 2

Relationship name	Attributes	Entities connected	Description
publish in	**pages - start/end** (both INT): starting and ending pages inside the journal. **volume** (STRING): identifies a group of editions of the journal (usually all editions of the journal in a year). **issue** (STRING): a numeric string that identifies the specific journal inside a volume of journals. **year** (INT): the year on which the edition of the journal is published	Publication & Journal	A publication could be published in 0 or 1 journal and each journal can publish 1 or n publication(s).
is successor	**no attributes**	Conference & Conference	Each conference can have 0 or 1 successor and/or be the successor of a 0 or 1 conference
is in	**Volume** (STRING): identifies a/ a group of publication(s) inside the series	Publication & Series	Each publication can be in 0 or 1 series and each series can contain of 1 or n publication(s)

Table 1.4: Relationships description part 3

1.3.3. Attributes

Attribute name	Type	Description
ORCID	STRING	ORCID is a unique digital identifier (community-driven) which identifies authors in all sources
first_name	STRING	The first name of the person
last_name	STRING	The last name of the person
country	STRING	The country of birth of the person
url	STRING	The URL of personal website
researcher id	STRING	Another unique identifier for scientific authors, owned by the "Web of Science" platform.
loop profile	STRING	A link to the profile of the author in the Loop platform (which makes authors and their research visible on multiple websites via a universal Loop profile).
scopus id	STRING	An identifier that is used in the database Scopus (all the publications of each author are grouped together to the Scopus author profile and allows citation metrics for each author).

Table 1.5: Person entity attributes

Attribute name	Type	Description
url	STRING	URL of the external resource

Table 1.6: External Resource entity attributes

Attribute name	Type	Description
timestamp	DATETIME	The moment on which the comment was written
text	STRING	The content the comment

Table 1.7: Comment entity attributes

Attribute name	Type	Description
ID	STRING	The unique identifier of the publication in the database
title	STRING	The title of the publication
year	INT	The year on which the publication is published
index term	STRING	A single word or a brief phrase that describes the document's content
ISBN	STRING	ISBN is an external key which identify publications on paper
Supplemental material	STRING	The URL where the pdf file can be downloaded
abstract	STRING	The abstract of the content of a publication
DOI	STRING	DOI is an external key which identify the publication online

Table 1.8: Publication entity attributes

Attribute name	Type	Description
id	STRING	The identifier of the agency
name	STRING	The name of the agency

Table 1.9: Agency entity attributes

Attribute name	Type	Description
city	STRING	The city of the implantation of the organization
country	STRING	The country of the implantation of the organization

Table 1.10: Organization entity attributes

Attribute name	Type	Description
name	STRING	The identifier of the category

Table 1.11: Category entity attributes

Attribute name	Type	Description
name	STRING	The identifier of the keyword

Table 1.12: Keyword entity attributes

Attribute name	Type	Description
DOI	STRING	DOI is an external key which identify venue online
title	STRING	title of the venue where the publication was published
document access	STRING	possibilities : public or private
status	STRING	possibilities : open or close

Table 1.13: Venue entity attributes

Attribute name	Type	Description
date	STRING	The date on which the conference is discussed
location	STRING	The location where the conference is discussed

Table 1.14: Conference entity attributes

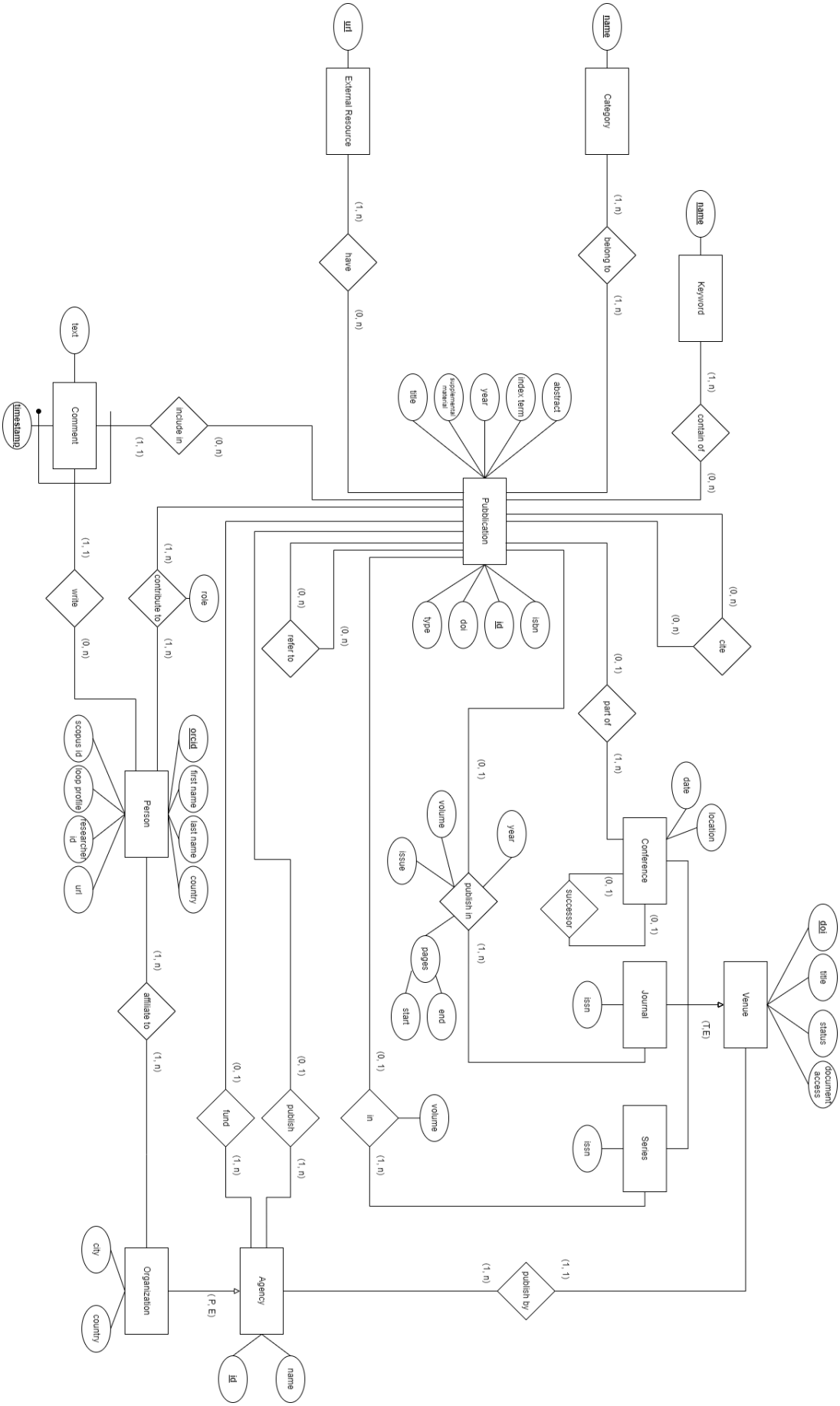
Attribute name	Type	Description
ISSN	STRING	ISSN is used to identify serial publications; in this case, it identifies the entire journal (all the editions)

Table 1.15: Journal entity attributes

Attribute name	Type	Description
ISSN	STRING	An identifier of all the publications of the series

Table 1.16: Series entity attributes

1.4. ER schema


url
External Resource
name
Category
name
Keyword
have
belong to
contain of
(1, n)
(0, n)
text
Comment
timestamp
include in
(1, 1)
(0, n)
title
supplementary material
year
index_term
abstract
Publication
write
contribute to
role
refer to
cite
type
doi
id
isbn
part of
scopus id
loop profile
researcher id
url
Person
orcid
first name
last name
country
affiliate to
Organization
city
country
(P, E)
Agency
id
name
fund
publish
in
volume
publish in
volume
year
issue
pages
start
end
Conference
date
location
successor
Journal
issn
Series
issn
Venue
(T, E)
doi
title
status
document access
publish by

2 | Neo4j

2.1. Dataset Description

We have used the **DBLP-Citation-network V13** dataset, downloaded from the Aminer website, as the reference dataset for our Neo4j implementation. This choice was made because the dataset was more complete with respect to the DBLP XML file. Indeed, this file contains also the field of study of a publication, its keywords and external resources. This has permitted us to make more different queries (instead of performing queries only on a small subset of entities). The dataset contains information extracted from DBLP, ACM and MAG sources and is organized in a JSON file. Table 2.1 shows the structure of the dataset; the attributes that have a "?" at the end are the ones that are sometimes missing in the dataset. However, some of the attributes were also sometimes empty (null or ""), even if they were present. The next section describes how some of these attributes were treated, before importing the dataset in Neo4j.

Name	Type	Description
id	string	id of the publication
title	string	title of the publication
authors	Array<Object>	authors of the publication
author.name?	string	name of the author
author.id?	string	orcid of the author
author.org?	string	affiliation of the author
author.oid?	string	id of organization of the author
venue?	Object	venue on which the publication is published
venue.id?	string	id of the venue
venue.name_d?	string	name of the venue
venue.publisher?	string	publisher of the venue
venue.type?	int	type of venue
year	int	year on which the publication is published
keywords	Array<string>	keywords of the publication
fos	Array<string>	fields of study of the publication
page_start	string	starting page of the publication in the venue
page_end	string	ending page of the publication in the venue
doc_type	string	type of publication (article, phd thesis, ...)
publisher?	string	publisher of the publication
volume	string	volume of the venue
issue	string	issue number of the venue (journal)
issn	string	issn of the venue
ibsn	string	isbn of the publication
doi	string	doi of the publication
pdf	string	url on which the pdf of the publication can be found
url	Array<string>	external resources of the publication
references	Array<string>	id of the publications that are referenced
abstract	string	abstract of the publication

Table 2.1: The dataset structure

2.2. Data upload process

Before importing the data in Neo4j, the dataset was preprocessed with a python script to:

- Select only a small part of the publications that are present in the dataset. In particular, only the first 100 publications that contain some references to other publications, plus all the publications that were referenced by those were kept (for a total of more than 1000 publications)
- Remove authors that don't have both the name and the id
- Assign an ORCID to the authors that don't have it, an id to the organizations that do not have it, an id to venues that don't have it and an id to the publishers of both the publications and venues
- Split the name of the author into first name and last name to match our ER schema
- Make fields of study and keywords case insensitive to remove duplicates that differ only for some capital letters
- Change the venue.type field from integer to string, where the string can be "conference", "journal" or "series" (since different integers were referring to the same type)

The following cypher command was used to import the resulting dataset in Neo4j:

```
CALL apoc.load.json("dblpv13-preprocessed.json") YIELD value AS p
MERGE (pub:Publication {id: p._id})
SET pub.title = p.title, pub.year = p.year, pub.doi = p.doi,
    pub.timestamp = p.timestamp, pub.abstract = p.abstract,
    pub.supplemental_material = p.pdf, pub.type = p.doc_type
FOREACH (a IN p.authors |
        MERGE (author:Person {orcid: a._id})
           ON CREATE SET author.firstname = a.firstname,
                author.lastname = a.lastname
           MERGE (pub)<-[c:CONTRIBUTES_TO]-(author)
           FOREACH (_ IN CASE WHEN a.org IS NOT NULL THEN [1] ELSE [] END |
                MERGE (org:Agency:Organization {id: a.oid})
                ON CREATE SET org.name = a.org
                MERGE (author)-[af:AFFILIATES_TO]->(org)
           )
```

```
)
FOREACH (k IN p.keywords |
           MERGE (keyword:Keyword {name: k})
            MERGE (pub)-[co:CONTAINS_OF]->(keyword)
)
FOREACH (f IN p.fos |
            MERGE (category:Category {name: f})
            MERGE (pub)-[co:BELONGS_TO]->(category)
)
FOREACH (u IN p.url |
            MERGE (ex_res:ExternalResource {url: u})
            MERGE (pub)-[h:HAS]->(ex_res)
)
FOREACH (r IN p.references |
            MERGE (ref_pub:Publication {id: r})
            MERGE (pub)-[h:REFERS_TO]->(ref_pub)
)
MERGE (publisher:Agency {id: p.publisher_id})
ON CREATE SET publisher.name = p.publisher
MERGE (publisher)-[pu:PUBLISH]->(pub)
FOREACH (v IN CASE WHEN p.venue IS NOT NULL AND p.venue._id IS NOT NULL
        THEN [p.venue] ELSE [] END |
            MERGE (venue: Venue {id: v._id})
            ON CREATE SET venue.title = v.name_d
            FOREACH (_ IN CASE WHEN v.type = "conference" THEN [1] ELSE [] END |
                SET venue:Conference
                MERGE (pub)-[part_of:PART_OF]->(venue)
            )
            FOREACH (_ IN CASE WHEN v.type = "journal" THEN [1] ELSE [] END |
                SET venue:Journal
                SET venue.issn = p.issn
                MERGE (pub)-[published_in:PUBLISHED_IN]->(venue)
                ON CREATE SET published_in.volume = p.volume,
                        published_in.issue = p.issue,
                        published_in.page_start = p.page_start,
                        published_in.page_end = p.page_end
            )
```

```
        FOREACH (_ IN CASE WHEN v.type = "series" THEN [1] ELSE [] END |
            SET venue:Series
            SET venue.issn = p.issn
            MERGE (pub)-[in_:IN_]->(venue)
            ON CREATE SET in_.volume = p.volume
        )
        MERGE (venue_publisher:Agency {id: v.publisher_id})
        ON CREATE SET venue_publisher.name = v.publisher
        MERGE (venue)-[is_pub:IS_PUBLISHED_BY]->(venue_publisher)
)
```

Since comments were not present in the dataset, they were added manually for some publications with the following command (where x, y, z and t are strings):

```
MATCH (pu:Publication), (pe:Person)
WHERE pu.id = x AND pe.orcid = y
CREATE (c:Comment {timestamp: datetime(z), text: t}),
        (pu)<-[:INCLUDE_IN]-(c)<-[:WRITES]-(pe)
```

Also some "fake" persons were created in order to have a writer of the comments.

2.3. Graph diagram

After the data upload has been completed, the resulting graph can have the following type of node labels:

- **Publication**
- **Person** - a person that can contribute to / comment on publications
- **Agency** - an agency that can publish/fund publications and/or venues
- **Agency:Organization** - an agency that is also an organization
- **Venue:Journal/Conference/Series** - the venue that contains publications
- **Category** - a field of study of a publication
- **ExternalResource** - an external resource referenced by a publication
- **Keyword** - a keyword of a publication
- **Comment** - a comment written by a person for a publication

Since some attributes were not present in the dataset, the following attributes were not considered for the queries: loop profile, URL, scopus id, researcher id and country for Person nodes; status and the document access for Venue nodes; date and location for Conference nodes; city and country for Organization nodes. Figure 2.1 is an example of a graph present in the database; for space convenience, not all attributes of a publication are reported in it (indeed, the type of document and the supplemental material are missing).

Figure 2.1: An example of graph

2.4. Commands

1. **Create a publication which is published by agency x and is funded by agency y.**
   As we mentioned in the assumptions, a publication can be published and/or funded by different/same agencies. This command will create a publication which is published by agency "a" and is funded by agency "b". First, we match two agencies in the database where their corresponding ids are x and y respectively and call them "a" and "b". Then, we create a new publication called "p" and form two relationships "fund" and "publish" between publication "p", agency x and agency y. Finally, we return the type of the two relationships.

```
MATCH (a:Agency), (b:Agency)
WHERE a.id = x AND b.id = y
CREATE (b)->[r1:FUNDS]-(p:Publication{id: "i", doi: "d", title: "t",
        isbn: "isbn", year: y, abstract: "a",  index_term: "it"})
        <-[r2:PUBLISH]-(a)
RETURN type(r1), type(r2)
```

2. **Delete Affiliate relationship between a person and organization.**
   A person who is affiliated with an organization may leave that organization. So we should be able to remove its corresponding relationship. First, we match a person whose ORCID is x and is in an "affiliates to" relationship. Then, we delete the relationship.

```
MATCH (p:Person)-[r:AFFILIATES_TO]-(o:Organization)
WHERE p.orcid = "x"
DELETE r
```

3. **Update title of the publication which has x as DOI.**
   We should be able to update a publication's attributes like "title", "year", etc. Because they might have typos or for some other reasons. This command, first of all, matches a publication which has x as DOI in the database. Then, it changes its title by means of the SET command. Finally, it returns the modified publication as the result.

```
MATCH (p: Publication)
WHERE p.doi = "x"
SET p.title = "y"
RETURN p
```

4. **Update the comment that was written by x on publication y on date z.**
   A person can comment on publications. He also should be able to edit his comment. Because he might change his mind about what he wrote or he might have typos. First, we match a person whose ORCID is x called "p", a publication whose DOI is y called "a" and a comment which is written by person "p" on publication "a" at time "z". Then, we change the comment's text using the set command. Finally, we return the modified comment.
   The parameter z is a string that must be a valid representation of a DateTime neo4j type of attribute.

```
MATCH (p:Person)-[:WRITES]-(c:Comment)-[:INCLUDE_IN]-(a:Publication)
WHERE p.orcid = "x" AND a.doi = "y" AND c.timestamp = datetime("z")
SET c.text = "modified text"
RETURN c.text
```

5. **Attach a keyword of a publication to all publications that are referenced by that publication.**
   We made this example to show how we can create a new relationship. In this case, we are creating a "contains of" relationship between some publications and a specific keyword. First, we match a keyword which is named y, a publication "p1" which has x as DOI and all the publications "p2" that "p1" refers to. Then, we create (if it does not exist) our desired relationship(s) named "consist of" between "k" and "p2" using the merge command. Finally, we return the type of relationship "r2" as the result.

```
MATCH (k:Keyword)-[]-(p1:Publication)-[r1:REFERS_TO]->(p2:Publication)
WHERE p1.doi = "x" AND k.name = "y"
MERGE (k)<-[r2:CONTAINS_OF]-(p2)
RETURN type(r2)
```

2.5. Queries

1. **Find persons who collaborated to a given publication that was discussed in a given conference.**
   This query covers the "3 nodes, conditions" example. Here, we want to show the relationship between some people who contribute to a publication (as an author or coauthor) and that publication is discussed at a conference. First, we match people "p" that contribute to a publication which has x (which is a string) as DOI and was published in year y (an integer), and is part of the conference which has y (a string)

as DOI. Finally, we return the title of the conference, the name of the contributors and the title of the publication.

```
MATCH (p:Person)-[:CONTRIBUTES_TO]-(a:Publication)
        -[:PART_OF]-(c:Conference)
WHERE c.id = x AND a.year = y AND a.doi = z
RETURN c.title, p.firstname, p.lastname, a.title
```

	c.title	p.firstname	p.lastname	a.title
1	"International Conference on Data Engineering"	"Carla"	"Schlatter Ellis"	"A Sampling-Based Approach to Optimizing Top-k Queries in Sensor Networks"
2	"International Conference on Data Engineering"	"Rebecca"	"Braynard"	"A Sampling-Based Approach to Optimizing Top-k Queries in Sensor Networks"
3	"International Conference on Data Engineering"	"Adam"	"Silberstein"	"A Sampling-Based Approach to Optimizing Top-k Queries in Sensor Networks"
4	"International Conference on Data Engineering"	"Jun"	"Yang"	"A Sampling-Based Approach to Optimizing Top-k Queries in Sensor Networks"
5	"International Conference on Data Engineering"	"Kamesh"	"Munagala"	"A Sampling-Based Approach to Optimizing Top-k Queries in Sensor Networks"

Figure 2.2: Result of query 1 for $y = 2006$ and $z =$ "10.1109/ICDE.2006.10"

2. **Find title of journals which have more than x publications and their publications consist of the keyword named y.**
   This query covers the "3 nodes, conditions, aggregations" example. Here we want to show the relationship between publication, journal and category entities. First, we match the publications which are published in any journal and which belong to a category named y (a string). After that, we sort the journals based on the number of their publications by decreasing order and select only those which have more than x (string) publications. Finally, we return the title of the journals, the name of the category and the number of publications of each journal.

```
MATCH (j:Journal)-[:PUBLISHED_IN]-(p:Publication)
        -[:BELONGS_TO]-(c:Category)
WHERE c.name = y
WITH j.title AS journalTitle, c.name as categoryName, count(p) as countPub
ORDER BY countPub DESC
WHERE countPub >= x
RETURN journalTitle, categoryName, countPub
```

	journalTitle	categoryName	countPub
1	"Journal of Artificial Intelligence Research"	"computer science"	7
2	"Operations Research"	"computer science"	6

Figure 2.3: Result of query 2 for $x = 3$ and $y =$ "computer science"

3. **Find the oldest journals published by the "North-Holland" agency.**
   In the following query, which is an example of "3 nodes, conditions, aggregations", we want to find the oldest journals (where the oldness of a journal is the publishing year of the oldest publication that the journal contains) of an agency named "North-Holland". First, we match all publications that are in a relationship with a journal published by "North-Holland". Then, we group the result by journal and for each journal, we keep track of the year of the oldest publication in it. The result of this phase is sorted by the year and by the title of the journal. Finally, the top 5 journals are returned.

```
MATCH (pu:Publication)-[:PUBLISHED_IN]-(j:Journal)
        -[:IS_PUBLISHED_BY]-(a:Agency)
WHERE a.name = "North-Holland"
WITH j, MIN(pu.year) AS oldestPub
ORDER BY oldestPub ASC, j.title ASC
LIMIT 5
RETURN j.title, oldestPub
```

	j.title	oldestPub
1	"Electronic Notes in Discrete Mathematics"	2001
2	"Information Sciences"	2001
3	"Pattern Recognition Letters"	2001
4	"ACM Special Interest Group on Data Communication"	2004
5	"IEEE Transactions on Circuits and Systems II: Express Briefs"	2004

Figure 2.4: Result of query 3

4. **Find publications of a certain category that have at least 5 references to other publications**
   The following query finds publications which belong to category x and refer to

at least 5 other publications. The query first finds all the publications "p1" that belong to a category named x (a string), then it aggregates the result using p1, calculates the number of referenced publications in a variable named "refNumber" and collects all referenced publications "p2" in a list. After, the query selects only the publications "p1" that have "refNumber" greater or equal to 5 and returns all publications "p1" and, for each of them, all the referenced publications.

```
MATCH (c:Category)<-[:BELONGS_TO]-(p1:Publication)
        -[:REFERS_TO]->(p2:Publication)
WHERE c.name = x
WITH p1, COUNT(p2) AS refNumber, collect(p2) AS ref
WHERE refNumber >= 5
UNWIND ref AS pu2
RETURN p1, pu2
```

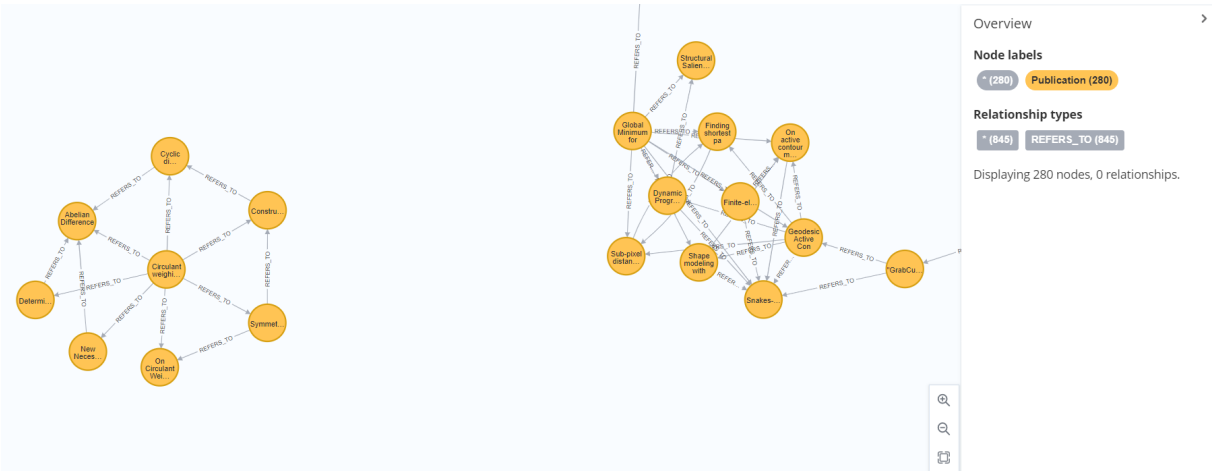


Figure 2.5: Small part of the result of query 4 for x = "mathematics"

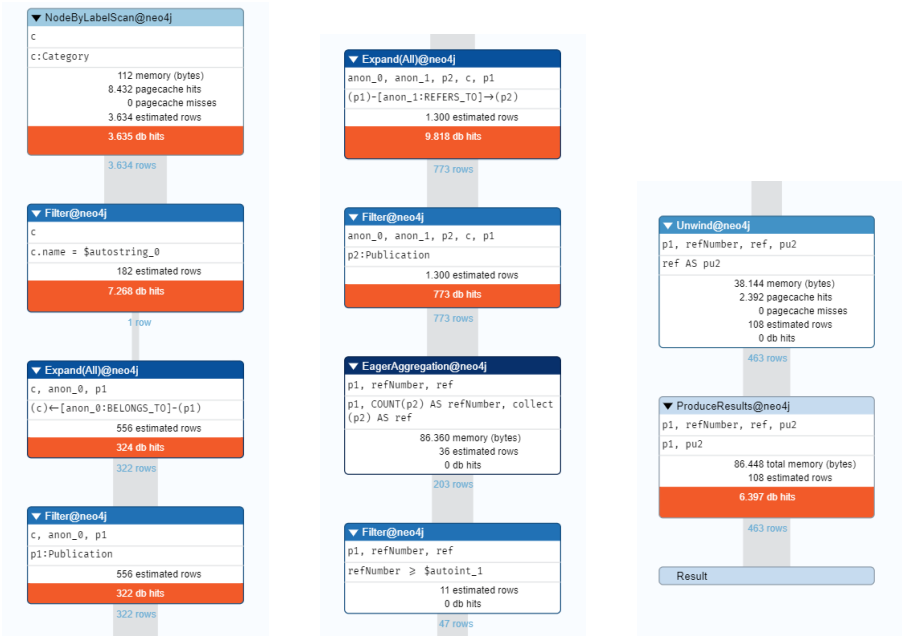


Figure 2.6: Profile result for query 4

5. **Given an organization and a name of a person, find publications of the person that are published in a certain conference and where the number of comments is 5.**

   In this query, we want to find all persons with x as their first name and y as the last name who are affiliated with the organization with id t. Also, we want to return three publications to which the above person(s) contributed, published in a conference with id z and with a number of comments equal to 5. After matching the nodes that have the specified property the query performs an aggregation to count the number of comments for each publication and to filter only publications with 5 comments, then it returns three of those publications.

   All parameters x, y, z and t are strings.

```
MATCH (o:Organization)-[:AFFILIATES_TO]-(p:Person)-[:CONTRIBUTES_TO]
      -(a:Publication)-[:PART_OF]-(j:Conference),
      (a)-[:INCLUDE_IN]-(c:Comment)
WHERE p.firstname = x AND p.lastname = y AND j.id = z AND o.id = t
WITH o, j, a, COUNT(c) AS numComments
WHERE numComments = 5
WITH o, j, a, numComments
LIMIT 3
```

```
RETURN o.name, j.title, a.title, numComments
```

"o.name"	"c.title"	"a.title"	"numComments"
"Department of Mathematics, National University of Singapore, Singapore, Republic of Singapore 119260"	"J. Comb. Theory, Ser. A"	"Cyclic relative difference sets with classical parameters"	5
"Department of Mathematics, National University of Singapore, Singapore, Republic of Singapore 119260"	"J. Comb. Theory, Ser. A"	"Constructions of relative difference sets with classical parameters and circulant weighing matrices"	5

Figure 2.7: An example of result for query 5 considering the author "Ka Hin Leung"

6. **Find the shortest path between publication x and y.**
   This query finds the shortest path in terms only of the "REFERS_TO" relationship from the publication x to the publication y (i.e. the shortest way someone can find the publication y starting from the publication x and navigating only the references of the publications). Since the constraints on the shortest path are referring only to the type of relationship, the conditions can be evaluated while searching for the path; therefore, the query performs a Fast Bidirectional Breadth-first Search Algorithm.

```
MATCH (p1:Publication), (p2:Publication),
        p = shortestpath((p1)-[:REFERS_TO*]->(p2))
WHERE p1.doi = x AND p2.doi = y
RETURN p
```

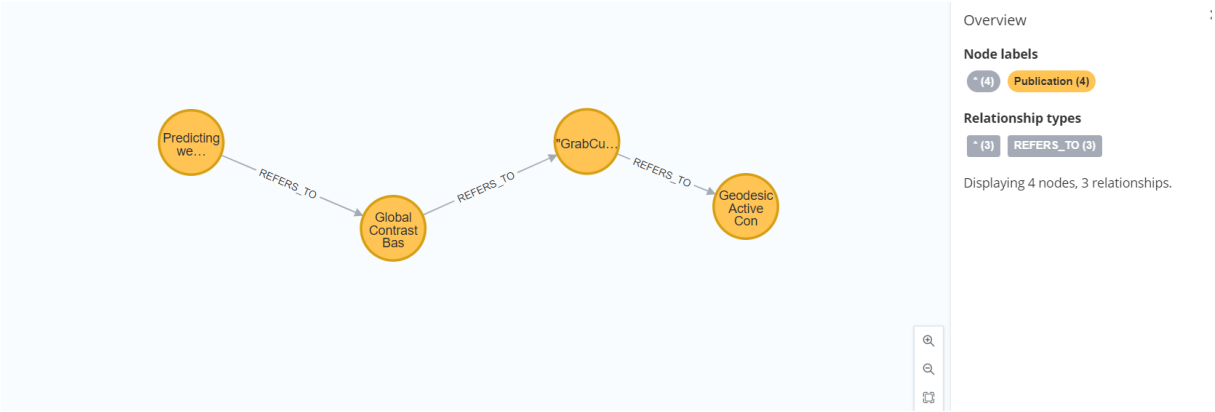


Figure 2.8: Result of query 6 for x = "10.1145/2522848.2522853" and y = "10.1023/A:1007979827043"

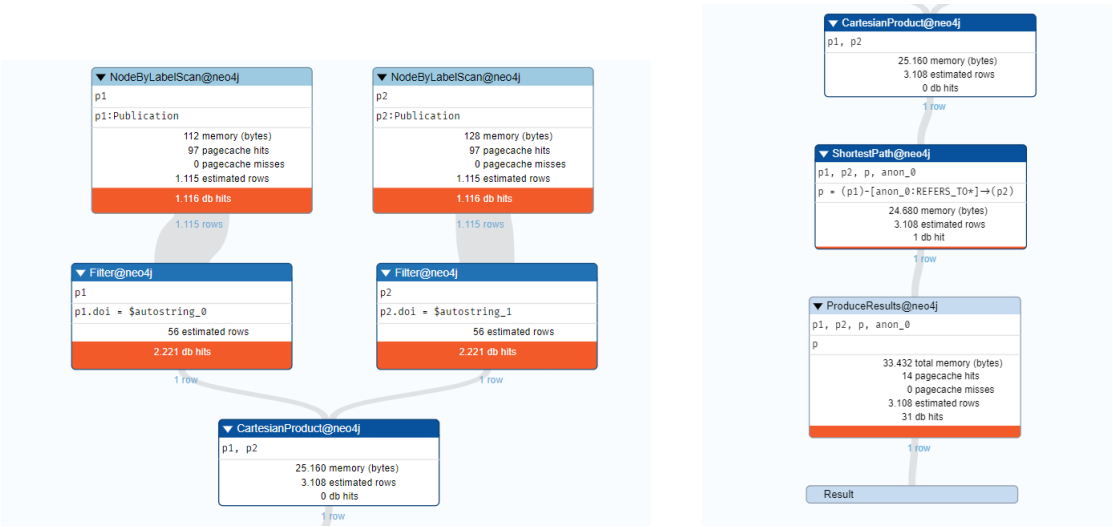


Figure 2.9: Profile result for query 6

7. **Find the quickest way a person can enter in touch with an author of a publication that contains a certain keyword.**
   Here we want to find how a person, that has x as ORCID, can enter in touch with a person that has contributed to a publication that has a keyword named y by passing through only a collaborator "relationship" (x can ask only his collaborators, then his collaborators, if needed, can ask their collaborators, ...). First, we match the person "pe1" that has x as ORCID and all the persons "pe2" that have contributed to at least a publication that contains the keyword named y. For each "pe2" we find the shortest path from "pe1" considering only the "CONTRIBUTES_TO" relationship (the shortest path function performs a Fast Bidirectional Breadth-first Search Algorithm). Then we calculate the length of the resulting path and we return only the shortest one.

```
MATCH (pe1:Person), (pe2:Person)-[]-(pu:Publication)-[]-(k:Keyword),
    p = shortestpath ((pe1)-[:CONTRIBUTES_TO*]-(pe2))
WHERE pe1.orcid = x AND k.name = y AND pe1 <> pe2
WITH p, pu, k, length(p) AS pathLength
ORDER BY pathLength ASC
LIMIT 1
RETURN p, pu, k
```

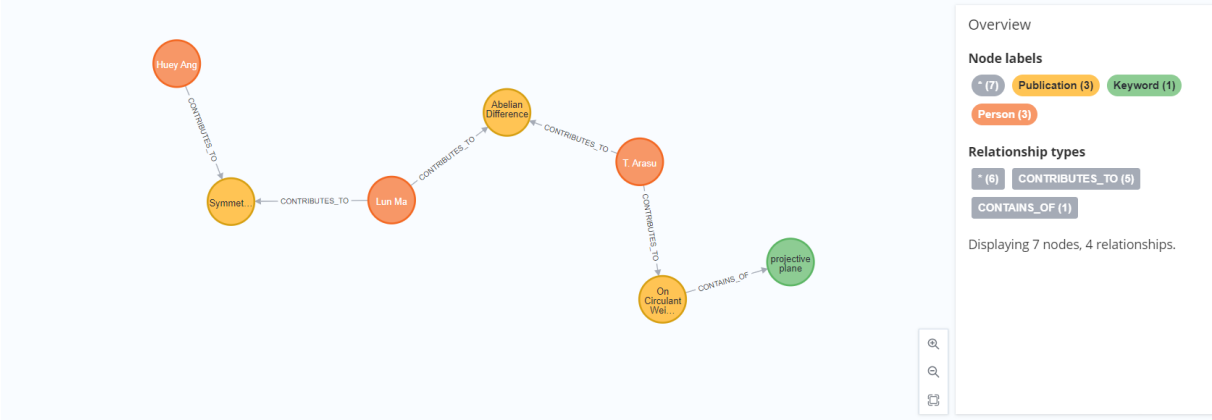
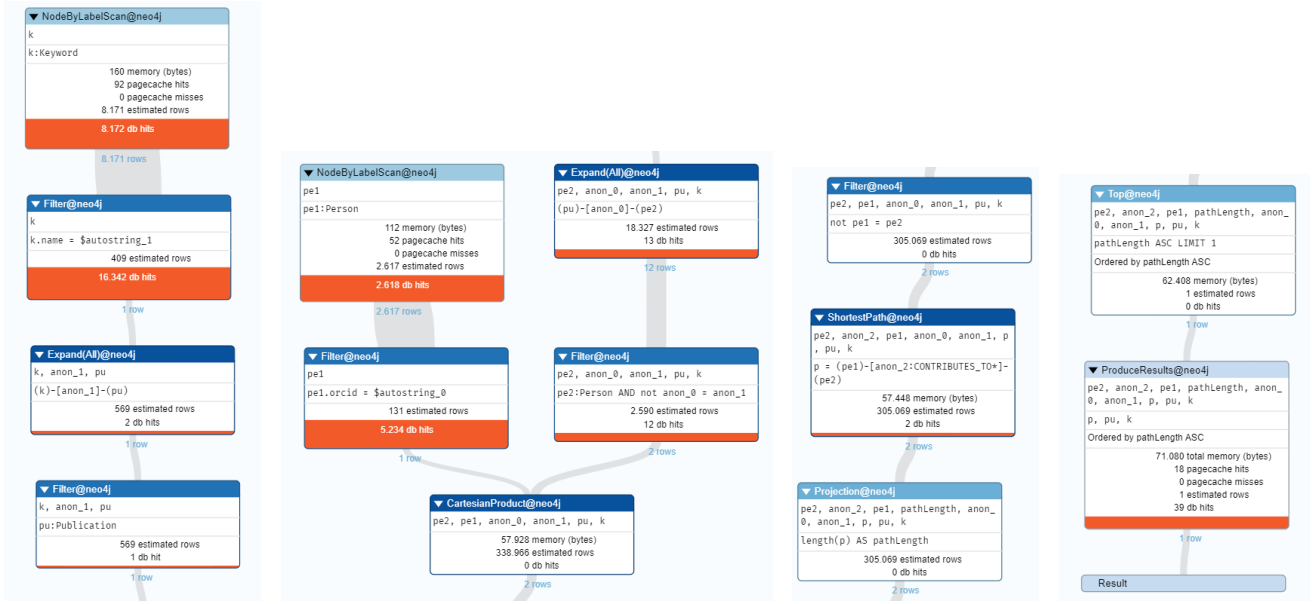


Figure 2.10: Result of query 7 for considering x as the orcid of "Min Huey Ang" and y equal to "projective plane"



(a) (b) (c) (d)

Figure 2.11: Profile result for query 7

8. **Find all the collaborators of a person.**
   This query finds all the persons who have collaborated for a certain publication with a person that has x as the first name and y as the last name (both x and y are strings).

```
MATCH (p1: Person)-[:CONTRIBUTES_TO]-(pu:Publication)
        -[:CONTRIBUTES_TO]-(p2:Person)
WHERE p1.firstname = x AND p1.lastname = y AND p1 <> p2
```

```
RETURN p1, p2, pu
```

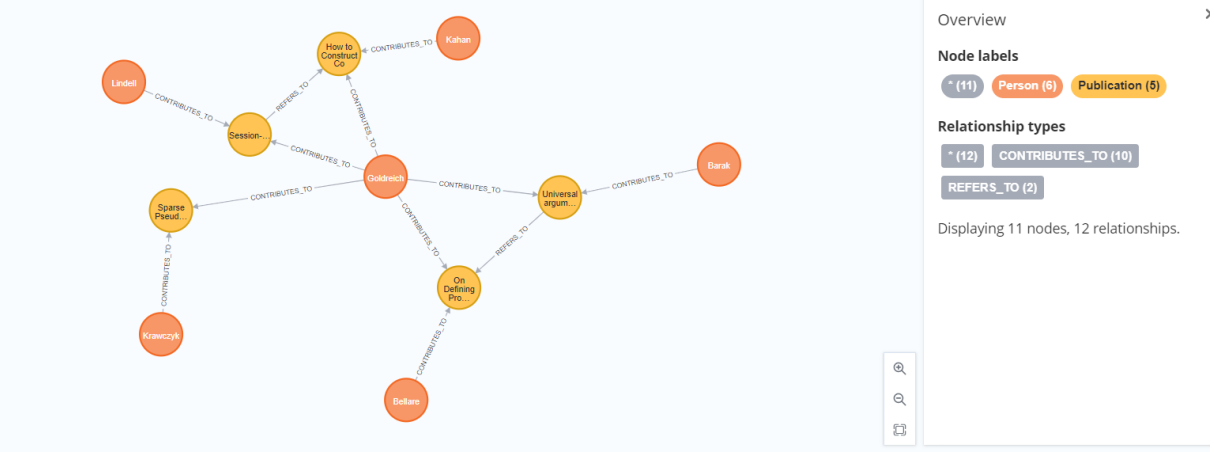


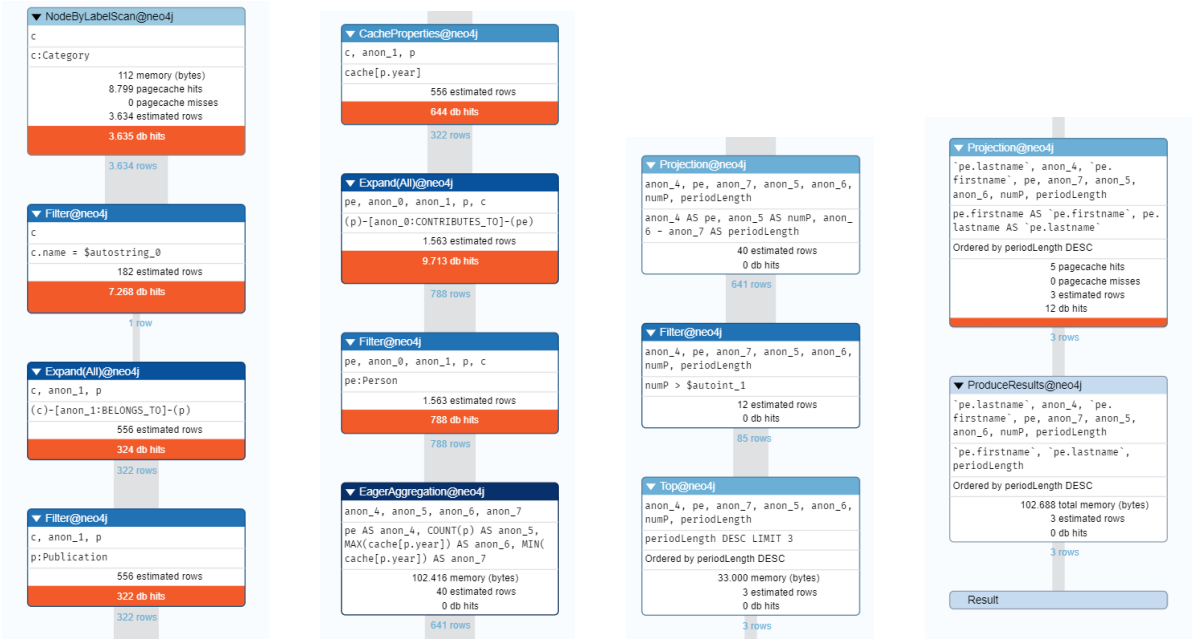
Figure 2.12: Result of query 8 for x = "Oded" and y = "Goldreich"

9. **Find the authors who have published for the longest period of time in the category of "mathematics".**
   We want to find the authors that have been active (i.e. have published) for the longest period in the category of mathematics (this implies that an author needs to contribute to at least 2 articles in the field of "mathematics").
   The query first finds all authors "pe" that have published some publications "p" that belong to the category "mathematics". Then it groups the results by "pe", counts the number of publications and calculates the active period length for that person "pe"; only the ones that have more than 1 publication are kept. The result of the previous aggregation is then ordered by the length of the period; at the end, only the top three persons are returned.

```
MATCH (pe:Person)-[:CONTRIBUTES_TO]-(p:Publication)
        -[:BELONGS_TO]-(c:Category)
WHERE c.name = "mathematics"
WITH pe , COUNT(p) AS numP, MAX(p.year) - MIN(p.year) AS periodLength
WHERE numP > 1
WITH pe, periodLength
ORDER BY periodLength DESC
LIMIT 3
RETURN pe.firstname, pe.lastname, periodLength
```

	pe.firstname	pe.lastname	periodLength
1	"George"	"L. Nemhauser"	21
2	"Oded"	"Goldreich"	16
3	"Michel"	"Grabisch"	11

Figure 2.13: Result of query 9



(a) (b) (c) (d)

Figure 2.14: Profile result for query 9

10. **Find the 5 most multidisciplinary persons in a given organization.** This query finds the first 5 persons who have commented on most publications of different categories in a given organization. The parameter x, which is a string, is the name of the organization.

```
MATCH (c:Category)-[:BELONGS_TO ]-(p:Publication )
        -[:INCLUDE_IN]-(co:Comment)-[:WRITES ]-(pe:Person )
        -[:AFFILIATE_TO ]-(o:Organization)
WHERE o.name = x
WITH pe, o, COUNT( DISTINCT c) AS categoryNumber
ORDER BY categoryNumber DESC
```

```
LIMIT 5
RETURN pe.firstname, pe.lastname, o.name, categoryNumber
```

	pe.firstname	pe.lastname	o.name	categoryNumber
1	"Mario"	"Rossi"	"Università di Udine"	21
2	"John"	"Doe"	"Università di Udine"	12
3	"Marco"	"Verdi"	"Università di Udine"	11
4	"Lorenzo"	"Rosa"	"Università di Udine"	11
5	"Chris"	"Williams"	"Università di Udine"	10

Figure 2.15: Result of query 10 for x = "Università di Udine"

3 | MongoDB

3.1. Description of the structure of the document

Documents are identified by an Object id. Each of them contains a title, an abstract and an array of keywords. The contributors of a document are stored with an array of authors which have as attributes an orcid, a first_name, a last_name, an email, a bio and an array of affiliations (which has a name, a country and a city as attributes). Some details about a publication are also inserted inside the document. We can find a Journal (which has a title and an ISSN as attributes), a volume, a number, a date and pages (which has a start_page and an ending_page as attributes). As for the content of the publication, we have some sections. These contain a title, an array of paragraphs, an array of subsections (which has a title, an array of paragraphs also and an array of figures which has its own caption and image URL) and also an array of figures (which has also its own caption and image URL). Each publication has a bibliography which is composed of an array of external references and an array of publication references (which are identified by the Object ids regarding the publications).

3.1.1. Attributes description

Attribute name	Description
_id	The document identifier.
authors._orcid	ORCID is a unique digital identifier (community-driven) which identifies authors in all sources.
authors.first_name	The first name of the person.
authors.last_name	The last name of the person.
authors.email	The email of the person.
authors.bio	The bio of the person.
authors.affiliations.name	The name of the affiliations regarding a person.
authors.affiliations.country	The country of the affiliations regarding a person.
authors.affiliations.city	The city of the affiliations regarding a person.
title	The title of the publication.
abstract	The abstract of the publication.
keywords	An array of keywords regarding a publication.
details.journal.title	The title of the journal.
details.journal.issn	ISSN is used to identify serial publications; in this case, it identifies the entire journal (all the editions)
details.volume	identifies a group of editions of the publication (usually all editions of the publication in a year).
details.date	The date when the publication was published.
details.number	A numeric string that identifies the specific publication inside a volume of publications.
details.pages.start_page	Starting page of the publication.
details.pages.ending_page	Ending page of the publication.
sections.title	Title of the section in particular.
sections.paragraphs	The paragraph regarding the section in particular.

Table 3.1: attributes part 1

sections.subsections.title	Title of the subsection regarding a section in particular.
sections.subsections.paragraphs	Paragraph of the subsection regarding a section in particular.
sections.subsections.figures.image_url	The URL of the image regarding a subsection in particular.
sections.subsections.figures.caption	The caption of the image regarding a subsection in particular.
sections.figures.image_url	The URL of the image regarding a section in particular.
sections.figures.caption	The caption of the image regarding a section in particular.
bibliography.external_ref	An external resource is a link related to a file that is not a publication (for example a Wikipedia link).
bibliography.publication_references	An array containing the object ids referring to other publications.

Table 3.2: attributes part 2

```
{
        _id :
        authors : [
                {
                _orcid :
                first_name :
                last_name :
                affiliations [{
                        name :
                        city :
                        country :
                           }]
                email :
                bio :
                }]
        title :
        abstract: :
        keywords : [  , ]
        details {
                journal {
                        title : ,
                        issn :
                        } ,
                volume :
                number :
                date :
                pages {
                        start_page :
                        ending_page :
                }
        }
        sections [ {
                title : ,
                paragraphs : [ , ]
                subsections : [ {
                        title : ,
                        paragraphs [ , ]
                        figures : [ {
                                image url : ,
                                caption : ,
                        } ]
                        } ]
                figures [ {
                        image url : ,
                        caption : ,
                        } ]
        } ]
        bibliography{
                external_ref : [ , ]
                publication_references : [ , ]
                }
}
```

Figure 3.1: Generic document structure

```
{
        _id : "12233"
        authors : [
                {
                _orcid : "IDGS1928"
                first_name : "Carlo"
                last_name : "Sgaravatti"
                affiliations [{
                        name : "Politecnico di Milano"
                        city : "Milano"
                        country : "Italia"
                           }]
                email : "carlo.sgaravatti@fakemail.com"
                bio : "Hi, i'm a Computer Science student"
                }]
        title : "AI and Passion in Action"
        abstract: : "text"
        keywords : ["AI" , "learning"]
        details {
                journal {
                        title : "The guardians" ,
                        issn : 2049-3036
                        } ,
                volume : 4
                number : 15
                date : 22/08/2021
                pages {
                        start_page : 27
                        ending_page : 45
                }
        }
        sections [ {
                title : "deep learning" ,
                paragraphs : [ "Paragrafo 1" ,  "Paragrafo 2 " ]
                subsections : [ {
                        title : "object classification with YOLO",
                        paragraphs [ , ]
                        figures : [ {
                                image url : "https://www.bing.com/images/search?view=detailV2" ,
                                caption : "CIAO"
                        } ]
                        } ]
                figures [ {
                        image url : "https://www.bing.com/images/search?view=detailV2" ,
                        caption : "WOW"
                        } ]
        } ]
        bibliography{
                external_ref : [ "https://it.wikipedia.org/wiki/Italia" ,
"https://en.wikipedia.org/wiki/Computer_science" ]
                publication_references : "12345"
                }}
```

Figure 3.2: Generic document example

3.2. Data upload and transformation

In order to run our queries, we need a data set. Among the available options, we decided to use "mockaroo.com" to generate a random data set. To do this, we inserted the structure of data we mentioned in previous sections, as follows:

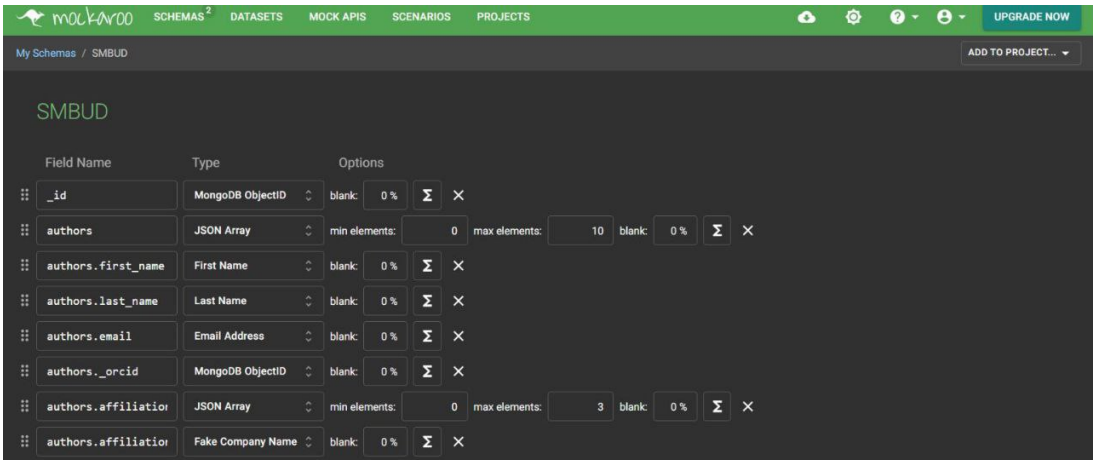


Figure 3.3: Generating data 1

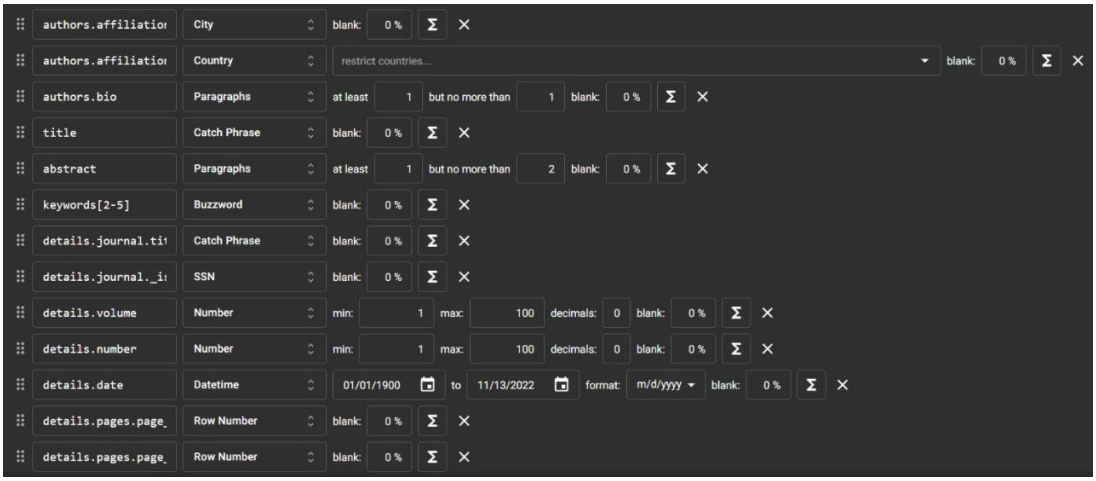


Figure 3.4: Generating data 2

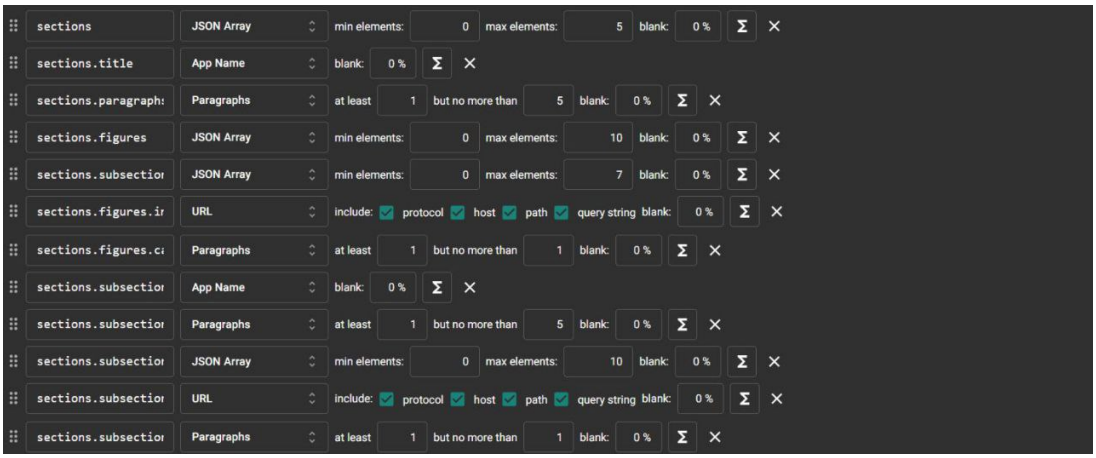


Figure 3.5: Generating data 3

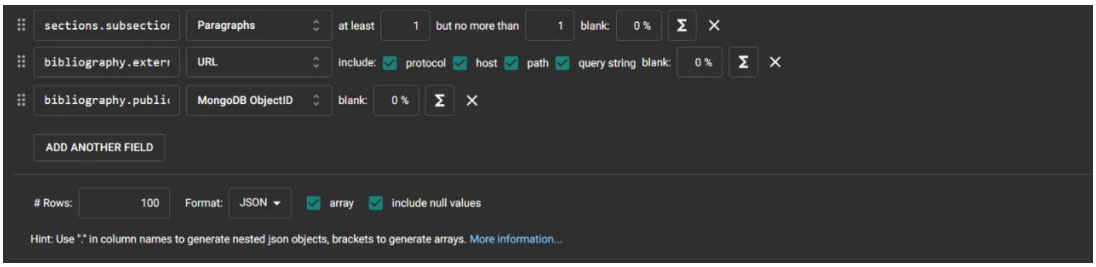


Figure 3.6: Generating data 4

- All unique identifiers namely "id" and "orcid" defined as "MongoDB object id".
- Some arrays namely "authors", "affiliations", "sections", "subsections" and "figures" are defined as "JSON array".
- Also the attributes namely "keywords", "publication references" and "external references" are arrays, but we defined them in a different way. Instead of being a JSON Array, we defined them in form of: attribute name[range].
- In order to make the "date" attribute compatible with the MongoDB format, we wrote some piece of code.
- Each object in the "publication references" array is assigned to a publication id randomly from the ones that were generated in the "_id" attribute (not to the ones generated from Mockaroo), in order to have consistent references.
- All other attributes are defined as shown in the pictures.

In addition, the generated authors, keywords and journals were also mixed up using a python script in order to have some authors and keywords that are present in more than one publication and more than one publication that contain the same journal.

3.3. Commands

1. **Insert a new publication.**
   The following command adds a new publication, with some relevant information about it, to the "publications" collection.

```
db.publications.insertOne({
    "title":  "x",
    "abstract": "y" ,
    "keywords": ["z" , "w"],
    "authors": [
        {"first_name" : "Carlo" , "last_name": "Sgaravatti"},
        {"first_name" : "Manu", "last_name": "Caputo"}
    ],
    "details": {
        "journal": {"title": "y", "_issn": "1234-5678"},
        "volume": 1,
        "issue_number": 1,
        "date": ISODate("2022-10-10")
    }
})
```

Figure 3.7: Data creation command 1

(a) (b)

Figure 3.8: Result of command 1

2. **Add external references to a publication.**
   With this command, we want to show how it is possible to add some external resources to a publication. First, we match the publication, given his title, and then we push into the external resources array all the URLs we want to insert.

```
db.publications.updateOne({ "title" : "x" }, {
    "$push" : {
        "bibliography.external_resources" : {
            "$each" : [ "https://abcd.com", "https://efgh.com"]
        }
}})
```

Figure 3.9: Data update command 2

```
db.publications.updateOne({ "title" : "x" }, {
    "$push" : {
        "bibliography.external_resources" : {
            "$each" : [ "https://abcd.com", "https://efgh.com"]
        }
}})
{ acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0 }
```

Figure 3.10: Result of command 2

3. **Remove an author from a publication.**
   Here we want to remove a given author from a publication, given its title. First, we match the publication that has the specified title and then we pull from the authors' array the author that has the specified first name and last name.

```
db.publications.updateOne({"title" : "x"}, {
    "$pull" : {
        "authors" : {
            "first_name" : "Manu",
            "last_name" : "Caputo"
        }
}})
```

Figure 3.11: Data update command 3

```
db.publications.updateOne({"title" : "x"}, {
    "$pull" : {
        "authors" : {
            "first_name" : "Manu",
            "last_name" : "Caputo"
        }
}})
{ acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0 }
```

Figure 3.12: Result of command 3

4. **Add two keywords to a publication, sort ascending, and trim the array to the last three keywords.**
   With this command we want to add two keywords to a publication, given its title, then we sort the keywords array in lexicographic order and we maintain only the last three elements of the array.

```
db.publications.updateOne({"title": "x"},{
    "$push":{
        "keywords":{
            "$each": ["o","r"],
            "$sort": 1,
            "$slice": -3
        }
    }
})
```

Figure 3.13: Data update command 4

```
db.publications.updateOne({"title": "x"},{
    "$push":{
        "keywords":{
            "$each": ["o","r"],
            "$sort": 1,
            "$slice": -3
        }
    }
})
{ acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0 }
```

Figure 3.14: Result of command 4

5. **Update the caption of a specific image, given his URL, in all documents.** In the following command, we want to update the caption of all the figures (in

all publications) that have a certain URL. First, we match the publications that contain an image with the specified URL. Then we change the caption of all the entries in the "sections.figures" arrays that have that image: this is done by means of the "arrayFilter" option, which permits to update only the elements in the array that have the specified property.

```
db.publications.updateMany(
    {"sections.figures.image_url": "http://dummyimage.com/175x100.png/5fa2dd/ffffff"},
    {"$set" : {"sections.$[].figures.$[el].caption" : "modified caption"}},
    {"arrayFilters": [{"el.image_url": "http://dummyimage.com/175x100.png/5fa2dd/ffffff"}]}
)
```

Figure 3.15: Data update command 5

```
db.publications.updateMany(
    {"sections.figures.image_url": "http://dummyimage.com/175x100.png/5fa2dd/ffffff"},
    {"$set" : {"sections.$[].figures.$[el].caption" : "modified caption"}},
    {"arrayFilters": [{"el.image_url": "http://dummyimage.com/175x100.png/5fa2dd/ffffff"}]}
)
{ acknowledged: true,
  insertedId: null,
  matchedCount: 11,
  modifiedCount: 11,
  upsertedCount: 0 }
```

Figure 3.16: Result of command 5

3.4. Queries

1. **Retrieve all the publications, that are published in a given period of time and that have a certain keyword, in descending order w.r.t the date and ascending order w.r.t. the title.**
   In the following query we want to find, given two dates x and y, all the publications that have been published in a date between x and y, and that contain a certain keyword. The result is sorted in descending order with respect to the date on which the publication is published; ties on the date are sorted in ascending order with respect to the title of the publication.

```
db.publications.find({
            "$and": [{"details.date": {"$gt": ISODate("1960-01-01")}},
                                {"details.date": {"$lt":ISODate("1980-01-01")}}],
            "keywords": {"$in" : ["artificial intelligence"]}
}).sort({"details.date": -1, "title": 1 })
```

Figure 3.17: Query 1

Figure 3.18: Result of query 1

2. **Retrieve the title and the abstract of all the publications that contain a given word in the abstract.**
   Here we want to return the title and the abstract of all the publications that contain a certain word in the abstract. This query runs thanks to the fact that a textual index on the abstract was created, as Figure 3.20 states.

```
db.publications.find( {
    "$text": {
        "$search": "Praesent blandit"
    }
}, {"title" : 1, "abstract" : 1})
```

Figure 3.19: Query 2

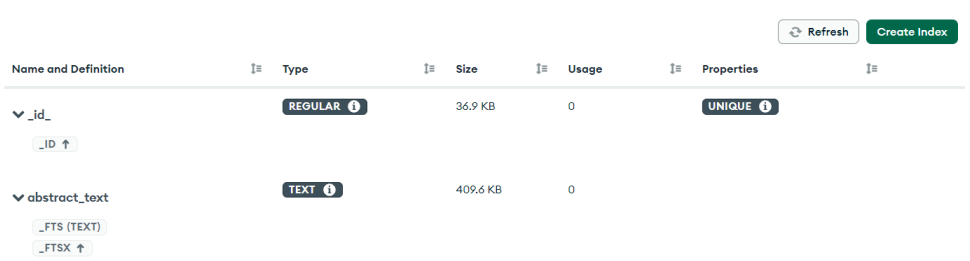


Figure 3.20: Indexes of the collection.

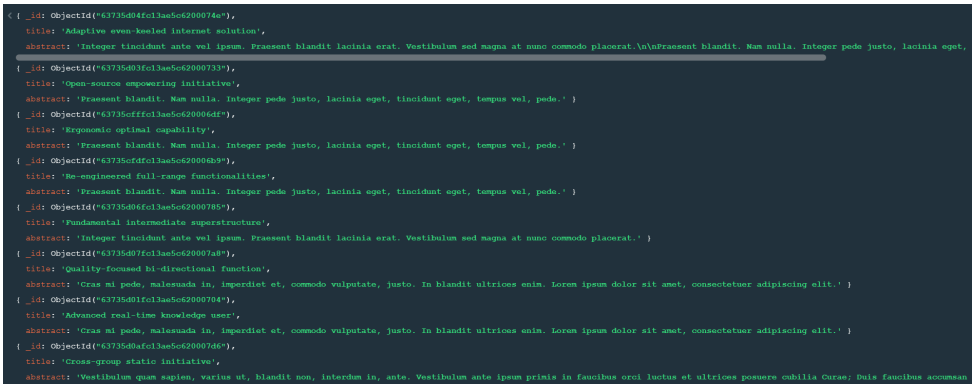


Figure 3.21: Result of query 2

3. **Retrieve the title of the publications which have more than 5 authors.**
   In this query, we want to find publications that have more than 5 authors. First, we calculate the number of authors that have contributed to the publication using the *$size* command on the authors' array (since the array can be not present, the *$ifNull* command creates an empty array in this cases) and we put it in a "count" attribute. Then, we keep only the publications that have "count" greater than or equal to 5.

```
db.publications.aggregate([{
        "$project" : {
            "title": 1,
            "count" : { "$size" : { "$ifNull": [ "$authors", [] ]}}
    }}, {
        "$match" : {
            "count" : {"$gte" : 5}
    }
}])
```

Figure 3.22: Query 3

```
{ _id: ObjectId("63735cfdfc13ae5c620006b9"),
  title: 'Re-engineered full-range functionalities',
  count: 6 }
{ _id: ObjectId("63735cfefc13ae5c620006c0"),
  title: 'Decentralized client-server intranet',
  count: 6 }
{ _id: ObjectId("63735cfffc13ae5c620006dc"),
  title: 'Proactive user-facing system engine',
  count: 5 }
{ _id: ObjectId("63735cfffc13ae5c620006e9"),
  title: 'Function-based client-server open system',
  count: 5 }
{ _id: ObjectId("63735d00fc13ae5c620006ee"),
  title: 'User-friendly attitude-oriented budgetary management',
  count: 5 }
{ _id: ObjectId("63735d00fc13ae5c620006f1"),
  title: 'Ergonomic client-driven hub',
  count: 7 }
{ _id: ObjectId("63735d00fc13ae5c620006f3"),
  title: 'Cloned dedicated pricing structure',
  count: 5 }
```

Figure 3.23: Result of query 3

4. **Retrieve the top 5 authors that have contributed to most publications.** Here we want to find the most active authors in the database, i.e. the ones that have contributed to the highest numbers of publications. To do this, we first unwind the authors' array, and then we group the results by the orcid of the author and we count the number of publications. The result is then sorted in descending order with respect to the count, and only the first 5 elements are returned using the *$limit* command.

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

Figure 3.24: Query 4

```
{ _id: ObjectId("63735d08fc13ae5c620007c1"), count: 8 }
{ _id: ObjectId("63735d05fc13ae5c62000756"), count: 7 }
{ _id: ObjectId("63735d05fc13ae5c6200076c"), count: 6 }
{ _id: ObjectId("63735cfffc13ae5c620006ec"), count: 6 }
{ _id: ObjectId("63735d0bfc13ae5c620007e2"), count: 5 }
```

Figure 3.25: Result of query 4

5. **Retrieve the title of all the publications that have a reference to a publication published in a certain journal.**
   In the following query, we want to find the publications that are referring to at least a publication that is published in a journal that has a certain title. First, we join the publications with their references by means of *$lookup*; all the joined references are inserted in an array named "referenced_publications". Then we filter the documents by keeping only the ones that have at least one reference that has the field "details.journal.title" equal to the title of the journal; this is done using the *$elemMatch* operator within the "referenced_publications" array. Finally, we keep only the title of the matched publication and some information about the referenced publications (title and journal title).

```
db.publications.aggregate([{
    "$lookup": {
        "from": "publications",
        "localField": "bibliography.publication_references",
        "foreignField" : "_id",
        "as" : "referenced_publication",
    }
}, {
    "$match": {
            "referenced_publication": {
                "$elemMatch": {"details.journal.title": "Greenlam"}
            }
    }
}, {
    "$project": {
        "title": 1,
        "referenced_publication.title": 1,
        "referenced_publication.details.journal": 1
    }
}])
```

Figure 3.26: Query 5

```
{ _id: ObjectId("63735d05fc13ae5c62000765"),
  title: 'Mandatory global benchmark',
  referenced_publication:
   [ { title: 'Ergonomic zero defect architecture',
       details: { journal: { title: 'Greenlam', _issn: '434-81-2133' } } },
     { title: 'Cross-platform logistical hardware',
       details: { journal: { title: 'Zathin', _issn: '224-71-4354' } } },
     { title: 'Multi-channelled optimal capability',
       details: { journal: { title: 'Regrant', _issn: '301-57-8573' } } } ] }
{ _id: ObjectId("63735d05fc13ae5c62000771"),
  title: 'Customizable zero tolerance throughput',
  referenced_publication:
   [ { title: 'Re-engineered full-range functionalities',
       details: { journal: { title: 'Asoka', _issn: '473-46-0544' } } },
     { title: 'Ergonomic zero defect architecture',
       details: { journal: { title: 'Greenlam', _issn: '434-81-2133' } } },
     { title: 'Ergonomic client-driven hub',
       details: { journal: { title: 'Lotstring', _issn: '756-11-1583' } } },
     { title: 'Advanced real-time knowledge user',
       details: { journal: { title: 'Cardify', _issn: '686-37-3706' } } },
     { title: 'Monitored regional model',
       details: { journal: { title: 'Span', _issn: '630-42-3947' } } },
     { title: 'Front-line grid-enabled Graphic Interface',
       details: { journal: { title: 'Domainer', _issn: '241-88-1691' } } },
     { title: 'Mandatory global benchmark',
       details: { journal: { title: 'Toughjoyfax', _issn: '148-85-3145' } } },
     { title: 'Customer-focused needs-based monitoring',
       details: { journal: { title: 'Cardguard', _issn: '679-98-8591' } } },
     { title: 'Integrated neutral framework',
       details: { journal: { title: 'Quo Lux', _issn: '116-12-4637' } } } ] }
```


Figure 3.27: Result of query 5

6. **Retrieve the number of images contained in all the sections and all the subsections of a specific document.**
   In this query, given the title of a publication, we want to find the total number of images (considering sections and subsections) that are present in the document. First, we match the document that has the specified title and we unwind the sections array (maintaining the index of the array element in order to group by it after). Then we calculate the number of figures in the section and we unwind all the subsections arrays: the calculation of the figures of the sections is done before the unwind in order to not repeat it many times. After, we calculate the number of figures per subsection. The result is first grouped by the index of the section (and by the id

of the document, since different publications can have the specified title) and then by the id of the publication: the first group is used to sum the number of figures in all the subsections of a specific section; the second is used to calculate the total number of figures in the document.

```
db.publications.aggregate([{
    "$match" : {"title" : "Re-engineered full-range functionalities" }
}, {
    "$unwind" : {
        "path" : "$sections",
        "includeArrayIndex" : "section_id"
    }
}, {
    "$project" : {
        "sections" : {
            "subsections": "$sections.subsections" ,
            "section_id": "$section_id" ,
            "sec_figures_num": {"$size" : {"$ifNull": [ "$sections.figures", [] ]}}
        }
    }
}, {
    "$unwind" : {"path" : "$sections.subsections"}
}, {
    "$project" : {
        "sec_figures_num" : "$sections.sec_figures_num",
        "section_id" : "$sections.section_id",
        "subsec_figures_num": {"$size" : {"$ifNull": [ "$sections.subsections.figures", [] ]}}
    }
}, {
    "$group" : {
        "_id": {"publ_id": "$_id", "sect_id": "$section_id"},
        "sec_figures_num": {"$first": "$sec_figures_num"},
        "sum_subsections": {"$sum" : "$subsec_figures_num"}
    }
}, {
    "$group" : {
        "_id" :  "$_id.publ_id",
        "total" : {"$sum" : {"$sum" : ["$sec_figures_num", "$sum_subsections"]}}
    }
}])
```

Figure 3.28: Query 6

```
{ _id: ObjectId("63735cfdfc13ae5c620006b9"), total: 14 }
```

Figure 3.29: Result of query 6

7. **Retrieve title, abstract and authors of the publications that a certain**

author has published starting from a certain date, and sort in descending order w.r.t. the title.

Here we want to find all the publications that have been published by a certain author, given his first name and last name, and after a certain date. The query first matches the documents with the specified properties, then it projects the result in order to keep only the title, the abstract and the name of the author, and finally, it sorts it with respect to the title in descending order.

```
db.publications.find({
    "details.date" : { "$gt" : ISODate("1980-01-01") },
    "authors": {
        "$elemMatch": {
            "first_name": "Kattie",
            "last_name": "Fredy"
        }
    }
},{"title" : 1, "abstract" : 1, "authors.first_name": 1, "authors.last_name": 1}).sort({"title" : -1 })
```

Figure 3.30: Query 7

```
{ _id: ObjectId("63735d0dfc13ae5c620007f6"),
  authors:
   [ { first_name: 'Berte', last_name: 'Ayars' },
     { first_name: 'Garfield', last_name: 'Aaronson' },
     { first_name: 'Marcellina', last_name: 'Ropkes' },
     { first_name: 'Joelle', last_name: 'Thouless' },
     { first_name: 'Kattie', last_name: 'Fredy' } ],
  title: 'Polarised local firmware',
  abstract: 'Praesent blandit. Nam nulla. Integer pede justo, lacinia eget, tincidunt eget, tempus vel, pede.\n\nMorbi port
{ _id: ObjectId("63735d01fc13ae5c62000702"),
  authors:
   [ { first_name: 'Amandy', last_name: 'Polo' },
     { first_name: 'Layney', last_name: 'Clements' },
     { first_name: 'Lorelle', last_name: 'Haddon' },
     { first_name: 'Catina', last_name: 'McOwen' },
     { first_name: 'Kattie', last_name: 'Fredy' },
     { first_name: 'Nickola', last_name: 'Klimkowski' } ],
  title: 'Function-based value-added concept',
  abstract: 'Duis consequat dui nec nisi volutpat eleifend. Donec ut dolor. Morbi vel lectus in quam fringilla rhoncus.' }
```

Figure 3.31: Query 7

8. **For each publication, get the number of authors that have already published something in the same journal before.**

   In the following query, for each publication x we want to find the number of authors that have already been published in a journal with the same title as the journal of x. This is done by means of the *$lookup* operator, using a pipeline inside it to simulate a sort of subquery. Inside the pipeline of the *$lookup* we unwind the authors' array and then, in the *$match* filter we find the publications that have the same title as the journal of x, that have been published before the date on which x was

published and such that the orcid of the considered author of the publication is in the authors' array of x (if a publication satisfies the first two properties and has two authors that have contributed to x this is counted twice since we want to count the authors, not the publications). At the end of the nested pipeline we group the results by the authors since if an author has already contributed to more than one publication with all the previous properties, this has to be counted just one time. Finally, the result of the pipeline is inserted into an array, that is projected to its size after the end of the *$lookup*. In the end, the result is sorted with respect to the title, in descending order.

```
db.publications.aggregate([{
    "$lookup": {
        "from": "publications",
        "let": {"journal_title": "$details.journal.title", "date": "$details.date", "orcid": "$authors._orcid"},
        "pipeline": [{
            "$unwind": {"path": "$authors"}
        }, {
            "$match": {
                "$expr": {
                    "$and": [
                        {"$eq": ["$details.journal.title", "$$journal_title"]},
                        {"$lt": ["$details.date", "$$date"]},
                        {"$in": ["$authors._orcid", "$$orcid"]}
                    ]
                }
            }
        }, {"$group": {"_id": "$authors._orcid"}}],
        "as" : "result"
    }
}, {
    "$project": {
        "title": 1,
        "details.journal.title": 1,
        "authors._orcid": 1,
        "authors.first_name": 1,
        "authors.last_name": 1,
        "n_authors": {"$size": "$result"}
    }
}, {
    "$sort" : { "title": -1}
}])
```

Figure 3.32: Query 8

```
{ _id: ObjectId("63735d0cfc13ae5c620007ef"),
  authors:
   [ { first_name: 'Rennie',
       last_name: 'Bezemer',
       _orcid: ObjectId("63735cfefc13ae5c620006c3") },
     { first_name: 'Reginald',
       last_name: 'Slyde',
       _orcid: ObjectId("63735cfefc13ae5c620006cf") } ],
  title: 'Digitized needs-based instruction set',
  details: { journal: { title: 'Tres-Zap' } },
  n_authors: 1 }
{ _id: ObjectId("63735d00fc13ae5c620006f3"),
  authors:
   [ { first_name: 'Stephie',
       last_name: 'Fishpond',
       _orcid: ObjectId("63735cfffc13ae5c620006e6") },
```


Figure 3.33: Result of query 8

9. **Retrieve the title of the publications with the maximum number of pages per journal.**
   In this query, we want to know, for each journal, the maximum number of pages that a publication published in it has. This is done with a pipeline of operations. First, we calculate the number of pages that a publication has. Then we sort the result with respect to the number of pages. Finally, we group the result by the ISSN of the journal and we keep track of the first document found with that ISSN in order to select the one with the maximum number of pages (since the documents are ordered by the number of pages, the first one is the maximum). We used this approach, instead of calculating the maximum with the *$max* operator because we wanted to find the publication that has the maximum number of pages (not just the actual maximum number of pages).

```
db.publications.aggregate([{
    "$project": {
        "_id": "$_id",
        "title": "$title",
        "journal": "$details.journal",
        "pages_count": {"$subtract": ["$details.pages.end_page",
                                      "$details.pages.start_page"]}
    }
}, {"$sort": {"pages_count": -1}}, {
    "$group": {
        "_id": "$journal._issn",
        "journal_title": {"$first": "$journal.title"},
        "max_pages": {"$first": "$pages_count"},
        "publication": {"$first": "$title"}
    }
}])
```

Figure 3.34: Query 9

```
{ _id: '630-42-3947',
  journal_title: 'Span',
  max_pages: 28,
  publication: 'Monitored regional model' }
{ _id: '681-28-7093',
  journal_title: 'Tin',
  max_pages: 16,
  publication: 'Ameliorated maximized database' }
{ _id: '655-04-2654',
  journal_title: 'Daltfresh',
  max_pages: 16,
  publication: 'Horizontal zero tolerance internet solution' }
{ _id: '728-08-4453',
  journal_title: 'Opela',
  max_pages: 16,
  publication: 'Open-architected empowering flexibility' }
{ _id: '125-66-2988',
  journal_title: 'Bigtax',
  max_pages: 21,
  publication: 'De-engineered dedicated archive' }
```

Figure 3.35: Result of query 9

10. **Retrieve the 10 most popular keywords in descending order.**
    Here we want to find the keywords that are present in the highest number of publications. To do so, the keywords array must be unwinded; then, for each keyword, we count the occurrences of it in the publications. Finally, we sort the result with respect to the count in descending order and we limit the result to the top 10 keywords.

```
db.publications.aggregate([{
    "$unwind": {
        "path": "$keywords"
    }
}, {
    "$group": {
        "_id": "$keywords",
        "count": {"$sum" : 1}
    }
}, {"$sort": {"count": -1 }}, {"$limit": 10}])
```

Figure 3.36: Query 10

```
{ _id: 'global', count: 15 }
{ _id: 'optimizing', count: 15 }
{ _id: 'capability', count: 14 }
{ _id: 'monitoring', count: 13 }
{ _id: 'database', count: 13 }
{ _id: 'background', count: 13 }
{ _id: 'info-mediaries', count: 13 }
{ _id: 'discrete', count: 13 }
{ _id: 'disintermediate', count: 13 }
{ _id: 'Versatile', count: 12 }
```

Figure 3.37: Result of query 10

4 | Spark

4.1. Description of the structure of the dataframes

The structure we have used is composed of a main dataframe called 'publications_spark' which contains the attributes '_id', 'title', 'year', 'n_citation', 'page_start', 'page_start', 'page_end', 'lang', 'volume', 'issue', 'ISBN', 'doi', 'pdf', 'abstract', 'publisher', 'venue_id' for a specific publication. In particular, 'venue_id' is the reference key to the 'venues_spark' dataframe. It also contains an array attribute 'references' which contains indexes pointing to the referenced publications and other 4 array attributes containing indexes which refer to other dataframes:

- attribute 'authors' points to the 'authors_spark'
- attribute 'keywords' points to the 'keywords_spark'
- attribute 'urls' points to the 'urls_spark'
- attribute 'fos' points to the 'fos_spark'

The dataframe 'authors_spark' contains '_id', 'firstname' and 'lastname' for each author.
Inside the dataframe 'venues_spark' there are attributes '_id', 'name_d' (the name of the venue), 'type', 'raw' (the abbreviation) and 'publisher' for each venue.
The dataframe 'keywords_spark' contains 'key_name' and 'key_index'.
The dataframe 'fos_spark' contains 'fos_name' and 'fos_index'.
The dataframe 'urls_spark' contains 'url_name' and 'url_index'.

4.1.1. Motivation of our structure

We decide to store in the main dataframe 'publications_spark' arrays of indexes of the tuples of the others dataframes. In particular, for 'keywords_spark' and 'fos_spark', it's better to perform a search over an index instead of parsing a string. So, for instance, when we perform queries on a keyword we first are able to retrieve the 'key_index', and later we can just search for the index of the array without parsing 'key_name' (since many

publications can have common keywords). This allows us to have better performances in the search.

```
root
 |-- _id: string (nullable = false)
 |-- title: string (nullable = false)
 |-- year: integer (nullable = false)
 |-- n_citation: integer (nullable = true)
 |-- page_start: string (nullable = true)
 |-- page_end: string (nullable = true)
 |-- lang: string (nullable = true)
 |-- volume: string (nullable = true)
 |-- issue: string (nullable = true)
 |-- isbn: string (nullable = true)
 |-- doi: string (nullable = false)
 |-- pdf: string (nullable = true)
 |-- abstract: string (nullable = true)
 |-- publisher: string (nullable = true)
 |-- venue_id: string (nullable = true)
 |-- keywords: array (nullable = true)
 |    |-- element: long (containsNull = true)
 |-- fos: array (nullable = true)
 |    |-- element: long (containsNull = true)
 |-- url: array (nullable = true)
 |    |-- element: long (containsNull = true)
 |-- authors: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- references: array (nullable = true)
 |    |-- element: string (containsNull = true)
```

Figure 4.1: The structure of the 'publications_spark' dataframe

```
root
 |-- _id: string (nullable = false)
 |-- name_d: string (nullable = false)
 |-- type: string (nullable = false)
 |-- raw: string (nullable = true)
 |-- publisher: string (nullable = true)
```

Figure 4.2: The structure of the 'venues_spark' dataframe

```
root
 |-- _id: string (nullable = false)
 |-- firstname: string (nullable = false)
 |-- lastname: string (nullable = false)
```

Figure 4.3: The structure of the 'authors_spark' dataframe

```
root
 |-- url_name: string (nullable = false)
 |-- url_index: long (nullable = false)
```

Figure 4.4: The structure of the 'urls_spark' dataframe

```
root
 |-- key_name: string (nullable = false)
 |-- key_index: long (nullable = false)
```

Figure 4.5: The structure of the 'keywords_spark' dataframe

```
root
 |-- fos_name: string (nullable = false)
 |-- fos_index: long (nullable = false)
```

Figure 4.6: The structure of the 'fos_spark' dataframe

4.2. Import

The dataset is the same one we used for Neo4j. In order to import it we have used a short script (better described in the Import.ipynb file) exploiting the pandas library in Python.

4.3. Commands

1. **Update the publications that were published in 1993**
   This command will change the published year of all the publications which are published in 1993. And the other publications will remain the same as before. First, we select the "year" column of publications by "withColumn" command. Then we check if the "year" is equal to "1993", we change it to "1994". Otherwise, we return the original "year". Finally, we show the results.

```
publications_spark = publications_spark.withColumn('year',
    when(col('year') == 1993, 1994)
    .otherwise(col('year'))
)

publications_spark.show(truncate = False)
```

Figure 4.7: Command 1

```
+------------------------+-----------------------------------------------------------------------------------------------+----+
|_id                     |title                                                                                          |year|
+------------------------+-----------------------------------------------------------------------------------------------+----+
|53e99784b7602d9701f3e151|A solution to the problem of touching and broken characters.                                   |1994|
|53e99784b7602d9701f3e15d|Timing yield estimation using statistical static timing analysis                               |2005|
|53e99784b7602d9701f3f411|Using XML to Integrate Existing Software Systems into the Web                                  |2002|
|53e99784b7602d9701f3f5fe|Research on resource allocation for multi-tier web applications in a virtualization environment|2011|
|53e99784b7602d9701f3f95d|FCLOS                                                                                          |2009|
|53e99785b7602d9701f4061f|GCP                                                                                            |2002|
|53e99785b7602d9701f442cb|Bhoomi                                                                                         |2009|
|53e9978ab7602d9701f4803e|A                                                                                              |2008|
|53e9978ab7602d9701f47e18|APEX                                                                                           |2001|
|53e9978ab7602d9701f4a085|N                                                                                              |2004|
|53e9978ab7602d9701f4b0e1|Overview                                                                                       |1980|
|53e9978ab7602d9701f4b2cc|Overview                                                                                       |1981|
|53e9978ab7602d9701f4bc6d|Laps                                                                                           |2006|
|53e9978ab7602d9701f4bc78|L                                                                                              |2005|
|53e9978ab7602d9701f4bf0c|L                                                                                              |2007|
|53e9978db7602d9701f4ccea|Mindful                                                                                        |2008|
|53e9978db7602d9701f4d1aa|MESHMdl                                                                                        |2007|
|53e9978db7602d9701f4eb65|k                                                                                              |2007|
|53e9978db7602d9701f4f829|H                                                                                              |2009|
|53e9978db7602d9701f4f82b|HOLCF=HOL+LCF                                                                                  |1999|
+------------------------+-----------------------------------------------------------------------------------------------+----+
only showing top 20 rows
```

Figure 4.8: Result of command 1

2. **Add a reference to a publication.**
   Given two ids of two publications, we want to add the first id to the references of the publications that have the second id. To do so, we use the "array_union" function

by passing the previous array and an array with only one element, i.e. the id to be added, in it. Finally, we show the result.

```
publications_spark = publications_spark.withColumn("references",
      when(col('_id') == '53e99784b7602d9701f3e151', array_union("references",array(lit('53e9978ab7602d9701f4b2cc'))))
      .otherwise(col("references"))
)

publications_spark.select(col('_id'), col('references')).show(truncate = False)
```

Figure 4.9: Command 2

```
+------------------------+--------------------------------------------------------------------------------------------------------------------------------------------------------------------+
|_id                     |references
+------------------------+--------------------------------------------------------------------------------------------------------------------------------------------------------------------+
|53e99784b7602d9701f3e151|[53e99cf5b7602d97025ace63, 557e8a7a6fee0fe990caa63d, 53e9a96cb7602d97032c459a, 53e9b929b7602d9704515791, 557e59ebf6678c77ea222447, 53e9978ab7602d9701f4b2cc]
|53e99784b7602d9701f3e15d|[53e9a8a9b7602d97031f6bb9, 599c7b6b601a182cd27360da, 53e9b443b7602d9703f3e52b, 53e9a6a6b7602d9702fdc57e, 599c7b6a601a182cd2735703, 53e9aad9b7602d970345afea,
|53e99784b7602d9701f3f411|[53e9adbdb7602d97037be8a2, 53e9bb53b7602d9704792f33, 558aa425e4b0b32fcb37fff4, 558abd44e4b031bae1f9653a, 53e9a326b7602d9702c32229, 53e9b1d7b7602d9703c6ce7c,
|53e99784b7602d9701f3f5fe|[53e9a073b7602d9702957efa, 53e9ad87b7602d970377bfb5, 53e9be51b7602d9704b11381, 53e9be04b7602d9704abb31d, 53e9992bb7602d9702169236, 53e998cdb7602d97021044db,
|53e99784b7602d9701f3f95d|[53e99ee0b7602d97027ae130, 53e9aca7b7602d9703681f1f, 53e9b9c0b7602d97045b36b4, 53e9b856b7602d970441929a, 53e9af75b7602d97039bbc74, 53e9bb23b7602d970475dcba,
|53e99785b7602d9701f4061f|[53e9adf6b7602d97037fedf7, 53e9bc15b7602d970487aa8e, 53e9a501b7602d9702e23597, 53e9ad72b7602d970375dcd4, 53e9b791b7602d970433ba33, 53e9be1ab7602d9704ad488b]
|53e99785b7602d9701f442cb|[53e9b2ffb7602d9703dc17e9, 53e9bca5b7602d9704923fd7]
|53e9978ab7602d9701f4803e|[53e9b775b7602d970431d99d, 558a2eefe4b0b32fcb354e82, 53e9a839b7602d9703186586, 53e9a102b7602d97029ed9fb, 53e9bddbb7602d9704a8c90e, 53e9a562b7602d9702e891b6]
|53e9978ab7602d9701f47e18|[53e9bc05b7602d97048667a6, 53e9af12b7602d9703948397, 557f027dd19faf961d16e1e4, 53e9bac1b7602d97046ed383, 53e99a92b7602d97023097d4, 53e9a5cdb7602d9702ef73ce,
|53e9978ab7602d9701f4a085|[573696816e3b12023e58cdca, 53e9aacab7602d9703446c42]
|53e9978ab7602d9701f4b0e1|[53e9986eb7602d97020a699b]
|53e9978ab7602d9701f4b2cc|[53e9aeb2b7602d97038d8f63, 53e9af12b7602d970394cc70, 53e9b4f5b7602d9704021c19]
|53e9978ab7602d9701f4bc6d|[573695936e3b12023e4b3817, 53e9af75b7602d97039baeef, 53e9b851b7602d9704417600, 53e9b275b7602d9703d18d70, 53e9aa56b7602d97033c3ed0, 53e9ac8eb7602d9703664007,
|53e9978ab7602d9701f4bc78|[53e9af2db7602d97039674d3, 53e99c20b7602d97024cedba, 558a848fe4b037c087579df1, 53e9a034b7602d9702917a55, 53e9a965b7602d97032be79a]
|53e9978ab7602d9701f4bf0c|[5736961e6e3b12023e52faa0, 53e9af99b7602d97039e14d1]
|53e9978db7602d9701f4ccea|[53e9b0c2b7602d9703b38520, 53e9a6b5b7602d9702feb55b, 53e9bc36b7602d97048a9356, 5736961f6e3b12023e530656, 558a340584ae84d265bc3481, 53e9a487b7602d9702da622b,
|53e9978db7602d9701f4d1aa|[53e99a49b7602d97022ac8f9, 53e99930b7602d970216b829, 53e9b903b7602d97044e6d76, 53e9b3c7b7602d9703eb257c, 53e9a97bb7602d97032d31f3, 53e9b091b7602d9703afcf63,
|53e9978db7602d9701f4eb65|[53e99d8fb7602d970264d332, 53e9b4f5b7602d970402082e, 558a89ffe4b0b32fcb376975, 53e9a877b7602d97031c1420, 53e9a63eb7602d9702f6e093, 53e9a8b0b7602d9703200071,
|53e9978db7602d9701f4f829|[53e9bb93b7602d97047d8d37, 53e9a49db7602d9702dbea08, 53e9a408b7602d9702d1f613, 53e99bd4b7602d970247acd4, 53e99938b7602d97021764e9, 53e9b976b7602d97045650e1,
|53e9978db7602d9701f4f82b|[53e99a8cb7602d9702300de4, 5c77db0c4895d9cbc65bbd02, 53e9a965b7602d97032bfcb5, 557dbfe4f6678c77ea21d168, 53e9b102b7602d9703b7b83d, 53e9a026b7602d970290e162,
+------------------------+--------------------------------------------------------------------------------------------------------------------------------------------------------------------+
only showing top 20 rows
```

Figure 4.10: Result of command 2

3. **Remove a certain URL from all the publications and from the URLs dataframe.**
In this command, given a certain URL, we want to remove it both from the URLs dataframe and from all the nested arrays of URLs in the publications dataframe. First, we find the index of the URL. Then we remove that index from the arrays in the publications dataframe by means of the "array_remove" function. Finally, we remove the entire row from the URLs dataframe and we show the results by comparing the previous dataframes with the new ones.

```
# previous dataframes
publications_spark.select(col('_id'), col('url')).limit(5).show(truncate = False)
urls_spark.limit(5).show(truncate = False)

url_id = urls_spark.filter(col("url_name")=="http://dx.doi.org/10.1109/ICDAR.1993.395663") \
                .select("url_index").collect()[0][0]

df1 = publications_spark.withColumn("url",
        when(array_contains(publications_spark.url, url_id), array_remove(publications_spark.url,url_id))
        .otherwise(col("url"))
)

df2 = urls_spark.filter(col("url_index") != url_id)

df1.select(col('_id'), col('url')).limit(5).show(truncate = False)
df2.limit(5).show(truncate = False)
```

Figure 4.11: Command 3

```
+------------------------+----------+
|_id                     |url       |
+------------------------+----------+
|53e99784b7602d9701f3e151|[0]       |
|53e99784b7602d9701f3e15d|[1, 2]    |
|53e99784b7602d9701f3f411|[3, 4, 5] |
|53e99784b7602d9701f3f5fe|[6, 7, 8] |
|53e99784b7602d9701f3f95d|[9, 10, 8]|
+------------------------+----------+

+-------------------------------------------------------------------------+---------+
|url_name                                                                 |url_index|
+-------------------------------------------------------------------------+---------+
|http://dx.doi.org/10.1109/ICDAR.1993.395663                              |0        |
|http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?tp=&arnumber=1465124  |1        |
|http://dx.doi.org/10.1109/ISCAS.2005.1465124                             |2        |
|http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?tp=&arnumber=1044548  |3        |
|http://dx.doi.org/10.1109/CMPSAC.2002.1044548                            |4        |
+-------------------------------------------------------------------------+---------+

+------------------------+----------+
|_id                     |url       |
+------------------------+----------+
|53e99784b7602d9701f3e151|[]        |
|53e99784b7602d9701f3e15d|[1, 2]    |
|53e99784b7602d9701f3f411|[3, 4, 5] |
|53e99784b7602d9701f3f5fe|[6, 7, 8] |
|53e99784b7602d9701f3f95d|[9, 10, 8]|
+------------------------+----------+

+-------------------------------------------------------------------------+---------+
|url_name                                                                 |url_index|
+-------------------------------------------------------------------------+---------+
|http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?tp=&arnumber=1465124  |1        |
|http://dx.doi.org/10.1109/ISCAS.2005.1465124                             |2        |
|http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?tp=&arnumber=1044548  |3        |
|http://dx.doi.org/10.1109/CMPSAC.2002.1044548                            |4        |
|http://doi.ieeecomputersociety.org/10.1109/CMPSAC.2002.1044548           |5        |
+-------------------------------------------------------------------------+---------+
```

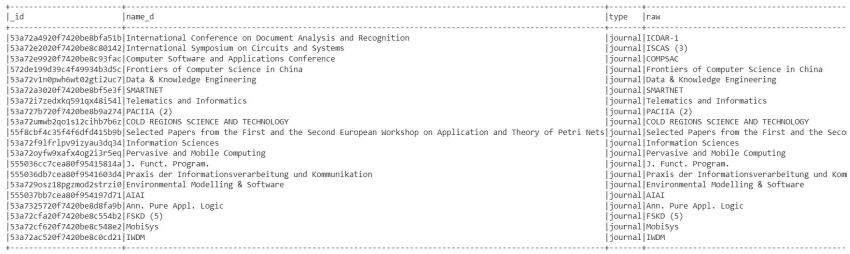
Figure 4.12: Result of command 3

4. **Update the type of venues from conference to journal**
   First of all, we should find the types of venues and then change them to journals if they are conferences. So, we have to find them in the venue data frame. In the second line, if the type of venue is a conference, it will change to the journal. Otherwise, we do not change the type of venue. Here are the results:

```
df = venue_spark.withColumn("type",
                            when(col('type') ==  'conference' , 'journal')
                            .otherwise(col("type")))

df.show(truncate = False)
```

Figure 4.13: Command 4



_id	name_d	type	raw
53a72a4920f7420be8bfa51b	International Conference on Document Analysis and Recognition	journal	ICDAR-1
53a72e2020f7420be8c80142	International Symposium on Circuits and Systems	journal	ISCAS (3)
53a72e9920f7420be8c93fac	Computer Software and Applications Conference	journal	COMPSAC
572de199d39c4f49934b3d5c	Frontiers of Computer Science in China	journal	Frontiers of Computer Science in China
53a72v1n0pwh6wt02gti2uc7	Data & Knowledge Engineering	journal	Data & Knowledge Engineering
53a72a3020f7420be8bf5a2f	SMARTNET	journal	SMARTNET
53a72i7zedxkq591qx48i54l	Telematics and Informatics	journal	Telematics and Informatics
53a727b720f7420be8b9a274	PACIIA (2)	journal	PACIIA (2)
53a72umwb2qo1s12cihb7b6z	COLD REGIONS SCIENCE AND TECHNOLOGY	journal	COLD REGIONS SCIENCE AND TECHNOLOGY
55f8cbf4c35f4f6df6d15b9b	Selected Papers from the First and the Second European Workshop on Application and Theory of Petri Nets	journal	Selected Papers from the First and the Seco
53a72f9lfrlpv9izyau3dg34	Information Sciences	journal	Information Sciences
53a72oyfw9xafx4og213r5eq	Pervasive and Mobile Computing	journal	Pervasive and Mobile Computing
555036cc7cea80f95415814a	J. Funct. Program.	journal	J. Funct. Program.
555036db7cea80f954160304	Praxis der Informationsverarbeitung und Kommunikation	journal	Praxis der Informationsverarbeitung und Kom
53a729osz18pgzmod2xtrz10	Environmental Modelling & Software	journal	Environmental Modelling & Software
555037bb7cea80f95419 7d71	AIAI	journal	AIAI
53a7325720f7420be8d8fa9b	Ann. Pure Appl. Logic	journal	Ann. Pure Appl. Logic
53a72cfa20f7420be8c554b2	FSKD (5)	journal	FSKD (5)
53a72cf620f7420be8c548e2	MobiSys	journal	MobiSys
53a72ac520f7420be8c0cd21	IWDM	journal	IWDM

Figure 4.14: Result of command 4

5. **Insert a new venue in the exiting data frame.**
   The purpose of this command was to add a new row in the venues data frame which had been created beforehand during the import of the data. The idea was to create a new data frame containing the first one and the new row created manually. To do this, a "PySpark union()" type transformation containing the initial data frame as an argument was applied to the new row. Here are the results obtained:

```
# Insert Venues data in a dataFrame
venues_schema = StructType([ \
    StructField("_id", StringType(), False), \
    StructField("name_d", StringType(), False), \
    StructField("type", StringType(), False), \
    StructField("raw", StringType(), True), \
    StructField("publisher", StringType(), True) \
])

# Show Venues data
venue_spark = spark.createDataFrame(data=venue, schema=venues_schema)
venue_spark.show()

# Add a new row to the existing rows and shows data
new_venue = spark.createDataFrame([("1234","new_venue","journal","top_v","team MC LAF")], venues_schema)
appended_venue = new_venue.union(venue_spark)
appended_venue.show()
```

Figure 4.15: Command 5

_id	name_d	type	raw	publisher
1234	new_venue	journal	top_v	team MC LAF
53a72a4920f7420be...	International Con...	conference	ICDAR-1	Unknown
53a72e2020f7420be...	International Sym...	conference	ISCAS (3)	Unknown
53a72e9920f7420be...	Computer Software...	conference	COMPSAC	Unknown
572de199d39c4f499...	Frontiers of Comp...	conference	Frontiers of Comp...	Unknown
53a72v1n0pwh6wt02...	Data & Knowledge ...	journal	Data & Knowledge ...	North-Holland
53a72a3020f7420be...	SMARTNET	journal	SMARTNET	Unknown
53a72i7zedxkq591q...	Telematics and In...	journal	Telematics and In...	Pergamon
53a727b720f7420be...	PACIIA (2)	journal	PACIIA (2)	Unknown
53a72umwb2qo1s12c...	COLD REGIONS SCIE...	journal	COLD REGIONS SCIE...	Unknown
55f8cbf4c35f4f6df...	Selected Papers f...	conference	Selected Papers f...	null
53a72f9lfrlpv9izy...	Information Sciences	journal	Information Sciences	Elsevier
53a72oyfw9xafx4og...	Pervasive and Mob...	journal	Pervasive and Mob...	Elsevier
555036cc7cea80f95...	J. Funct. Program.	conference	J. Funct. Program.	null
555036db7cea80f95...	Praxis der Inform...	conference	Praxis der Inform...	null
53a729osz18pgzmod...	Environmental Mod...	journal	Environmental Mod...	Elsevier
555037bb7cea80f95...	AIAI	conference	AIAI	null
53a7325720f7420be...	Ann. Pure Appl. L...	conference	Ann. Pure Appl. L...	null
53a72cfa20f7420be...	FSKD (5)	conference	FSKD (5)	Unknown
53a72cf620f7420be...	MobiSys	conference	MobiSys	Unknown

Figure 4.16: Result of command 5

4.4. Queries

1. **Retrieve title of 10 English publications which have at least one keyword containing "method" inside**

   First of all, we filter the publications to retrieve only the English ones. Then we select our two desired columns named “title” and “keywords”. In this step, we explode the keywords array and rename the result to keywords again. If we show the result here, we can see that we have a table with columns “title” and “keywords index”. But we need the names of the keywords. So we join this result with the keywords data frame. After all, we filter the result to get only the ones containing “method”. Finally, we show only 10 results.

```
publications_spark.filter(col('lang') == 'en') \
        .select(col('title'), explode(col('keywords'))) \
        .withColumnRenamed('col', 'keywords') \
        .join(keywords_spark, col('keywords') == keywords_spark.key_index,'inner') \
        .filter(col("key_name").like("%method%")) \
        .select(col('title'), col('key_name')) \
        .limit(10) \
        .show(truncate = False)
```

Figure 4.17: Query 1

```
+-----------------------------------------------------------------------------------------+-----------------------------+
|title                                                                                    |key_name                     |
+-----------------------------------------------------------------------------------------+-----------------------------+
|Quality-aware collaborative question answering: methods and evaluation                   |design method                |
|Filtering for uncertain 2-D discrete systems with state delays                           |design method                |
|Timing yield estimation using statistical static timing analysis                         |design method                |
|Scalable timers for soft state protocols                                                 |receiver estimation methods  |
|New approach to mixed H2/H&infin; filtering for polytopic discrete-time systems          |developed filter design method|
|Analysis of brain electrical topography by spatio-temporal wavelet decomposition         |boundary element method      |
|A Gait Generation for an Unlocked Joint Failure of the Quadruped Robot with Balance Weight|control method              |
|A game-theoretic approach to energy-efficient power control in multicarrier CDMA systems |iterative methods            |
|Distributed multiuser power control for digital subscriber lines                         |iterative methods            |
|Computation and Refinement of Statistical Bounds on Circuit Delay                        |iterative methods            |
+-----------------------------------------------------------------------------------------+-----------------------------+
```

Figure 4.18: Result of query 1

2. **Return the top 5 years with the highest number of publications in the field of "computer science" in descending order.**

   First, we select our desired columns named “id”, “title”, “year” and “field of study” of the publications. Here, if we show results, we can see that we have a table with the column “field of study index”. But we need names of fields of study. So we join this result with the fields of study data frame. Then, we filter the result to get only the publications in the “computer science” field of study. After that, the result is grouped by the published year and we count the number of publications for each year. Finally, we sort the result by year in descending order and show the top 5 of them.

```
retrieved_df = publications_spark.select(col('_id'), col('title'), col('year'), explode(col('fos'))) \
                    .withColumnRenamed('col', 'fields of study') \
                    .join(fos_spark, col('fields of study') == fos_spark.fos_index, 'inner') \
                    .select(col('_id'), col('title'), col('year'), col('fos_name'))

retrieved_df.filter(col('fos_name') == 'computer science') \
            .groupBy(col('year')) \
            .agg(count('_id').alias('number of publications')) \
            .sort(col('number of publications').desc()) \
            .limit(5) \
            .show(truncate = False)
```

Figure 4.19: Query 2

```
+----+----------------------+
|year|number of publications|
+----+----------------------+
|2004|51                    |
|2006|44                    |
|2002|44                    |
|1998|43                    |
|2001|42                    |
+----+----------------------+
```

Figure 4.20: Result of query 2

3. **Retrieve the 3 oldest publications published in a venue published by the 'North-Holland' agency.**
   First, we want to find all the venues that are published by the "North-Holland" agency. Then we join the filtered venue dataframe with the publications dataframe. Since we are doing an inner join, all the tuples that we retrieve are all publications that have been published inside a venue published by the "North-Holland" agency. Finally, we sort the retrieved dataframe with respect to the year in ascending order and we select only the top 3 rows. For visualization purposes, only a meaningful subset of attributes is selected.

```
venue_df = venue_spark.filter(col('publisher') == 'North-Holland') \
                    .withColumnRenamed('publisher', 'venue_publisher') \
                    .withColumnRenamed('_id', 'id_venue')

retrieved_df = publications_spark.join(venue_df, publications_spark.venue_id == venue_df.id_venue, 'inner') \
                    .sort(col('year').asc()).limit(3) \
                    .select(col("_id"), col("title"), col("year"), col("id_venue"), col("venue_publisher"), col("name_d"))

retrieved_df.show(truncate=False)
```

Figure 4.21: Query 3

```
+------------------------+-----------------------------------------------------------------+----+------------------------+---------------+-----------------------------------+
|_id                     |title                                                            |year|id_venue                |venue_publisher|name_d                             |
+------------------------+-----------------------------------------------------------------+----+------------------------+---------------+-----------------------------------+
|53e9af99b7602d97039e14d1|Fuzzy proximities structures and fuzzy grills                    |1996|539078f320f770854f5a88f2|North-Holland  |Fuzzy Sets and Systems             |
|53e9a15bb7602d9702a4f6fc|Constraint satisfaction problems: Algorithms and applications    |1999|548259f6582fc50b5e138119|North-Holland  |European Journal of Operational Research|
|53e9aacab7602d9703446c42|Analysis of third-party warehousing contracts with commitments   |2001|548259f6582fc50b5e138119|North-Holland  |European Journal of Operational Research|
+------------------------+-----------------------------------------------------------------+----+------------------------+---------------+-----------------------------------+
```

Figure 4.22: Result of query 3

4. **Retrieve the authors that have contributed to more than 2 publications after the year 2000.**

   First, we want to select the publications that have been published after the year 2000. Then we explode the authors' array, which contains the orcid of the authors, and we group be the orcid. For each group, we count the number of publications and we keep only the tuples that have more than 2 in the field produced by the count operation. Finally, we perform an inner join with the authors' dataframe, in order to retrieve the firstname and the lastname of the authors.

```
df = publications_spark.filter(col('year') > 2000) \
        .select(col('_id'), explode(col('authors'))) \
        .withColumnRenamed('col', '_orcid') \
        .groupBy(col('_orcid')) \
        .agg(count('_id').alias('num publications')) \
        .filter(col('num publications') > 2) \
        .join(authors_spark, col('_orcid') == authors_spark._id) \
        .select(col('_orcid'), col('num publications'), col('firstname'), col('lastname'))

df.show(truncate=False)
```

Figure 4.23: Query 4

_orcid	num publications	firstname	lastname
53f47b80dabfae8a6845cd1d	3	Jian	Pei
56127a6545ce1e5962c3c93a	4	Dana	Nau
548effffdabfaef989f0972d	3	Fabio	Marton
540837d5dabfae8faa6332a2	5	Markus	Gross
53f434b3dabfaeb22f461d7b	3	Aseem	Agarwal
53f43b64dabfaefedbaf97e4	5	Ilias	Michalarias
5433ca63dabfaebba5827ae0	6	Rolf	Niedermeier
53f810d1dabfae8faa4d7ef8	5	Jean-Luc	Marichal
53f3a90edabfae4b34aeb7d2	3	Samson	Lasaulce
53f63533dabfaeeec51b3a95	3	Shaosheng	Zhou
5405aefadabfae92b41f3a30	7	Huijun	Gao
542d255edabfae478c1a0b61	3	yongzhi	cao
53f469b2dabfaee02ada1fc0	4	Qing-Long	Han
53f7899edabfae8faa496187	3	HweeHwa	Pang
53f434fcdabfaee0d9b58d1e	8	FRED	F. FERRI
54406fc3dabfae805a6bddf3	3	Michael	Wand
53f31b07dabfae9a84434a82	3	Pavel	B. Brazdil
53f43886dabfaedd74db44bd	3	Leif	P. Kobbelt
53f59150dabfaede4af8045b	3	David	M. Blei
53f4448ddabfaee1c0ae5538	4	Panos	K. Chrysanthis

only showing top 20 rows

Figure 4.24: Result of query 4

5. **For each keyword, retrive the author(s) with the highest number of publications that contain that keyword**
   First of all, we want to find, for each keyword, the maximum number of publications of the same author that contain it. So, we select “id”, “keywords” and explode authors and rename it to “author”. Then, we select “id”, “authors” and explode “keywords” at this time. After that, we group by the author and the keyword and count the number of times that keyword appears in the publications of that author. Then, we group by the keyword and we select the maximum number retrieved in the previous step.
   Then we use the above result for finding, for each keyword, the authors with the most publications that contain it. To do this, first, we do the same job as the past and select “id” and “keywords”, and we explode the authors’ array and rename it to “author”. Then, we select “id”, and “authors” and explode “keywords”, and group by the author and keyword in order to retrieve the number of publications of that author that contain that keyword (this number will be inserted in the "number" field). Then apply an inner join between max-keyword and keywords in the publications data frame. We can now filter the result by keeping only the authors that have "number" equal to the maximum number (i.e. the author(s) that has the maximum number of publications for each keyword). Finally, we join the retrieved dataframe with the authors’ dataframe (in order to retrieve the firstname and lastname of the authors) and we select only the attributes that are meaningful for the results. After

that, we sort the result in ascending order with respect to the index of the keyword.

```
max_keyword = publications_spark.select(col('_id'),col('keywords'), explode(col('authors'))).withColumnRenamed('col', 'author') \
                        .select(col('_id'),col('author'), explode(col('keywords'))).withColumnRenamed('col', 'keyword') \
                        .groupby(col('author'), col('keyword')).agg(count('_id').alias('number')).groupby('keyword').max('number')

df = publications_spark.select(col('_id'),col('keywords'), explode(col('authors'))).withColumnRenamed('col', 'authors') \
                        .select(col('_id'),col('authors'), explode(col('keywords'))).withColumnRenamed('col', 'keywords') \
                        .groupby(col('authors'), col('keywords')).agg(count('_id').alias('number')) \
                        .join(max_keyword, col('keywords') == max_keyword.keyword) \
                        .filter(col('number') == col('max(number)')) \
                        .select(col('authors'), col('keyword'), col('number')) \
                        .join(authors_spark, col('authors') == authors_spark._id) \
                        .select(col('authors'), col('keyword'), col('number'), col('firstname'), col('lastname')) \
                        .join(keywords_spark, col('keyword') == keywords_spark.key_index) \
                        .select(col('authors'), col('keyword'), col('number'), col('firstname'), col('lastname'), col('key_name')) \
                        .sort(col('keyword').asc()) \

df.show(truncate=False)
```

Figure 4.25: Query 5

```
+------------------------+-------+------+-----------+---------------+----------------------------+
|authors                 |keyword|number|firstname  |lastname       |key_name                    |
+------------------------+-------+------+-----------+---------------+----------------------------+
|53f46797dabfaeb22f542630|0      |1     |Jairo      |Rocha          |handwriting recognition     |
|54328883dabfaeb4c6a8a699|0      |1     |Theo       |Pavlidis       |handwriting recognition     |
|53f431f8dabfaee0d9b372ae|1      |2     |Maria-Elena|Nilsback       |shape                       |
|53f46ca8dabfaec09f2584aa|1      |2     |Andrew     |Zisserman      |shape                       |
|54328883dabfaeb4c6a8a699|1      |2     |Theo       |Pavlidis       |shape                       |
|53f43abddabfaeecd6988ae9|1      |2     |David      |W. Jacobs      |shape                       |
|53f4345fdabfaedf4356c5e7|2      |2     |Laurent    |Itti           |feature extraction          |
|54328883dabfaeb4c6a8a699|2      |2     |Theo       |Pavlidis       |feature extraction          |
|54353ae1dabfaebba58b6925|2      |2     |Laurent    |D. Cohen       |feature extraction          |
|53f4329adabfaec09f1566a7|2      |2     |Ali        |Borji          |feature extraction          |
|53f467aedabfaedf4364b3bf|3      |1     |Raymond    |McCall         |knowledge base              |
|53f5757bdabfae7e2df8045b|3      |1     |Steven     |Feiner         |knowledge base              |
|53f45bbbdabfaee0d9c07abd|3      |1     |Frank      |M. Shipman, III|knowledge base              |
|53f46797dabfaeb22f542630|3      |1     |Jairo      |Rocha          |knowledge base              |
|540570f0dabfae92b41d421a|3      |1     |Blair      |Macintyre      |knowledge base              |
|54328883dabfaeb4c6a8a699|3      |1     |Theo       |Pavlidis       |knowledge base              |
|53f43918dabfaeb22f48b756|3      |1     |Dorée      |Seligmann      |knowledge base              |
|54328883dabfaeb4c6a8a699|4      |2     |Theo       |Pavlidis       |prototypes                  |
|54328883dabfaeb4c6a8a699|5      |3     |Theo       |Pavlidis       |optical character recognition|
|543348c4dabfaeb4c6ab4c32|6      |2     |Boaz       |Barak          |computer science            |
+------------------------+-------+------+-----------+---------------+----------------------------+
only showing top 20 rows
```

Figure 4.26: Result of query 5

6. **Retrieve the authors that have contributed to publications that contain keywords like "np complete", "np hard" or "algorithm" in three different years and all publications of those authors that contain those keywords.** First of all, we want to explode the "authors" and "keywords" arrays. Then we join the publications with the keywords data frame and we keep only the tuples that have a keyword that contains both the words "np" and "hard" or both the words "np" and "complete" or the word "algorithm"; to do so we use the "rlike" function that allows us to write these conditions in terms of regular expressions. Then we group by the id of the publications and by the orcid of the author and we collect the keyword names in an array since we want to maintain the keywords related to

a certain publication together for visualization purposes in the result. To select the authors that have contributed to these types of publications in three different years we group by the orcid of the authors and count the number of distinct years. By doing this, we also collect in two arrays the titles of the publications and the arrays of keywords. Then we join the resulting dataframe to the authors' dataframe in order to retrieve the firstname and lastname of each author. After this step we can finally explode two arrays (one containing the titles of the publications and one containing the arrays of keywords for each publication); we use the "array_zip" function since we want to associate each title to the corresponding array of keywords (the classic explode function would have needed to be performed two times, one for the titles and one for the keywords, but by doing this we would have obtained a cartesian product of the two arrays).

```
df = publications_spark.select(col('_id'), col('title'), col('year'), col('keywords'), explode(col('authors'))) \
        .withColumnRenamed('col', '_orcid') \
        .withColumnRenamed('_id', 'pub_id') \
        .select(col('pub_id'), col('title'), col('year'), explode(col('keywords')), col('_orcid')) \
        .withColumnRenamed('col', 'keyword_id') \
        .join(keywords_spark, col('keyword_id') == keywords_spark.key_index, 'inner') \
        .filter(col("key_name").rlike("np.*hard") | col("key_name").rlike("np.*complete") | col("key_name").rlike(".* algorithm.*")) \
        .groupBy(['pub_id', '_orcid']) \
        .agg(collect_set('key_name').alias('keywords'), first("title").alias("title"), first('year').alias('year')) \
        .groupBy('_orcid') \
        .agg(collect_set('title').alias('titles_set'), collect_list('keywords').alias('keywords_set'), countDistinct("year").alias('years')) \
        .filter(col('years') > 2) \
        .join(authors_spark, col('_orcid') == authors_spark._id, 'inner') \
        .withColumn("tmp", arrays_zip("titles_set", "keywords_set")) \
        .withColumn("tmp", explode("tmp")) \
        .select(col('firstname'), col('lastname'), col("tmp.titles_set").alias("title"), col("tmp.keywords_set").alias("keywords"))

df.show(truncate = False)
```

Figure 4.27: Query 6

```
+---------+-------------+-----------------------------------------------------------------------------------------+-----------------------------------------------------------------------------------------+
|firstname|lastname     |title                                                                                    |keywords                                                                                 |
+---------+-------------+-----------------------------------------------------------------------------------------+-----------------------------------------------------------------------------------------+
|Ron      |Alford       |A hierarchical goal-based formalism and algorithm for single-agent planning              |[translation algorithm]                                                                  |
|Ron      |Alford       |The GoDeL planning system: a more perfect union of domain-independent and hierarchical planning|[planning algorithm]                                                                  |
|Ron      |Alford       |Translating HTNs to PDDL: a small amount of domain knowledge can go a long way           |[planning algorithm]                                                                     |
|Jochen   |Alber        |Fixed Parameter Algorithms for PLANAR DOMINATING SET and Related Problems                |[fixed parameter algorithms]                                                             |
|Jochen   |Alber        |Efficient Data Reduction for DOMINATING SET: A Linear Problem Kernel for the Planar Case |[np hard problem, faster exact algorithm, exact algorithms,                              |
|Jochen   |Alber        |Faster exact algorithms for hard problems: a parameterized point of view                 |[np-complete dominating set problem]                                                     |
|Natalia  |V. Shakhlevich|Scheduling two jobs with fixed and nonfixed routes                                      |[np-hard problem., polynomial time algorithm]                                            |
|Natalia  |V. Shakhlevich|Shop-scheduling problems with fixed and non-fixedmachine orders of the jobs              |[pseudopolynomial algorithm]                                                             |
|Natalia  |V. Shakhlevich|Complexity of mixed shop scheduling problems: A survey                                  |[np-hard problem, polynomial algorithm]                                                  |
|Rolf     |Niedermeier  |An efficient exact algorithm for constraint bipartite vertex cover                      |[efficient exact algorithm]                                                              |
|Rolf     |Niedermeier  |Fixed Parameter Algorithms for PLANAR DOMINATING SET and Related Problems                |[fixed-parameter-tractable algorithm]                                                    |
|Rolf     |Niedermeier  |A general method to speed up fixed-parameter-tractable algorithms                        |[np complete problem, np complete problems]                                              |
|Rolf     |Niedermeier  |Efficient Data Reduction for DOMINATING SET: A Linear Problem Kernel for the Planar Case |[fixed parameter algorithms]                                                             |
|Rolf     |Niedermeier  |New upper bounds for maximum satisfiability                                              |[np hard problem, faster exact algorithm, exact algorithms,                              |
|Rolf     |Niedermeier  |An efficient fixed-parameter algorithm for 3-hitting set                                 |[np-complete dominating set problem]                                                     |
|Rolf     |Niedermeier  |Faster exact algorithms for hard problems: a parameterized point of view                 |[efficient fixed-parameter algorithm, time algorithm, np com                             |
|Ugur     |Kuter        |A hierarchical goal-based formalism and algorithm for single-agent planning              |[translation algorithm]                                                                  |
|Ugur     |Kuter        |The GoDeL planning system: a more perfect union of domain-independent and hierarchical planning|[planning algorithm]                                                                  |
|Ugur     |Kuter        |Translating HTNs to PDDL: a small amount of domain knowledge can go a long way           |[planning algorithm]                                                                     |
|Dana     |Nau          |A hierarchical goal-based formalism and algorithm for single-agent planning              |[translation algorithm]                                                                  |
+---------+-------------+-----------------------------------------------------------------------------------------+-----------------------------------------------------------------------------------------+
only showing top 20 rows
```

Figure 4.28: Result of query 6

7. **Retrieve, for each year, the number of publications that their citations are greater than 5, and in their titles exist the word "solution"**

First, we select columns “id”, “n-citation”, “year”, and “title”. Then we filter the selected items with two conditions: if the number of citations is greater than 5 and the title is like “solution”. After that, we group by the year column and finally, for each year, we count the number of ids with the above conditions. Here are the results:

```
df = publications_spark.select(col('_id'),col('n_citation'), col('year'), col('title')) \
      .withColumnRenamed('n_citation', 'citation') \
      .filter(col('citation') > 5) \
      .filter(col("title").like("%solution%")) \
      .groupBy(col('year'))\
      .agg(count('_id').alias('num publications'))

df.show(truncate = False)
```

Figure 4.29: Query 7

```
+----+----------------+
|year|num publications|
+----+----------------+
|2003|1               |
|2006|1               |
|1997|2               |
|1994|2               |
|2004|1               |
|1985|1               |
|1995|1               |
|2001|1               |
|1992|1               |
|2005|4               |
|1986|1               |
+----+----------------+
```

Figure 4.30: Result of query 7

8. **In ascending order, retrieve the publication’s titles which have less than 4 authors (with the number of authors sorted in descending order).**
   First, we select all the titles contained in the "publications" data frame and exploded the list of authors associated with these titles. Exploding the list of authors allowed

us to use the "groupBy" operation on the different titles in order to sum each author separately who contributed to the same publication. After that, we filter on the number of authors calculated by imposing it to be less than or equal to 4. Finally, we sort the titles in ascending order and the number of authors in descending order. Here are the results obtained:

```
exploded_df = publications_spark.select(col("title"), explode(publications_spark.authors))

exploded_df.groupBy("title") \
    .agg(count("col").alias("number of authors")) \
    .filter(col("number of authors") < 4) \
    .sort(col('number of authors').desc(), col('title').asc()) \
    .show(truncate = False)
```

Figure 4.31: Query 8

```
+----------------------------------------------------------------------------------------------------+-----------------+
|title                                                                                               |number of authors|
+----------------------------------------------------------------------------------------------------+-----------------+
|"GrabCut": interactive foreground extraction using iterated graph cuts                              |3                |
|/spl Hscr//sub /spl infin// filtering for continuous-time linear systems with delay                 |3                |
|3-D Gradient Coil Design—Initial Theoretical Framework                                              |3                |
|A                                                                                                   |3                |
|A Case Study of Case-Based CBR                                                                      |3                |
|A Fast Neighbor Discovery and DAD Scheme for Fast Handover in Mobile IPv6 Networks                  |3                |
|A Fast and Secure Image Hiding Scheme Based on LSB Substitution                                     |3                |
|A GIS tool for hydrogeological water balance evaluation on a regional scale in semi-arid environments|3                |
|A Gait Generation for an Unlocked Joint Failure of the Quadruped Robot with Balance Weight          |3                |
|A Generalized Rapid Development Environment for Cellular Automata Based Simulations                 |3                |
|A Language for Conveying the Aliasing Properties of Dynamic, Pointer-Based Data Structures          |3                |
|A Lightweight Coordination Middleware for Mobile Computing                                          |3                |
|A Low Bandwidth Broadcasting Protocol for Video on Demand                                           |3                |
|A Meta-Learning Method to Select the Kernel Width in Support Vector Regression                      |3                |
|A Methodology for Classifying Self-Organizing Software Systems                                      |3                |
|A New Proposal for RSVP Refreshes                                                                   |3                |
|A Regression-Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications      |3                |
|A Review of the Development and Performance of the ARPANET Routing Algorithm                        |3                |
|A Semi-Fragile Watermark Scheme For Image Authentication                                            |3                |
|A TSAR model for daily evapotranspiration at broad spatial scales: A case study in Northern China  |3                |
+----------------------------------------------------------------------------------------------------+-----------------+
only showing top 20 rows
```

Figure 4.32: Result of query 8

9. **Retrieve the number of citations and the titles of the 11 publications with the least number of pages published in a journal**
   First, we selected the types and their corresponding ids in the "venues" data frame. Then, we filtered on the types of venues in order to obtain only the "journals". This action allowed us to apply a "join" between the "publications" data frame (containing the id of each venue) and the ids of the venues corresponding only to the "journal" type. After that, we had access to the necessary data contained in the "publications" data frame (start_page, end_page, n_citation and title).

   Then we calculated the number of pages per journal (end_page - start_page +1) and sorted them in ascending order. Finally, we had to filter on the number of pages

by making sure they were not 'null' values (in the case of missing data) and retrieve the first 11 lines of the query. Here are the results:

```
retrieved_df = venue_spark.select(col('type'),col('_id')) \
               .withColumnRenamed('_id', '_id_journals') \
               .filter(col("type") == "journal") \
               .join(publications_spark, col('_id_journals') == publications_spark.venue_id)\
               .select(col('page_start'), col('page_end'), col('n_citation'),col('title'), col('type'))

retrieved_df = retrieved_df.withColumn('nb_pages', (col('page_end') - col('page_start') + 1)) \
               .sort(col('nb_pages').asc()) \
               .filter(col('nb_pages').isNotNull()) \
               .limit(11) \
               .select(col('type'), col('title'), col('nb_pages'), col('n_citation')) \
               .show(truncate=False)
```

Figure 4.33: Query 9

```
+-------+-----------------------------------------------------------------------------------------------------------------------+--------+----------+
|type   |title                                                                                                                  |nb_pages|n_citation|
+-------+-----------------------------------------------------------------------------------------------------------------------+--------+----------+
|journal|Fuzzy logic, neural networks and soft computing                                                                        |1.0     |73        |
|journal|GCP                                                                                                                    |1.0     |1         |
|journal|A Semi-Fragile Watermark Scheme For Image Authentication                                                               |1.0     |71        |
|journal|Vulnerability of water quality in intensively developing urban watersheds.                                             |2.0     |0         |
|journal|The Complex Nature of e-Government Projects: A Case Study of Bhoomi, an Initiative in Karnataka, India                 |2.0     |6         |
|journal|2k-AryExponentiation.                                                                                                  |2.0     |0         |
|journal|Compound Analytics of Compound Data within RDBMS Framework – Infobright's Perspective                                 |2.0     |3         |
|journal|Distributed power control in cellular radio systems                                                                    |3.0     |283       |
|journal|Infinite-horizon differential games of singularly perturbed systems: a unified approach                               |4.0     |16        |
|journal|Catastrophic cascade of failures in interdependent networks.                                                           |4.0     |2548      |
|journal|Delay-Dependent Robust H∞ Filtering for Uncertain Discrete-Time Systems With Time-Varying Delay Based on a Finite Sum Inequality.|5.0     |114       |
+-------+-----------------------------------------------------------------------------------------------------------------------+--------+----------+
```

Figure 4.34: Result of query 9

10. **Retrieve all the publications that have been published in the first or second edition of a volume of a journal (issue = 1 or issue = 2) and that have a reference to another publication published by North-Holland**
    First, we filter on the venues data frame in order to retrieve only the rows where the publisher was "North-Holland". Thanks to this, we are able to apply a join between the publications data frame and the ids of the venues regarding North-Holland. Then, this action allowed us to collect the corresponding ids in a list.

    After that, we filter the publications data frame with the condition that the publications had to contain at least "1" or "2" as the issue (edition). Thus, we were able to select the different attributes allowing us to answer our query and the imposed condition (_id, title, issue, volume). In addition, we explode the "references" column in order to filter only on the list of ids of the venues collected previously ("North-Holland"). Finally, we sorted the results in ascending order of titles. Here are the results:

```
north_holland_ids = venue_spark.filter(col('publisher') == 'North-Holland') \
                      .withColumnRenamed('_id', 'venue') \
                      .join(publications_spark, col('venue') == publications_spark.venue_id, 'inner') \
                      .select(collect_list("_id"), collect_list('title')).collect()[0][0]

df = publications_spark.select(col('_id'), col('title'), col('issue'), col('volume'), col('references')) \
            .filter(((col('issue') ==  '1') | (col('issue') == '2') | col('issue').rlike('[12]-.*')) & col('volume').isNotNull()) \
            .select(col('_id'), col('title'), col('issue'), col('volume'), explode(col('references'))) \
            .withColumnRenamed('col', 'reference') \
            .filter(col('reference').isin(north_holland_ids)) \
            .sort(col('title').desc())

df.show(truncate=False)
```

Figure 4.35: Query 10

```
+------------------------+-----------------------------------------------------------------------------------------------------------------+-----+------+------------------------+
|_id                     |title                                                                                                            |issue|volume|reference               |
+------------------------+-----------------------------------------------------------------------------------------------------------------+-----+------+------------------------+
|53e9984fb7602d9702084b0d|The SAT2002 competition                                                                                          |1-4  |43    |53e9b9adb7602d970459fe66|
|53e9978ab7602d9701f4a085|N                                                                                                                |2    |156   |53e9aacab7602d9703446c42|
|53e9978db7602d9701f4f82e|H                                                                                                                |2    |44    |53e9b24db7602d9703cec51a|
|53e9978db7602d9701f4f662|H                                                                                                                |2    |158   |53e9abd4b7602d970358e48f|
|53e9bd8cb7602d9704a32148|An axiomatic approach to the definition of the entropy of a discrete Choquet capacity                            |1-2  |172   |53e9a877b7602d97031c12be|
|53e99792b7602d9701f5b119|A fuzzy multi-objective programming for optimization of fire station locations through genetic algorithms|2    |181   |53e9bbf5b7602d9704850823|
|53e99792b7602d9701f5b119|A fuzzy multi-objective programming for optimization of fire station locations through genetic algorithms|2    |181   |53e9ab42b7602d97034d404d|
+------------------------+-----------------------------------------------------------------------------------------------------------------+-----+------+------------------------+
```

Figure 4.36: Result of query 10

5 | References

- Neo4j dataset source
- Apoc json documentation
- DBLP website
- Mockaroo website
- Pandas documentation