POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Christian Andreoli**
Pietro Caforio
Maria Cristina Centorame
Sara Merengo
Nicolò Francesco Resmini

Group Number: **19**

Academic Year: 2022-2023

Contents

1 | Introduction

1.1. Problem Specification

The purpose of this project is to build an information system that supports computer science researchers in their daily efforts by providing access to bibliographic meta-data and links to the electronic editions of publications in the field.

The database should be able to register all the available information and relations between publications in the computer science field to provide a simple interface for information retrieval, for example using an hypothetical web API.

1.1.1. Possible use cases

The main possible use case could be providing a simple User Interface to simplify information retrieval for a variety of users, for example:

1. Fresh graduates who want to find the best universities based on their interests (see query 6.1.3).
2. Companies searching for top researchers or top universities for recruiting (see query 6.1.3 or 6.1.4).
3. Data analyzers who want to retrieve and study global information about the entire domain (see query 6.1.10).
4. Researchers who want to find the best publications based on their content or main topic (see queries from 6.1.7 to 6.1.9).
5. University's executives who want to analyze the social structure based on authors working on common publications (see query 6.1.11).

1.2. Hypotheses

The assumptions taken into account are the following:

- The data collecting backend system assigns a category to each publication (for example article, book,...).
- The data collecting backend system provides the possibility to add keywords with respect to the publication's content.
- A publication has at least one author.
- The data collecting system always associates a valid orcid to every author.
- The editor of a publication or collection is the specific person curating it.
- The data collecting system registers authors' web pages. Each web page is linked to exclusively one author.
- The data collecting system keeps track of the number of visits of each web home page of the authors every month.
- The data collecting system registers affiliations from authors' websites, but exclusively affiliations with Universities.
- Among thesis, the data collecting system registers phd theses exclusively.

2 | ER Diagram

In the first group meetings we talked about which shape the database should have, according to our basic knowledge on bibliographic resources. We immediately separated the two main entities (Person and Work) and decided to put them on top of hierarchies in the ER diagram; however, we later agreed on the fact that "Work" should be split in two other levels, to highlight the difference between single resources and collections. In fact, the former (named "Publications" in the ER) had to be connected to "Authors", whereas the latter should refer to "Editors" because a collection just gathers already existing works, which could have different authors.

The only entity connected to both types of "Person" is "Book", because in this case both relations were meaningful and equally important. A book can be a "Work" but it can also be a collection of shorter publications, which in our application are represented by the "Incollection" category.

"Website" is not considered a pubblication because it refers to authors' webpages and not to their works, therefore we made it a separated entity.

Finally, here are some helpful notes to understand the ER diagram:

- the relationships must be read from left to right and from top to bottom in order to understand the name given to them;
- in the relationships, cardinalities refer to the entity they are closest to, w.r.t the rhombus of the relationship; for example, looking at the diagram we obtain "An Inproceeding CAN be gathered in ONE Proceeding, whereas a Proceeding MUST gather MORE Inproceedings";
- for the entity "Publication", "Keywords" is a multivalued attribute and instead of representing it with a double circle, we put the cardinality next to it to highlight that it can have multiple values; the same applies to the attribute "URL" of the entity "Website";
- for the entities "Book" and "Proceeding", the attribute "volume" refers to the

attribute "series" with the former being the relative number of the book inside the latter;

- to emphatize primary keys we colored them grey instead of using the classic underline;
- for what regards the ISA hierarchies, the one between Person and Editor/Author is overlapping because a person can be both an author and an editor, whereas the one between Publication and the types of publications is exclusive since a work cannot be part of two different categories (like "Book" and "Article") at the same time;
- a more precise description of entity categories and attributes as they appear in the actual used dataset can be found in the following "Data description" chapter;
- further information can be found at https://dblp.org/xml/docu/dblpxml.pdf, which is a paper containing a review of the evolution of DBLP.

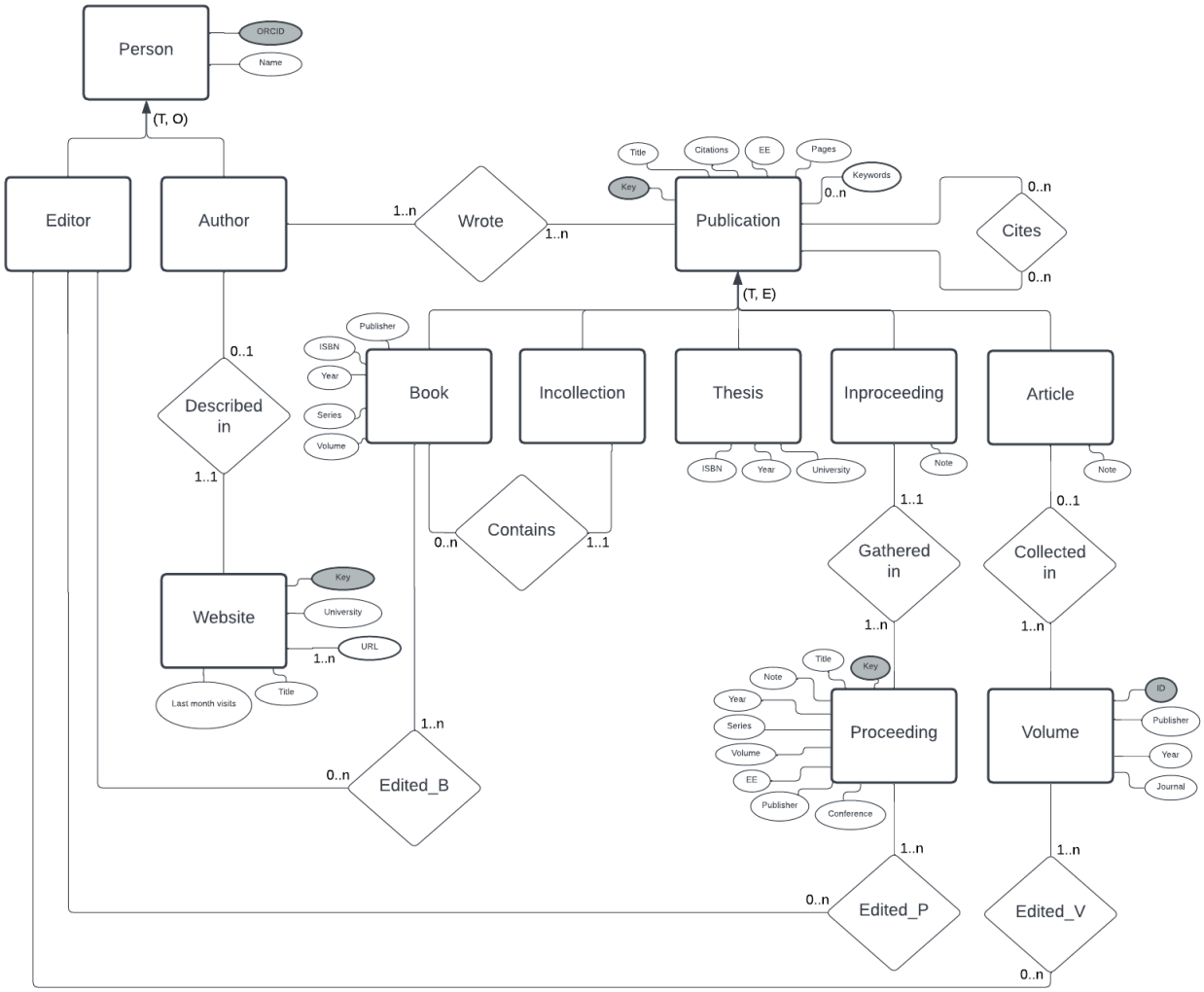

Figure 2.1: ER diagram

3 | Data Description

We decided to use the official dblp dataset available in an xml file at https://dblp.uni-trier.de/xml/ because of its variety in types of entries. We used the most recent version of the dataset, which is about 3.5GB in xml format. The dataset is updated to October 2022 and contains more than 9 million entries, divided among the following categories:

1. "article" (total number: 2971780);
2. "book" (total number: 19547);
3. "incollection" (total number: 68690): refers to publications contained in a larger collection such as a book;
4. "phdthesis" (total number: 95062);
5. "proceeding" (total number: 52069): refers to publications related to a particular convention or conference;
6. "inproceeding" (total number: 3109603): refers to articles or other types of papers contained in the publications related to a convention ("proceedings");
7. "www" (total number: 3091995): refers to homepages of authors' or editors' personal websites;
8. "masterthesis" (total number: 15);
9. "person" and "data": no entries of this type were found in the dataset so these categories were ignored.

The "masterthesis" category was also ignored in the application because Master Theses are usually less impactful in the scientific world than Phd Theses (which is also confirmed by their small number in the original dataset), so we decided to exclude them from our application.

The most relevant attributes and subelements in most categories (the website category was obviously fairly different from the others) were:

1. "key" attribute: string representing a unique key in the dataset for each entry. Note that this key is unique internally to the dataset and does not comply with any official regulation.
2. "mdate" attribute: represents the date the entry was last updated or modified in the dataset. The documentation states that this field is often not updated due to problems with the handling of the database. Due to this fact and to the scarce relevance of this information for the type of queries we wanted to perform, the attribute was ignored.
3. "title": title of the publication. In the "www" category, this is set to "Home Page" for all entries considered.
4. "author": name of the author, sometimes with the orcid as an attribute. Each entry can have zero, one or multiple authors.
5. "editor": same structure as the "author" field. In this dataset the editor attribute always contains the name of a person, sometimes with the associated orcid, so we made the assumption that it represents the specific person or people curating a particular publication. For the "proceedings" category, the documentation of the dataset states that for most entries the editor(s) listed are the leading organizers of the conference.
6. "year": publishing year.
7. "publisher": name of the publisher.
8. "isbn": isbn code of the publication. Each publication can contain multiple isbn fields.
9. "cite": reference to other publications in the dataset that are cited in the publication. This field can also contain multiple citations.
10. "crossref": reference to another specific publication in the dataset that this entry is a part of. For example for incollection this attribute contains the key of the book the paper was published in.
11. "school": for publications that are associated with a university.
12. "ee": contains one or more urls referencing the doi code of the publication or other unique identifiers (this is different from the "key" field because the key is internal to the dataset, while this references external identifiers).
13. "url": for publications, this contains an address internal to the database. Since we

did not have access to a method for converting these addresses into usable urls, this field was ignored. For the "www" category, this contains one or more urls referencing authors' personal web pages or other web pages related to them (university page, wikipedia,...).

In the "www" category, the "author" field contains the name of person the homepage refers to. The original dataset has some problems handling authors' names because it does not require the orcid key for authors and so it has to rely mostly on the name string to identify them, therefore sometimes it happens that a website seems to be connected to multiple authors when actually it's just different instances of the same author due to their name being shortened in some publications (ex. a website listing as authors "John Smith" and "J. Smith").

Other notable attributes:

1. "series": for books, the name of the larger series the book is published in.
2. "journal": for articles, the name of the journal in which the article was published.
3. "volume": for articles, the number or other type of index of the specific volume or issue of a journal the article was published in. For books, the number or other type of index indicating the position of a book in a series it was published in.
4. For the website ("www") category, "note" attributes containing an affiliation to a university were kept while other types of notes were discarded.

4 | Data pre-processing and upload

4.1. Data Pre-Processing

The dblp dataset was used directly in its xml format and handled with the ElementTree library in Python and the APOC tool in Neo4j.

The xml file contained a reference to a dblp.dtd file which had additional information about the structure of the database and contained in particular a list of entities representing special characters. As Neo4j does not support this type of reference, it was necessary to replace all special characters to correctly process the dataset. This was done by copying the list of special characters codifications from the dtd file and replacing them in the xml file using a simple Python script reading the file as text.

The dataset was then split according to the different categories of entries contained within it so that the files to be processed would be smaller and so that each category could be handled separately.

Each split dataset was then further filtered according to its own data structure and according to the following criteria:

1. Relationships between entries were preserved as much as possible in order to build a more connected graph. For example the article category had an attribute for references to other publications, so entries containing values for that attribute were privileged in the filtering.

2. Where this was not possible, for example in the book category which contains no references in the original dataset, the entries were filtered according to the publication year (keeping more recent entries) so that it was more likely to have more books contained in the same series or written by the same author(s).

3. When possible, relevant data was kept. For example entries containing at least one valid orcid, a publication year or a publisher were privileged in the filtering.

4. If after these steps the dataset was still too big, a random selection was performed

over the remaining entries.

5. The inproceeding category was filtered by selecting only entries that referenced proceeding elements that were kept in the dataset after filtering.
6. The Website category was filtered by selecting only the websites of authors and editors that were actually present in the dataset after filtering the other datasets.

After considerably reducing the size of the dataset, we further processed it to add attributes that were relevant to our application. The most significant changes were:

1. As we intended to use the orcid as key for the Person category (author or editor) in the database, an orcid was added to all author and editor entities. Where a valid orcid was already present in the dataset element it was kept, while where it was missing we set it to the same string used for the name of the person. In theory, this does not create problems with homonyms because the dblp database already distinguishes between homonyms in the name string (for example "John Smith 0001" is a different person from "John Smith 0002"). In practice as the dblp dataset does not have an orcid for all authors it has problems keeping track of different version of the same person's name, but we chose to ignore these issues for simplicity's sake.
2. The "pages" attribute was modified. While in the original dataset it indicated in which pages of the corresponding volume an article or paper was present, we used it to signify the total number of pages of the publication and generated random values for entries that previously had the "pages" attribute.
3. A citations number attribute representing the number of other works that cite a publication was added to publication categories. The attribute was randomly generated.
4. Keywords were added to publications to search for topics more effectively. Keywords attributes were randomly generated from a list of about 40 topics relating to computer science. Each publication was assigned a number of keywords between 0 and 10, with a bigger number of keywords being more unlikely.
5. A last month visits attribute was added to the Website category. This was also randomly generated. Since a Website could contain more than one url, the meaning of the "last_month_visits" attribute is that it's the sum of visits across all websites listed.

With the exception of the orcid, which was used as a key, all other attributes were generated only for a part of the entries to keep the dataset heterogeneous.

All pre-processing operations were performed with the help of the ElementTree library in Python.

The xml dataset is structured as a tree. There is a root node called "dblp" and all publication and website entries are children of this node. Each entry then has multiple children containing its attributes like title, author(s), editor(s), publisher,...

We report here one of the programs used for the processing of the dataset, in particular the one adding orcid attributes to all authors and editors. In this particular case the orcid is represented in the data tree as an attribute of the author and editor entities instead of as a child.

```
import xml.etree.ElementTree as ET

inputfile = "db-noorcid.xml"
outputfile = "db.xml"

with open(inputfile) as input:
    #parses the file and creates a tree from the data
    input_tree = ET.parse(inputfile)
    input_root = input_tree.getroot()

    #iterates over children of the tree root
    for element in list(input_root):
        #finds all children of element with the "author" or "editor" tag
        author = element.findall("author")
        editor = element.findall("editor")

        for a in author:
            if("orcid" not in a.attrib):
                a.set("orcid", a.text)
        for e in editor:
            if("orcid" not in e.attrib):
                e.set("orcid", e.text)

input_tree.write(outputfile)
```

This script builds a tree out of the input xml file and then iterates over the children of the root node. In this case the children of the root node are all the entries of the

database. For every entry, the script finds all subelements of type "author" and "editor" and checks their attributes for the "orcid" attribute, setting it to the same string as the "author"/"editor" subelement (a.text / e.text) if it does not find it.

After filtering, the size of the dataset was the following:

1. "article": 2014 elements
2. "book": 818 elements
3. "incollection": 2308 elements
4. "phdthesis": 2376 elements
5. "proceedings": 102 elements
6. "inproceedings": 2187 elements
7. "www": 2614 elements

For a total of almost 12 thousand entries.

4.2. Data Upload

The data was then imported into a Neo4j database using the APOC tool Neo4j provides.

We report here some example cypher code used for the import process, in particular the creation of "article" nodes and some of their relationships. The code for the creation of other nodes and relationships is fairly similar to this one, with only small changes depending on the data structure and which attributes were considered relevant.

For a more precise description of the nodes and relationships that were taken into consideration see the following chapter "Graph Diagram".

```
//Load article nodes
CALL apoc.load.xml("file:///article-db.xml") YIELD value
UNWIND value._children AS foo
WITH [x in foo WHERE x._type = 'article'] AS article_s
UNWIND article_s AS article
WITH article.key AS articleKEY,
     [item in article._children WHERE item._type = "title"][0] AS title,
     [item in article._children WHERE item._type = "note"] AS note_s,
     [item in article._children WHERE item._type = "pages"][0] AS pages,
     [item in article._children WHERE item._type = "citations"][0] AS citations,
     [item in article._children WHERE item._type = "ee"] AS ee_s,
```

```
MERGE (a:Publication:Article {key: articleKEY})
SET
    a.title = title._text,
    a.note = [note IN note_s | note._text],
    a.ee = [ee IN ee_s | ee._text],
    a.pages = toInteger(pages._text),
    a.citations = toInteger(citations._text)
RETURN count(a);


//Load Author nodes
CALL apoc.load.xml("file:///article-db.xml") YIELD value
UNWIND value._children AS foo
WITH [x in foo WHERE x._type = 'article'] AS article_s
UNWIND article_s AS article
WITH [item in article._children WHERE item._type = "author"] AS author_s
UNWIND author_s AS author
WITH author._text AS name, author.orcid as _orcid
MERGE (auth:Person {orcid: _orcid})
SET auth.name = name
RETURN count(auth);
// create relationships :AUTHOR_OF
CALL apoc.load.xml("file:///article-db.xml") YIELD value
UNWIND value._children AS foo
WITH [x in foo WHERE x._type = 'article'] AS article_s
UNWIND article_s AS article
WITH article.key AS articleKEY,
     [item in article._children WHERE item._type = "author"] AS author_s
UNWIND author_s AS author
WITH articleKEY, author.orcid AS _orcid
MATCH (a:Article {key: articleKEY})
MATCH (auth:Person {orcid:_orcid})
MERGE (auth)-[r:AUTHOR_OF]->(a)
RETURN count(r);

// create relationships :CITES
CALL apoc.load.xml("file:///article-db.xml") YIELD value
UNWIND value._children AS foo
WITH [x in foo WHERE x._type = 'article'] AS article_s
```

```
UNWIND article_s AS article
WITH article.key AS articleKEY,
     [item in article._children WHERE item._type = "cite"] AS cite_s
UNWIND cite_s AS cite
MATCH (a:Article {key: articleKEY})
MATCH (o:Publication {key: cite._text})
MERGE (a)-[r:CITES]->(o)
RETURN COUNT(*);
```

The different categories of nodes and relationships were created with different calls to the apoc.load.xml function for efficiency and memory reasons. For each call to the function, thanks to the "WITH" operator Neo4j can select only a portion of the data to process and keep in memory.

When importing the children of an element, by default Neo4j imports the selected children as an array. So for attributes of which there is only one per entry (such as "title") it is necessary to select only the first element of the array using "[0]".

All nodes and relationships were created using the "MERGE" function, which searches the database for duplicates before creating a new element. This is very important not only because some of the data may be redundant, but because some entities like authors, editors, publishers an keywords can appear multiple times. The "MERGE" function looks for elements matching the information provided between the brackets. Wherever it was possible, we used keys provided by the database to do this matching operation. This ensures that for example if an author appears in the database with two different names (for example "John Smith" and "J. Smith"), if the two instances have the same orcid attribute the node for the author is created only once.

5 | Graph Diagram

In designing the Neo4j graph diagram the ER diagram structure (see 2.1) has been extended with the purpose of making references and relationships among the different objects more explicit. The dataset at our disposal has been split into categories (article, incollection, book, inproceeding, proceeding, phd thesis and www) and each of them has been modeled with a different label. However, not all the attributes from the ER diagram have been modeled as actual properties of the respective node. For example focusing on the Book node, as shown in fig. 5.1, the fields of Keywords, Publisher, Year, Series and Volume have been changed into nodes linked through relationships (see 5.3) in order to make queries more efficient.

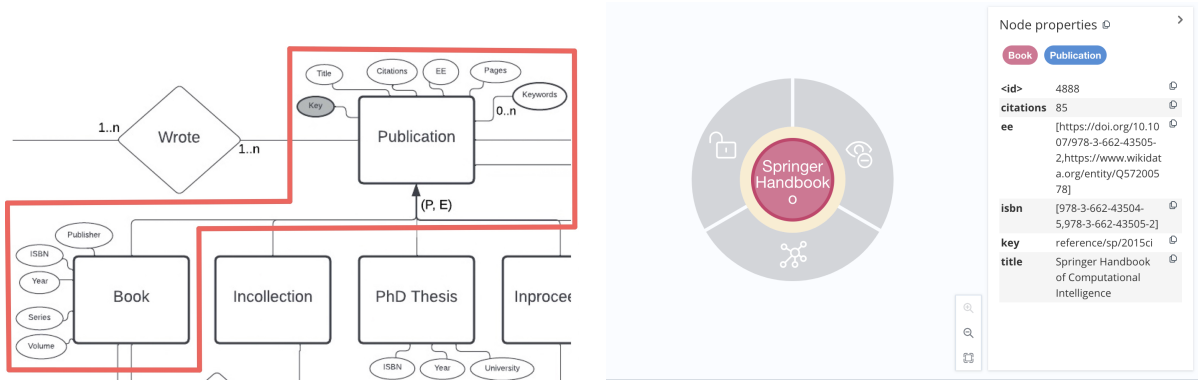

Figure 5.1: Differences between the ER diagram and the graph diagram for the Book category in terms of attributes

The same approach has been applied to the remaining categories which present a similar structure (except for the Website category), but we'll discuss their differences later on.

In addition, a node labeled as Person has been created representing both the editor and the author of a generic publication. The distinction between the two different roles has then been outlined through the definition of two types of relationships called respectively AUTHOR_OF and EDITOR_OF.

All publication categories (Article, Book, Incollection, Inproceeding, Phdthesis) also fall under the common label "Publication". This is both to make searching for generic pub-

lications easier if the category is not important and because the analysis of the dataset has brought out the presence of lots of interconnections between publications; these connections are more easily handled if there is a common "Publication" label.

Here follows a representative graph for each category accompanied by a short description of the reasons that stood behind the introduction of specific nodes and relationships: (NOTE: for each category, see the corrisponding graph diagram below)

- Article: An article could appear in a journal's volume which forms a self-contained unit provided with its own title, so, in order to link the article to its own journal and volume, two relationships have been created:

  1. COLLECTED_IN: it links the single article to the volume which it belongs to by using the number field as a relationship property and by selecting the corresponding volume node through a composite attribute consisting of both "journal" and "volume" since it may happen to have the same number of volume in different journals;

  2. REFERS_TO: it links the volume to the specific journal containing the article. This allows a direct access to the different journals and it simplifies the search for articles in a single journal.

  Moreover, the information regarding the year of publication and the actual publisher have been reasonably connected to the volume, while the author has been considered directly related to the article due the fact that a single volume usually includes several articles by different authors. Finally, an article can contain various citations to different publications, so the CITES relationship has been created linking an article to the publication(s) the key of which matches the content of the article "cite" field.

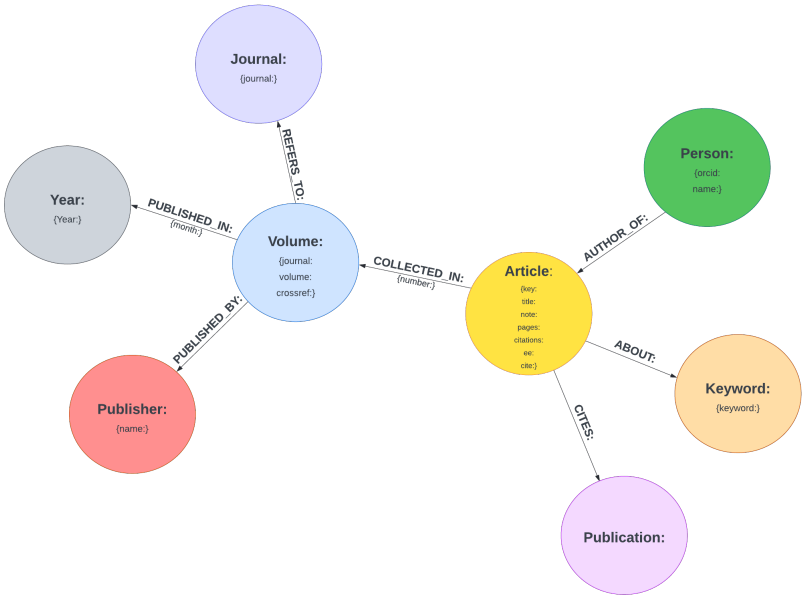

Figure 5.2: Article, graph diagram

- Book: It's a single-volume book that may be contained in a series. When the series is present, it's linked through the CONTAINS relationship, which is charachterized by the volume property indicating the ordinal position of the book in the series. In this case, the node Person is allowed to cover the role of both the author and the editor, and the publisher is directly referred to a book as well as the year of publication. Additionally, the image below (see 5.3) shows one more relationship called ABOUT that connects the book to its keywords.

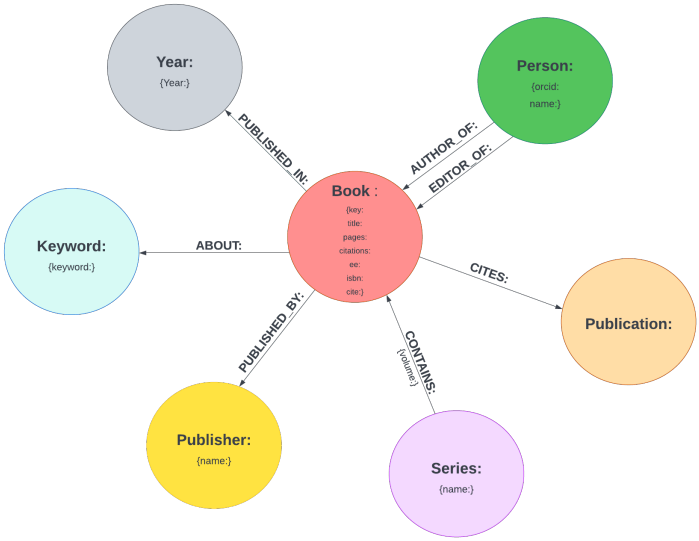

Figure 5.3: Book, graph diagram

- Incollection: It's a contribution to a collection with a distinct author and title. The COLLECTED IN relationship links the incollection to the Publication node selected through the "crossref" field, while the CITE relationship uses the "cite" field. Besides, the relationships regarding the publisher and the year of publication have been considered related to the collection labeled as Publication, which the single incollection is part of, therefore the Incollection graph diagram doesn't show any of them(see 5.4).

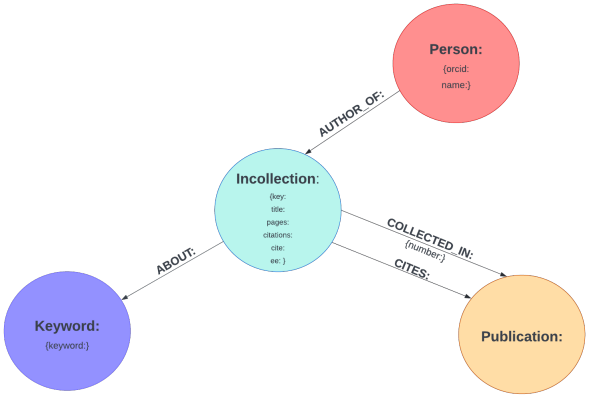

Figure 5.4: Incollection, graph diagram

- Proceeding: It's a single volume conference proceeding and the main differences from the other categories consist in the introduction of the Conference node, identified by the booktitle field, and in the new meaning of the Year node which here

represents the year in which the conference was held and not the one in which it was published. In this case a proceeding volume of a joint conference could be a memeber of several conference series, thus creating the need to distinguish between the PART_OF relationship, which includes the properties volume and number, and the REFERS_TO relationship, which refers to the conference series as a whole.

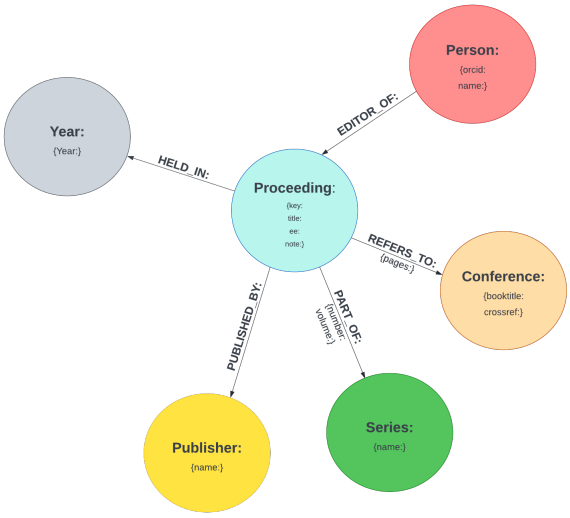

Figure 5.5: Proceeding, graph diagram

- Inproceeding: An article in a conference proceeding. Its graph is similar to the Incollection one except for the relationship called CROSSREF which links the inproceeding to its proceeding through the crossref field in the inproceeding.

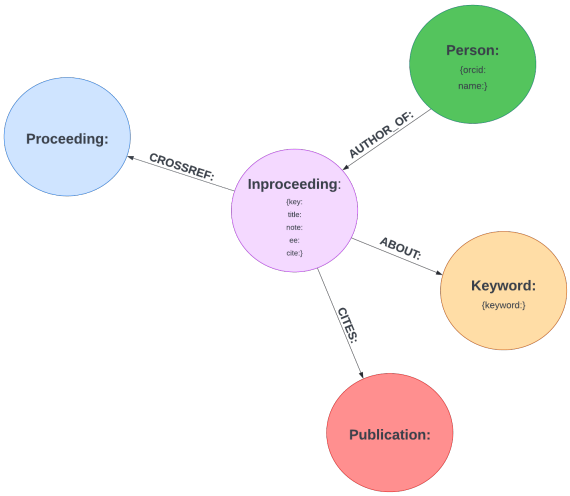

Figure 5.6: Inproceeding, graph diagram

- Phd thesis: Here, the University node has been added in order to provide information regarding where the thesis was produced and to allow to search for the range of a specific school academic production.

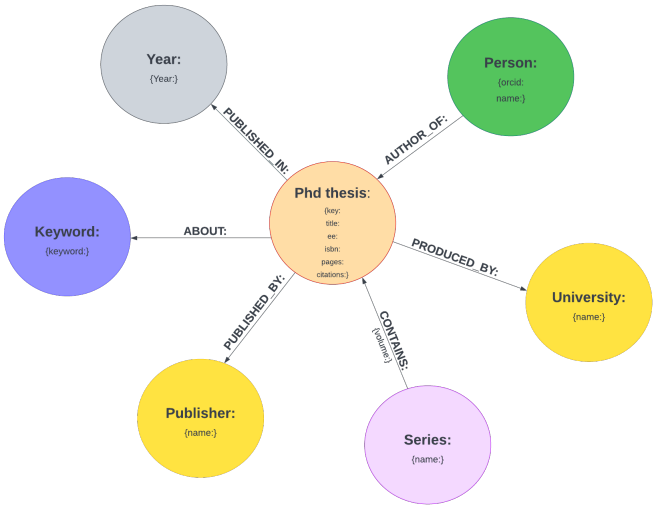

Figure 5.7: Phd thesis, graph diagram

- WWW: This category is the most different from the others since it collects authors' web pages. It presents only two relationships:
  1. HOMEPAGE_OF: it links the website to the author it refers to;
  2. AFFILIATED_TO: it connects the webpage to the author's affiliations which are indicated in the "note" field in the dataset.

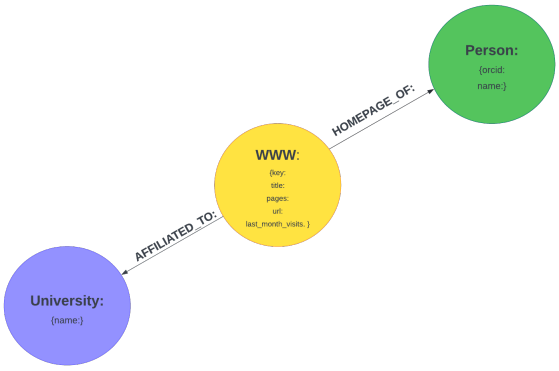

Figure 5.8: WWW, graph diagram

In the end, the whole database is composed as follow(see 5.9):

"labels(n)"	"SampleSize"	"Avg_PropertyCount"	"Min_PropertyCount"	"Max_PropertyCount"	"Avg_RelationshipCount"	"Min_RelationshipCount"	"Max_RelationshipCount"
["Publication","Article"]	2015	6.340942928039706	3	7	9.9801488833747	1	100
["Person"]	20589	2.0	2	2	1.2427024139103275	1	31
["Publication","Book"]	818	5.123471882640591	4	6	8.380195599021993	2	182
["Publication","Incollection"]	2308	5.284228769497394	3	6	4.666811091854418	0	50
["Journal"]	14	1.0	1	1	22.42857142857143	1	154
["Volume"]	314	2.0	2	2	9.127388535031853	2	92
["Proceedings"]	102	4.0	4	4	27.568627450980394	7	378
["Year"]	59	1.0	1	1	61.23728813559321	1	256
["Publisher"]	109	1.0	1	1	12.036697247706423	1	700
["Keyword"]	46	1.0	1	1	350.804347826087	1	391
["Series"]	132	1.0	1	1	4.083333333333334	1	61
["Conference"]	102	1.0	1	1	1.0	1	1
["Publication","Inproceedings"]	2188	4.996343692870218	4	5	4.587751371115172	1	17
["Publication","Phdthesis"]	2377	4.7193941943626365	4	6	5.282288599074469	3	14
["University"]	2115	1.0	1	1	2.3843971631205707	1	51
["Website"]	2615	3.0	3	3	2.2386233269598463	1	11

Figure 5.9: Database

6 | Queries and Commands

After understanding the dataset available to us and how we loaded it into the database, in this chapter we are going to look at some of the tools that Neo4j provides. Theese tools are useful to query the database in order to get relevant information, or to create and update data.
Understanding data can be a daunting task. Neo4j allows us to connect data to reveal hidden insights without using JOINs or lookups. So let's see some examples.

6.1. Queries

In this section we will analyze various useful queries to obtain the different informations we are looking for. It is impossible to cover all needs, so we propose at least one example per category, trying to show the full potential of Neo4j.

Before getting started, it is important to keep in mind some aspects:

1. How to visualize the data is a subjective choice; here we try to adopt different solutions
2. Whenever there is a CALL function, the query refers to its content; anything that is not in the CALL function is used for a better visualization of the data.
3. Whenever we refer to a publication, it is possible to replace it with an eligible subcategory (i.e. Article | Phdthesis | Inproceedings | Book | Incollection)
4. A single colored legend will be used throughout the chapter

Figure 6.1: Label legend

6.1.1. Most cited AUTHORS in a given field

The following query can be used to find the authors, and their website (if existing), who wrote about a given field (e.g. "data mining") ordering them by the sum of citations they received for thoose publications.

```
MATCH (a:Person)-[:AUTHOR_OF]-(p:Publication)-[:ABOUT]->(k:Keyword {keyword: "data mining"})
OPTIONAL MATCH (w:Website)-[:HOMEPAGE_OF]-(a:Person)
WHERE p.citations IS NOT null
WITH w AS Website, a AS Author, SUM(p.citations) AS cit
ORDER BY cit DESC
RETURN Website, Author, cit
LIMIT 10
```


"Website"	"Author"	"cit"
{"title":"Home Page","url":["http://www.cs.uic.edu/~yu/"],"key":"homepages/y/ClementTYu"}	{"name":"Clement T. Yu","orcid":"Clement T. Yu"}	231
{"title":"Home Page","key":"homepages/s/KennethCSevcik","url":["http://www.cs.utoronto.ca/~kcs/"]}	{"name":"Kenneth C. Sevcik","orcid":"Kenneth C. Sevcik"}	229
null	{"name":"K. Lam","orcid":"K. Lam"}	207
{"title":"Home Page","url":["http://dblp.org/db/about/ml/","https://www.uni-trier.de/universitaet/fachbereiche-faecher/fachbereich-iv/faecher/informatikwissenschaften/professuren/datenbanken-und-informationssysteme/team/dr-michael-ley","https://scholar.google.com/citations?user=2jE4KhkAAAAJ","https://dl.acm.org/profile/81100182162","https://orcid.org/0000-0001-7580-4351","https://www.wikidata.org/entity/Q6832241","https://www.scopus.com/authid/detail.uri?authorId=14826827100","https://viaf.org/viaf/305009965","https://id.loc.gov/authorities/names/n95055551","https://gepris.dfg.de/gepris/person/1553179","https://awards.acm.org/award_winners/ley_2903227","https://ieeexplore.ieee.org/author/37086187058","http://scigraph.springernature.com/person.011365533341.48"],"key":"homepages/00/1"}	{"name":"Michael Ley","orcid":"Michael Ley"}	171
null	{"name":"Bala Rajaratnam","orcid":"Bala Rajaratnam"}	148

Figure 6.2: Most cited authors in a given field.

6.1.2. Most active EDITORS

The following query can be used to find the top 5 editors who organised (editor of) the most number of conferences (proceedings) held after 2010.

```
CALL{
    MATCH (p:Person)-[:EDITOR_OF]-(c:Proceedings)-[:HELD_IN]->(y:Year)
    WHERE y.year >= 2010
    WITH p, COUNT(*) AS pcount
    ORDER BY pcount DESC
    RETURN p, pcount
    LIMIT 5
}
WITH p, pcount
MATCH (p:Person)-[:EDITOR_OF]-(c:Proceedings)
RETURN p,c
```

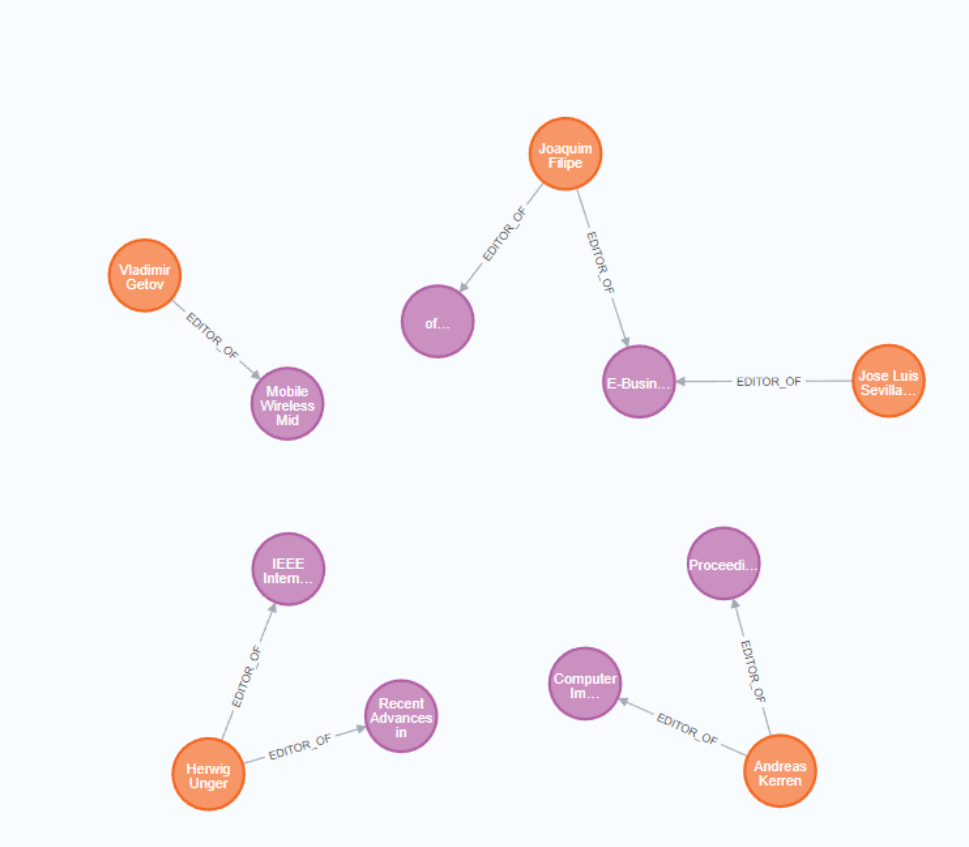

Figure 6.3: Most active editors.

6.1.3. Top UNIVERSITY per number of publications about a specific topic

The following query can be used to find the top 5 universities which have produced the highest number of phd thesis regarding a desired topic (e.g. 'data mining') in the last ten years

```
CALL{
    MATCH (u:University)<-[:PRODUCED_BY]-(p:Phdthesis)-[:ABOUT]->(k:Keyword),
          (p)-[:PUBLISHED_IN]->(y:Year)
    WHERE y.year > 2012 AND k.keyword = "data mining"
    WITH u, COUNT(DISTINCT p) AS pcount
    ORDER BY pcount DESC
    RETURN u, pcount
    LIMIT 5
}
WITH u, pcount
MATCH (u:University)<-[:PRODUCED_BY]-(p:Phdthesis)-[:ABOUT]->(k:Keyword),
      (p)-[:PUBLISHED_IN]->(y:Year)
WHERE y.year > 2012 AND k.keyword = "data mining"
RETURN u, p, k, pcount
```

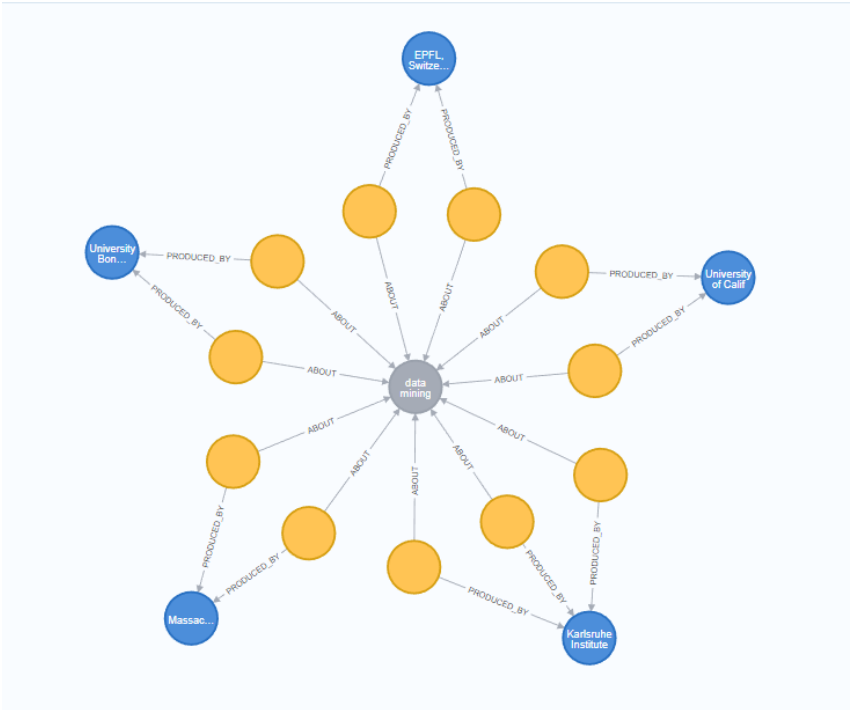

Figure 6.4: Top university per number of Phd Thesis about a specific topic.

Alternatively it is possible to find the top 5 universities which have produced the highest number of any pubblications written by a person affiliated to the university (i.e. professor)

```
CALL{
    MATCH (u:University)-[:AFFILIATED_TO]-(w:Website)-[:HOMEPAGE_OF]-(a:Person),
        (a:Person)-[:AUTHOR_OF]-(p:Publication)-[:ABOUT]->(k:Keyword{keyword:"data mining"})
    WITH u, COUNT(DISTINCT p) AS pcount
    ORDER BY pcount DESC
    RETURN u, pcount
    LIMIT 5
}
WITH u, pcount
MATCH (u:University)-[:AFFILIATED_TO]-(w:Website)-[:HOMEPAGE_OF]-(a:Person),
      (a:Person)-[:AUTHOR_OF]-(p:Publication)-[:ABOUT]->(k:Keyword{keyword:"data mining"})
RETURN u, w, a, p, k, pcount
```

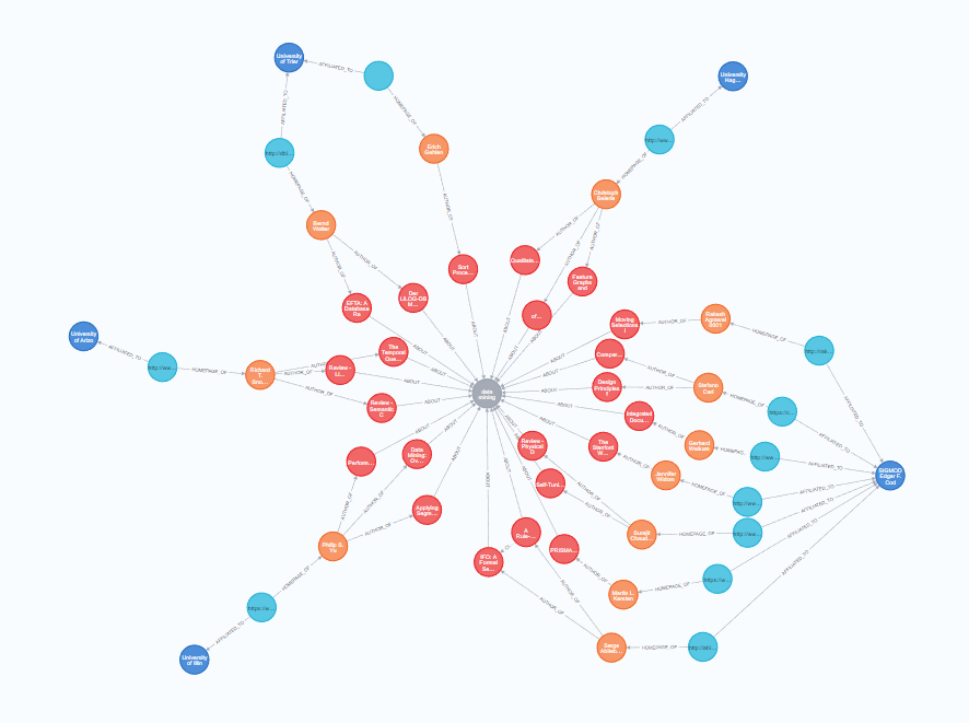

Figure 6.5: Top university per number of publications about a specific topic.

6.1.4. Best PROFESSORS according to number of citations at a given university

The following query can be used to find the authors affiliated with a specific university (e.g.'Politecnico di Milano') ordering them by the average number of citations across their publications.

```
MATCH (u:University {name:"Polytechnic University of Milan, Italy"})<-[:AFFILIATED_TO]-(w:Website),
      (w:Website)-[:HOMEPAGE_OF]->(p:Person)-[]-(a:Publication)
WHERE a.citations IS NOT null
WITH p, avg(a.citations) AS avgcit
ORDER BY avgcit DESC
RETURN p, avgcit
```

"p"	"avgcit"
{"name":"Piero Fraternali","orcid":"Piero Fraternali"}	99.0
{"name":"Stefano Ceri","orcid":"Stefano Ceri"}	91.62500000000001
{"name":"Maria Grazia Fugini","orcid":"Maria Grazia Fugini"}	42.0

Figure 6.6: Best professors per number of citations at a given university.

6.1.5. Top PUBLISHER by popularity of their works

The following query can be used to find the top 5 publishers by the average number of citations of their works that were published after 2015.

```
MATCH (p:Publisher)<-[:PUBLISHED_BY]-(a:Publication)-[:PUBLISHED_IN]->(y:Year)
WHERE y.year >= 2015 AND a.citations IS NOT null
WITH p, avg(a.citations) AS avgcit
ORDER BY avgcit DESC
RETURN p.name, avgcit
LIMIT 5
```

"p"	"avgcit"
{"name":"De Gruyter"}	142.0
{"name":"Springer Fachmedien Wiesbaden"}	128.0
{"name":"epubli"}	126.0
{"name":"Springer Vieweg Wiesbaden"}	124.0
{"name":"Cuvillier Verlag"}	104.0

Figure 6.7: Top publisher by popularity of their works.

6.1.6. Most attended PROCEEDINGS (conference)

The following query can be used to find the top 5 proceedings held in the last 10 years with the highest number of authors who have contributed with an inproceedings.

```
MATCH (p:Proceedings)<-[c:CROSSREF]-(i:Inproceedings)<-[:AUTHOR_OF]-(a:Person),
      (y:Year)<-[h:HELD_IN]-(p:Proceedings)
WHERE y.year >= 2012
WITH y, p, COUNT(DISTINCT a) AS author_count
ORDER BY author_count DESC
RETURN y, p, author_count
LIMIT 5
```

"y"	"p.title"	"author_count"
{"year":2017}	"2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Jeju Island, South Korea, July 11-15, 2017"	1521
{"year":2019}	"Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI 2019, Glasgow, Scotland, UK, May 04-09, 2019"	952
{"year":2013}	"Interaction Design and Children 2013, IDC '13, New York, NY, USA - June 24 - 27, 2013"	159
{"year":2016}	"11th International Conference on Industrial and Information Systems, ICIIS 2016, Roorkee, India, December 3-4, 2016"	151
{"year":2020}	"26th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2020, Hong Kong, December 2-4, 2020"	127

Figure 6.8: Most attended proceedings.

6.1.7. Top JOURNALS

Regarding Journals we can have assorted selection criteria.
For instance we could be interested in finding the top 5 journals whose articles deal with the most varied topics (top Journals by heterogeneity of topics).

```
MATCH (j:Journal)<-[:REFERS_TO]-(v:Volume)<-[:COLLECTED_IN]-(a:Article),
      (a:Article)-[:ABOUT]->(k:Keyword)
WITH j, COUNT(DISTINCT k) AS keywordcount
ORDER BY keywordcount DESC
RETURN j, keywordcount
LIMIT 5
```

"j"	"keywordcount"
{"name":"ACM SIGMOD Digit. Rev."}	45
{"name":"IEEE Data Eng. Bull."}	45
{"name":"SIGMOD Rec."}	45
{"name":"ACM Trans. Database Syst."}	45
{"name":"VLDB J."}	45

Figure 6.9: Top Journals by heterogeneity of topics.

Another idea might be to order journals by the average number of citations of articles contained in it.

```
MATCH (j:Journal)<-[:REFERS_TO]-(v:Volume)<-[:COLLECTED_IN]-(a:Article)
WHERE a.citations IS NOT null
WITH j, avg(a.citations) AS avgcit
ORDER BY avgcit DESC
RETURN j, avgcit
LIMIT 5
```

"j"	"avgcit"
{"name":"Technical Report"}	143.0
{"name":"LILOG-Memo"}	96.0
{"name":"Commun. ACM"}	88.52941176470588
{"name":"LILOG-Report"}	83.43333333333334
{"name":"IEEE Data Eng. Bull."}	76.59550561797755

Figure 6.10: Top Journals by the average number of citations.

6.1.8. Top SERIES of book per topic

The following query can be used to find the series that contains the largest number of books about a specific topic (e.g. 'data mining') published after a reference year (e.g. 2018).

```
CALL{
    MATCH (s:Series)-[:CONTAINS]->(b:Book)-[:ABOUT]->(k:Keyword {keyword: "data mining"}),
        (b:Book)-[:PUBLISHED_IN]->(y:Year)
    WHERE y.year >= 2018
    WITH s, COUNT(*) AS bookcount
    ORDER BY bookcount DESC
    RETURN s, bookcount
    LIMIT 5
}
WITH s, bookcount
MATCH (s:Series)-[:CONTAINS]->(b:Book)-[:ABOUT]->(k:Keyword {keyword: "data mining"}),
      (b:Book)-[:PUBLISHED_IN]->(y:Year)
WHERE y.year >= 2018
RETURN s, b, k
```

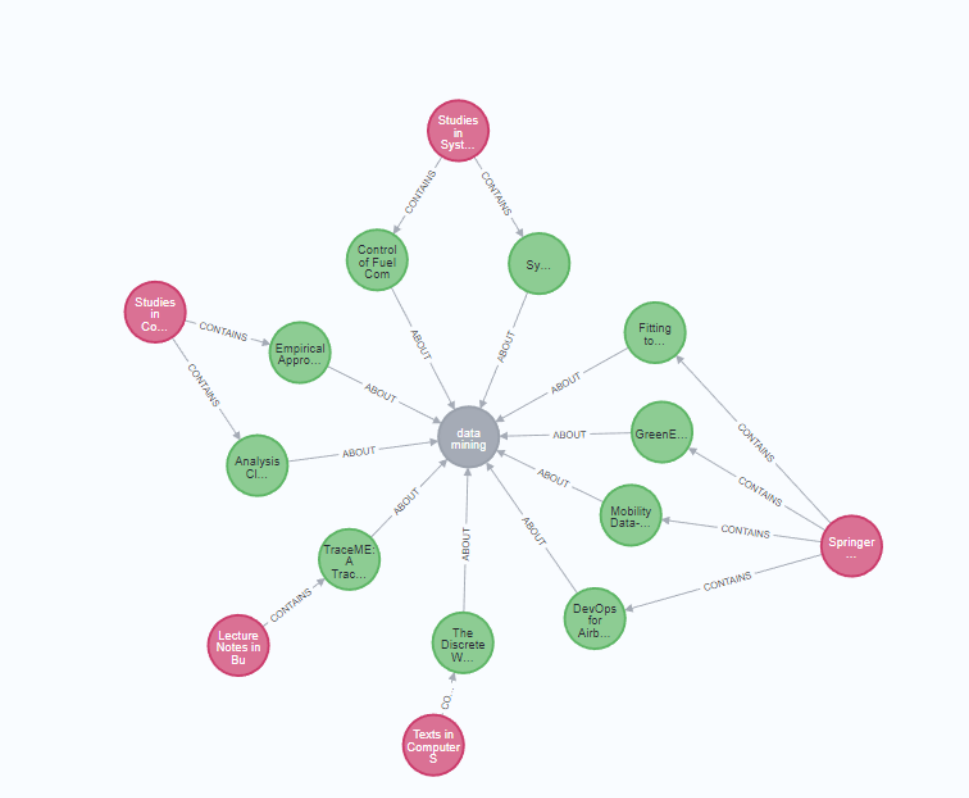

Figure 6.11: Top series of book per topic.

6.1.9. Top PUBLICATIONS

Regarding publications , we have a vast variety of queries that we can make because of its rich representation in the database. In this section we show two relevant examples. For instance we could be interested in finding publications about a specific topic (e.g. 'data mining') written by any professor associated with a particular university(e.g. 'University of Arizona, Tucson, USA'), ordered by the number of citations.

```
CALL{
    MATCH (u:University)-[:AFFILIATED_TO]-(w:Website)-[:HOMEPAGE_OF]-(a:Person),
          (a:Person)-[:AUTHOR_OF]-(p:Publication)-[:ABOUT]->(k:Keyword{keyword:"data mining"})
    WHERE p.citations IS NOT NULL AND u.name = 'University of Arizona, Tucson, USA'
    WITH p
    ORDER BY p.citations DESC
    RETURN p, p.citations AS pcit
    Limit 5
}
WITH p, pcit
MATCH (u:University)-[:AFFILIATED_TO]-(w:Website)-[:HOMEPAGE_OF]-(a:Person),
      (a:Person)-[:AUTHOR_OF]-(p:Publication)-[:ABOUT]->(k:Keyword{keyword:"data mining"})
WHERE p.citations IS NOT NULL AND u.name = 'University of Arizona, Tucson, USA'
RETURN u, w, a, p, k, pcit
```

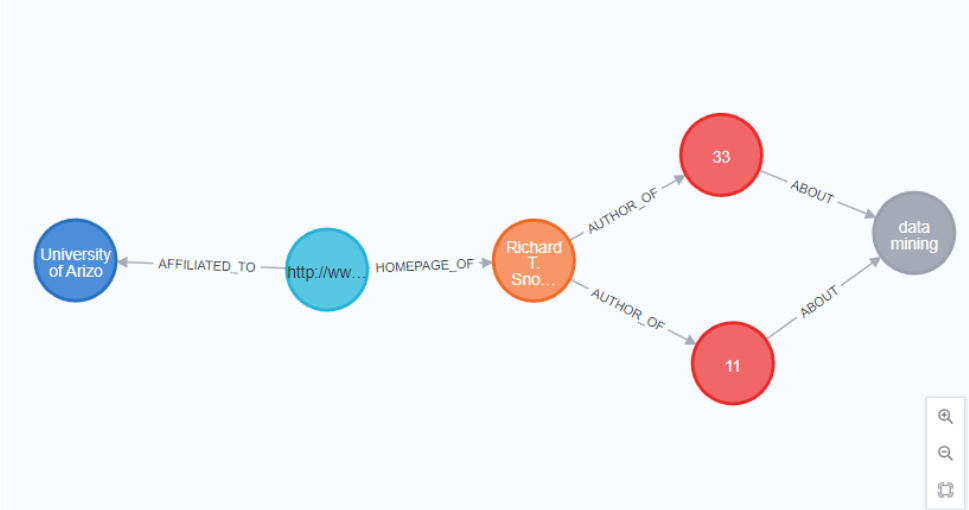

Figure 6.12: Publications per topic.

Another idea might be to find publications about a certain topic (e.g. 'data mining'), written after a reference year (e.g. 2010), ordered by the number of pages contained in it.

```
CALL{
    MATCH (k:Keyword {keyword:"data mining"})<-[:ABOUT]-(p:Publication)-[*1..2]->(y:Year)
    WHERE y.year >=2010
    WITH p
    ORDER BY p.pages ASC
    RETURN p, p.pages AS ppages
    LIMIT 10
}
WITH p, ppages
MATCH (a:Person)-[:AUTHOR_OF]-(p:Publication),
      (k:Keyword {keyword:"data mining"})<-[:ABOUT]-(p:Publication)-[*1..2]->(y:Year)
WHERE y.year >=2010
RETURN a,p,k, ppages
```

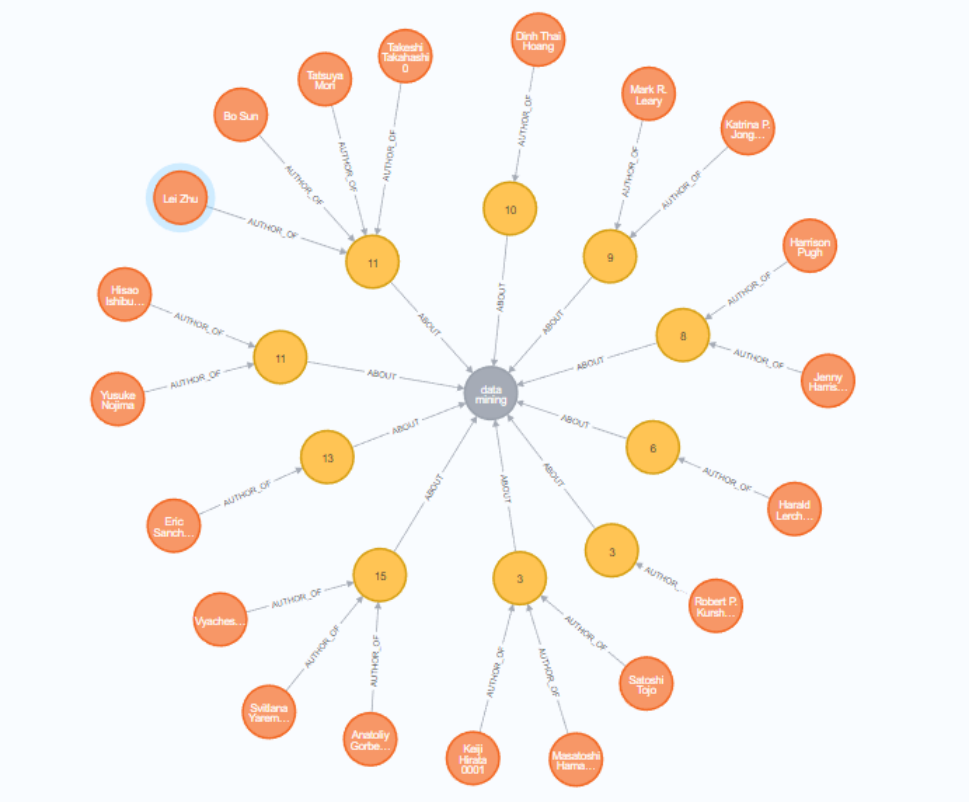

Figure 6.13: Publications ordered by number of pages.

6.1.10. Top KEYWORDS

As mentioned in the publications section, we have unlimited possibilities to customize the queries.
For instance we could be interested in finding the top 10 keywords used together with data mining .

```
MATCH (k:Keyword)<-[:ABOUT]-(p:Publication)-[:ABOUT]->(m:Keyword {keyword: "data mining"})
WITH k, COUNT(*) AS count
ORDER BY count DESC
RETURN k LIMIT 10
```

"k"	"count"
{"keyword":"minimum path"}	42
{"keyword":"dijkstra"}	41
{"keyword":"computer science engineering"}	41
{"keyword":"software"}	39
{"keyword":"c++"}	38
{"keyword":"c89"}	38
{"keyword":"A*"}	37
{"keyword":"interfaces"}	37
{"keyword":"java"}	37
{"keyword":"machine learning"}	35

Figure 6.14: Top keywords used together with data mining.

Another idea might be to find the top 5 keywords used in phdthesis published under a specific university (e.g. "University of Erlangen-Nuremberg, Germany") after a reference year (e.g 2015).

```
MATCH (u:University)<-[:PRODUCED_BY]-(p:Phdthesis)-[:ABOUT]->(k:Keyword),
(p:Phdthesis)-[:PUBLISHED_IN]->(y:Year)
WHERE y.year > 2015 AND u.name = "University of Erlangen-Nuremberg, Germany"
WITH k, COUNT(*) AS kcount
ORDER BY kcount DESC
RETURN k, kcount
LIMIT 5
```

"k"	"kcount"
{"keyword":"machine learning"}	2
{"keyword":"software"}	1
{"keyword":"computer science"}	1
{"keyword":"q-learning"}	1
{"keyword":"graph search"}	1

Figure 6.15: Top keywords used in phdthesis.

6.1.11. shortestpath

A great feature of Neo4j is the SHORTESTPATH function, which finds the minimum path between two nodes.
For instance we could be interested in finding the human chain between two people through people that have worked together on a publication.

```
MATCH (a:Person{name:"Hector Garcia-Molina"}), (b:Person{name: "Philip S. Yu"}),
      p = SHORTESTPATH((a)-[*]-(b))
WHERE all(r IN relationships(p) WHERE type(r) = 'AUTHOR_OF')
RETURN p
```

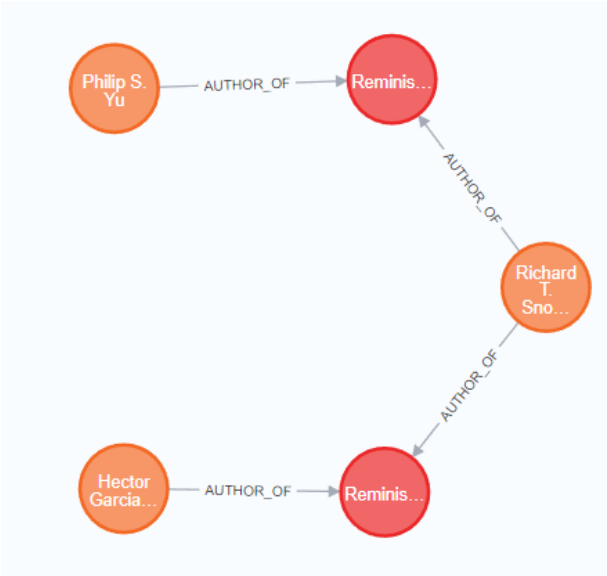

Figure 6.16: Exemple of shortest path.

6.2. Commands

In this section we will analyze various commands which are useful to create/update nodes and relationships. It is impossible to cover all combinations of create/update command, so we propose 5 examples, trying to show the full potential of Neo4j.

6.2.1. Add a KEYWORD to a publications

In general, adding a keyword to a publication is very simple; here we can see a specific example with the binding of the keyword "machine learning" to books in the "Intelligent Systems Reference Library" series from volume 85 to volume 100.

```
MATCH (s:Series)-[r:CONTAINS]-(b:Book)
WHERE r.volume IS NOT NULL AND r.volume >= 85 AND r.volume <= 100
MERGE (k:Keyword {keyword: "machine learning"})
MERGE((b)-[t:ABOUT]->(k))
RETURN s, b, t, k
```

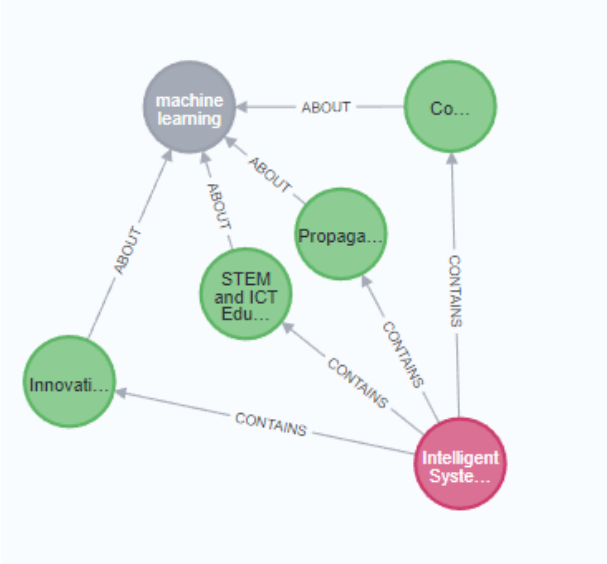

Figure 6.17: Exemple of shortest path.

6.2.2. Add a new PHDTHESIS

In general adding a new publication is very simple; here we can see a specific example with the inclusion of a new PhD Thesis .

```
MERGE(p:Publication:Phdthesis{key: "newphdthesiskey"})
SET p.title = "New Phd Thesis",
p.pages = 54,
p.citations = 0
MERGE(a:Person {orcid: "Barbara Hausmann"})
SET a.name = "Barbara Hausmann"
MERGE((a)-[r:AUTHOR_OF]->(p))
MERGE(y:Year {year: 2022})
MERGE((p)-[:PUBLISHED_IN]->(y))
MERGE(pub:Publisher {name: "Springer"})
MERGE((p)-[:PUBLISHED_BY]->(pub))
RETURN p, a, y, pub
```

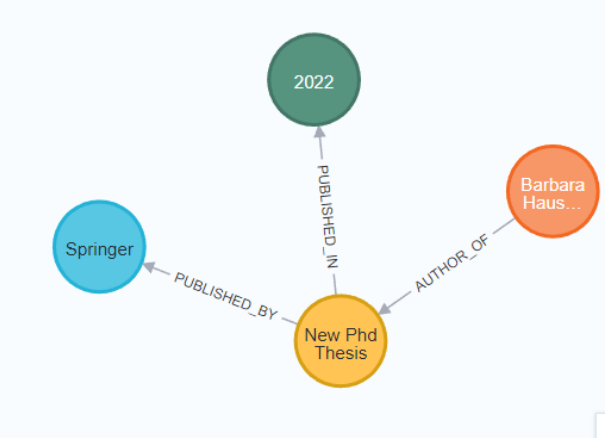

Figure 6.18: Add a new Phd Thesis.

6.2.3. Add a new WEBSITE

The following command can be used to insert a website for Barbara Hausmann, affiliating her to the Technical University of Graz, Austria .

```
MATCH(a:Person {orcid: "Barbara Hausmann"})
MERGE(w:Website {key: "newwebsitekey"})
SET w.title = "Barbara Hausmann's homepage",
    w.url = "www.barbarahausmann.com"
MERGE((w)-[:HOMEPAGE_OF]->(a))
MERGE(u:University {name: "Technical University of Graz, Austria"})
MERGE((w)-[:AFFILIATED_TO]->(u))
RETURN a, w, u
```

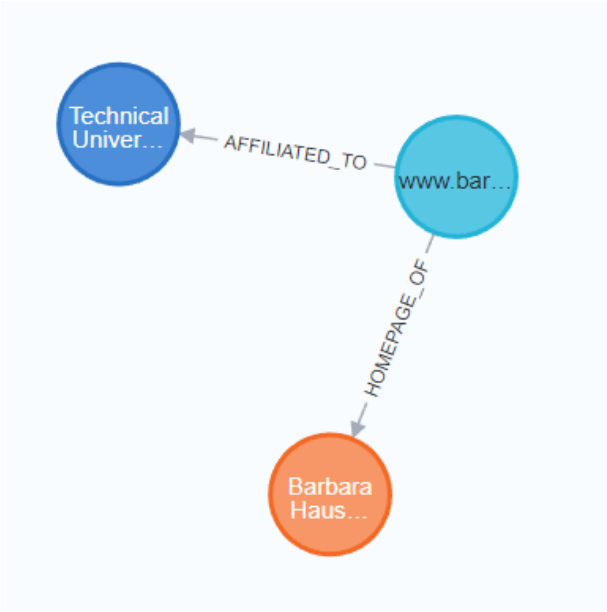

Figure 6.19: Add a new Website.

6.2.4. Add a new inproceedings to the proceedings

The following command can be used to Add a new inproceedings to the proceedings of "26th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2020, Hong Kong, December 2-4, 2020", which has key "conf/icpads/2020", and add keyword "distributed systems" .

```
MATCH(p:Proceedings {key: "conf/icpads/2020"})
MERGE(i:Publication:Inproceedings {key: "newinprockey"})
SET i.title = "New article about Distributed Systems",
    i.pages = 36,
    i.citations = 15
MERGE((i)-[:CROSSREF]->(p))
MERGE(k:Keyword {keyword: "distributed systems"})
MERGE((i)-[:ABOUT]->(k))
RETURN p, i, k
```

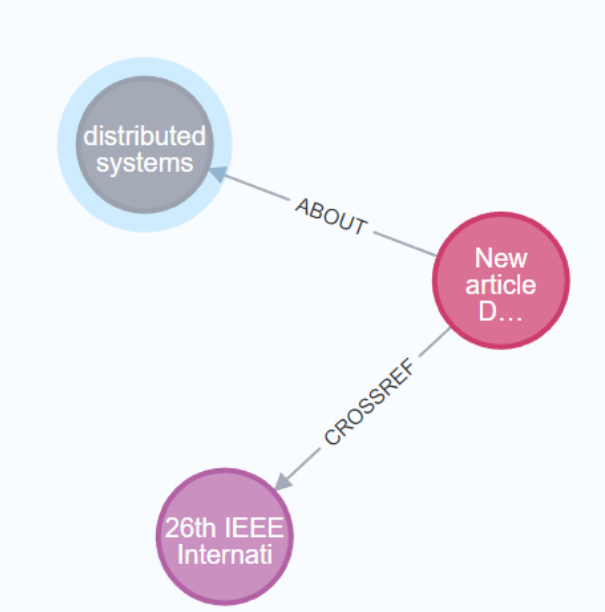

Figure 6.20: Add a new proceedings.

6.2.5. Add a new ARTICLE

The following command can be used to Add a new article to volume 51 of journal "Commun. ACM". .

```
MERGE (a:Publication:Article {key: "newarticlekey"})
SET a.title = "New Article",
    a.pages = 112
MERGE (j:Journal {name: "Commun. ACM"})
MERGE (v:Volume {journal: "Commun. ACM", volume: 51})
WITH j, v, a
MERGE ((a)-[:COLLECTED_IN]->(v)-[:REFERS_TO]->(j))
RETURN j, v, a
```

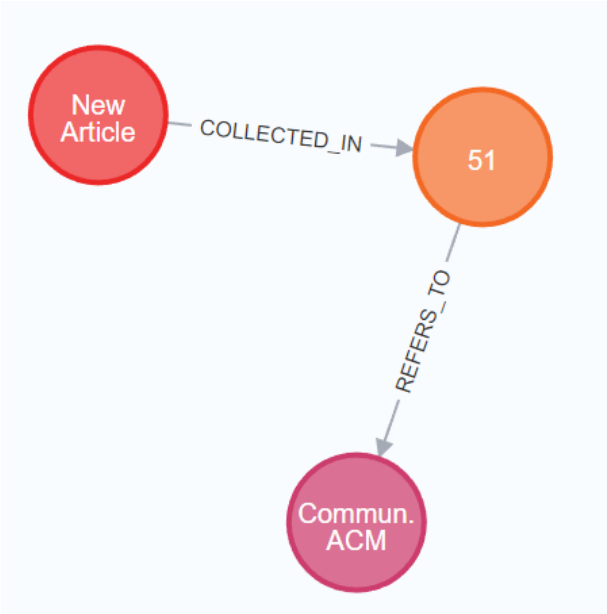

Figure 6.21: Add a new article.

List of Figures