POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Alessio Buda (10675063)**

Leonardo Cesani (10674905)

Fausto Lasca (10661818)

Gabriele Munafò (10654531)

Matteo Paraboschi (10681473)

Group Number: **20**

Academic Year: 2022-2023

Contents

1 | Problem specification

1.1. Description

The aim of the project is to implement a database storage solution for a large-scale dataset recording scientific publications. The database contains various types of scientific publications, including journal articles, books, thesis, conference and workshop papers. Each publication is stored together with important information regarding the publication itself (e.g., its author(s), title, year of publication . . .).

The system should allow users to search for specific publications or to find publications based on some of their characteristics.

1.2. Assumptions

The assumptions considered are the following:

- Every conference edition is a separate entity, since a conference can be held after variable time spans. For journals however a journal entity is needed because it is a stable institution, that publishes the journal edition at a fixed pace.
- The author cannot be affiliated to no organization, and he can be affiliated to at most one organization. That means that there is not a record of all organizations where the author worked during his life.
- The names of organizations, publishers and schools are considered to be unique.

2 | ER model

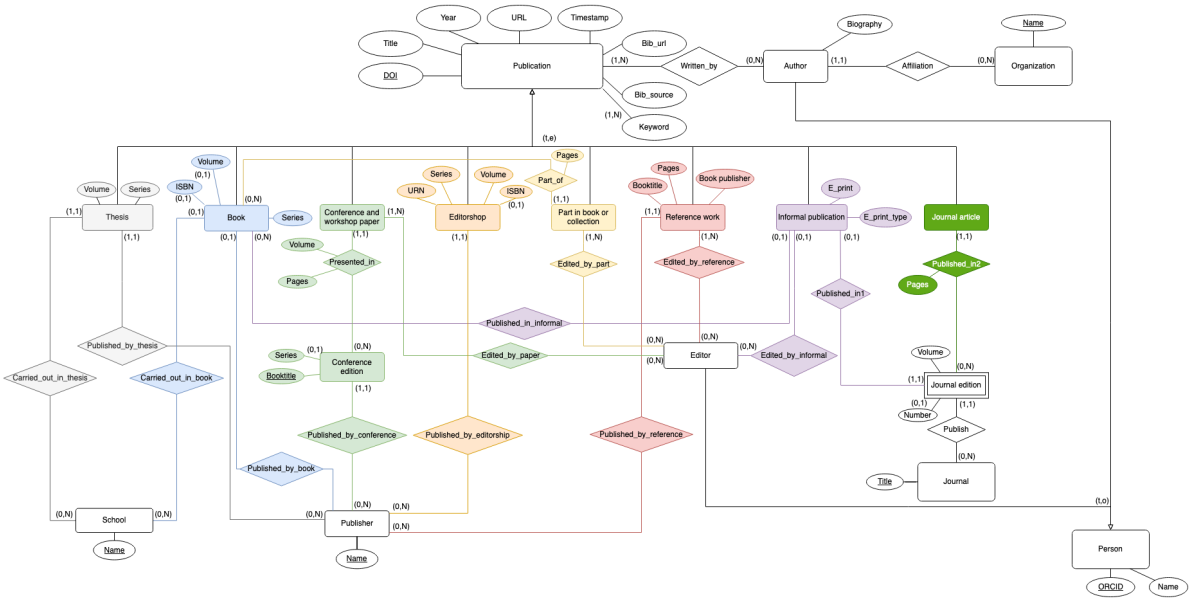

Figure 2.1: ER diagram.

Note: The relationships should be read as follows: a publication is written by at least one author and at most N authors. An author may have written from 0 to N publications.

The ER model presents the following entities:

- **Publication**: represents a generic scientific publication. It is identified by DOI (primary key), the title, the year of publication, the URL of the publication, the timestamp, the Bib URL, the Bib source and potentially multiple keywords associated to the publication. Publications are divided into the subgenres:
  - **Book**: specifically represents a book. Apart from the publication attributes, it is also characterized by an optional ISBN, an optional volume and a series. A book can be carried out in a school (`Carried_out_in_book`) and it is published by a publisher (`Published _by _book`).

  - **Journal article**: specifically represents a journal article. It is published in a journal edition (`Published _in2`), specifically in a certain interval of pages.
  - **Conference and workshop paper**: specifically represents a conference or workshop paper. It is presented in a conference edition (`Presented _in`), specifically in a certain volume and in a certain interval of pages, and it is edited by an editor (`Edited _by _paper`).
  - **Part in book or collection**: specifically represents a part of a book or collection. It is part of a book (`Part _of`), specifically it is a certain interval of pages of that book, and it is edited by an editor (`Edited _by _part`).
  - **Editorship**: specifically represents an editorship. Apart from the publication attributes, it is also characterized by a series, a URN, a volume and an optional ISBN. An editorship is published by a publisher (`Published _by _editorship`).
  - **Reference work**: specifically represents a reference work. Apart from the publication attributes, it is also characterized by a book title which it refers to, a certain interval of pages and the book publisher. A reference work is published by a publisher (`Published _by _reference`) and it is edited by an editor (`Edited _by _reference`).
  - **Informal publication**: specifically represents an informal publication. Apart from the publication attributes, it is also characterized by the E print and the E print type. It can also be published in multiple books (`Published _in _informal`), it can be edited by an editor (`Edited _by _informal`) and it can be published in a journal edition (`Published _in1`).

- **Author**: represents the author who wrote the publication (`Written _by`). It is a type of person. Apart from the person attributes, he is characterized by a biography. He is affiliated to an organization (Affiliation).
- **Organization**: represents the organization to which the author is affiliated (Affiliation). It is identified by a name (primary key), which is considered to be unique.
- **Journal**: represents a journal. It is identified by a title (primary key), which is considered to be unique. Since a journal can publish different editions (Publish) of the journal, this element is represented with a weak entity:
  - **Journal edition**: represents a single journal edition, characterized by a volume and an optional number. In a journal edition, a journal article (`Published`

_in2) or an informal publication (Published _in1) are published.

- **Conference edition**: the conference edition where a conference and workshop paper are presented (Presented _in). It is identified by a book title (primary key) and an optional series. A conference edition is published by a publisher (Published _by _conference).
- **Editor**: represents the editor of a conference and workshop paper (Edited _by _paper), part in book or collection (Edited _by _part), reference work (Edited _by _reference) or informal publication (Edited _by _informal). An editor is a type of person.
- **Publisher**: represents the publisher of a book (Published _by _book), thesis (Published _by _thesis), conference edition (Published _by _conference), editorship (Published _by _editorship) or reference work (Published _by _reference). It is identified by a name (primary key), which is considered to be unique.
- **School**: represent a school where a thesis (Carried _out _in _thesis) or a book (Carried _out _in _book) have been carried out in. It is characterized by a name (primary key) which is considered to be unique
- **Person**: represents a generic person, who can be an editor, an author or both of them (overlapping). It is characterized by a ORCID (primary key) and a name.

3 | Neo4j

3.1. Creation of the database

3.1.1. Data set

The data has been taken from the `dblp` database provided by the organization as a XML file and a DLD file, which formally explains the XML's structure. Since the original `XML` contained a huge number of records, it has been divided into smaller files and reconstructed by taking some pieces in order to have the largest variety of data possible. The resulting subset contains roughly 113000 nodes.
The detailed description of the data set, provided by the `dblp` organization can be found at `https://dblp.org/xml/docu/dblpxml.pdf`.

3.1.2. Data importation in Neo4j

The XML file has been translated into a `CSV` file, an useful format especially for data importation in Neo4j. It has been done by using a dedicated python script built specifically to translate the `dblp` XML database (link: `https://github.com/ThomHurks/dblp-to-csv`). By using the `DLD` file, this script is able to split the data in different `CSV` files, one for each entity. This way, the data gets separated from the headers, making easier to import and create sets of files representing the relationships. The script also assigns a unique ID to each record, in order to recognize the elements in the files and have a feasible relation scheme between entities.
The usage of the python script in order to translate the files format makes the import of data in Neo4j a lot easier to perform, by only generating the following command:

```
neo4j-admin import --database=dblp --delimiter ";" --array-delimiter "|" --id-type INTEGER
--nodes=Publication:Book="output_book_header.csv,output_book.csv"
--nodes=Publication:Thesis="output_phdthesis_header.csv,output_phdthesis.csv"
--nodes=Publication:Thesis="output_mastersthesis_header.csv,output_mastersthesis.csv"
--nodes=Publication:Editorship="output_proceedings_header.csv,output_proceedings.csv"
--nodes=Publication:Article="output_article_header.csv,output_article.csv"
--nodes=Publication:Book_part="output_incollection_header.csv,output_incollection.csv"
--nodes=Publication:Conference_paper="output_inproceedings_header.csv,output_inproceedings.csv"
--nodes=Publication="output_www_header.csv,output_www.csv"
--nodes=Publisher="output_publisher.csv"
--relationships=published_by="output_publisher_published_by.csv"
--nodes=Person:Editor="output_editor.csv"
--relationships=edited_by="output_editor_edited_by.csv"
--nodes=Journal="output_journal.csv"
--relationships=published_in="output_journal_published_in.csv"
--nodes=Person:Author="output_author.csv"
--relationships=written_by="output_author_written_by.csv"
--nodes=School="output_school.csv"
--relationships=submitted_at="output_school_submitted_at.csv"
```

Figure 3.1: Importation command.

To import the data, it is necessary to put the `CSV` files, generated by the python script, in the `bin` in folder of the DBMS where the database will run. After the command has been run, a database called `dblp` is created. When it is opened, the database is already populated with the specified nodes and relationships.

3.1.3. First schema

Below is the representation of the schema loaded in Neo4j:

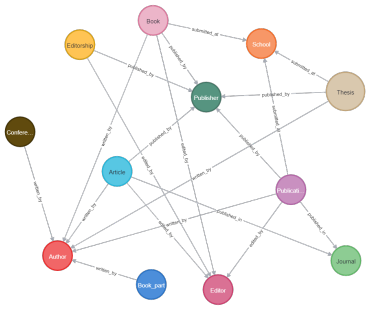

Figure 3.2: First schema.

The data set only contains a list of publications, the nodes that don't represent publication only contain a single attribute, since they had to be extracted from the publications' data. The nodes that represent publications contain all the data present in the data set. It is possible to identify the following nodes:

- **Article**: corresponds to the entity "`journal article`" and "`Informal publication`" in the ER diagram.
- **Author**: corresponds to the entity "`author`" in the ER diagram.
- **Book**: corresponds to the entity "`book`" in the ER diagram.
- **Book part**: corresponds to the entity "`part in a book or collection`" in the ER diagram.
- **Conference paper**: corresponds to the entity "`conference and workshop paper`" in the ER diagram.
- **Editor**: corresponds to the entity "`editor`" in the ER diagram.
- **Editorship**: corresponds to the entity "`editorship`" the ER diagram.
- **Journal**: corresponds to the entity "`journal`" in the ER diagram.
- **Publication**: corresponds to the entity "`publication`" in the ER diagram.
- **Publisher**: corresponds to the entity "`publisher`" in the ER diagram.
- **School**: corresponds to the entity "`school`" in the ER diagram.
- **Thesis**: corresponds to entity "`thesis`" in the ER diagram.

To represent inheritance in the graph, the interested nodes have the label of both the parent and the child. For example, an article is both `Article` and `Publication`. Regarding relationships between nodes, they are shown in the following table:

Relationship	Start node	End node
written by	Article	Author
	Conference_paper	
	Book	
	Book_part	
	Thesis	
	Publication	
Edited_by	Article	Editor
	Editorship	
	Book	
	Publication	
Published_in	Article	Journal
	Publication	
Published_by	Article	Publisher
	Editorship	
	Book	
	Thesis	
	Publication	
Submitted_at	Thesis	School
	Book	
	Publication	

Table 3.1: Relationships

3.1.4. Differences with the ER diagram

Due to the limitations imposed by the chosen data set, there are some inevitable differences between the ER diagram and the graph implementation. The main difference is the absence of some entities:

1. Organization;
2. Journal edition: to compensate for the absence of this entity, the relationships that were connected to it are now instead routed directly to the Journal entity;
3. Conference edition: the same approach used to the Journal edition also applies

here.

4. `Reference work`;
5. `Journal article` and `Informal publication`: this two entities have been combined in the more general `Article` label;

The differences concerning the attributes have already been discussed int the section 3.1.3.

3.2. Queries

In this section we provide as brief description of the queries and their output by showing some screenshots of the result.

3.2.1. Creation and update queries

Query 1

This query merge the duplicates `Authors` and `Editors` (i.e. nodes with the same name).

```
1 match(a:Author), (e:Editor), (p:Publication)-[]→(e) where a.author = e.editor set a:Author:Editor
  create (p)-[r:edited_by]→(a)
2 match(a:Author), (e:Editor) WHERE a.author = e.editor detach delete e
```

Figure 3.3: Query 1.

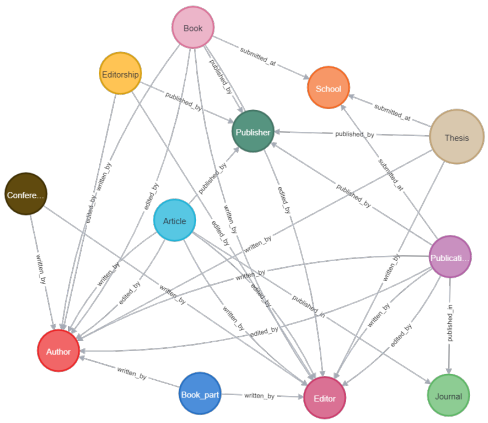

Figure 3.4: Output pf query 2.

Query 2

This query creates the generalization `Person` for `Authors` and `Editors`.

```
1 MATCH(a:Author) SET a:Author:Person Return a
2 MATCH (n:Editor) set n:Editor:Person return n
```

Figure 3.5: Query 2.

3

Figure 3.6: Output pf query 2.

Query 3

This query search for two authors that have written two or more articles together, and creates a relationship `friend with` between them.

```
1 match (a1:Author)<-[r1:written_by]-(p1:Publication)-[r2:written_by]->(a2:Author)<-[r3:written_by]-
  (p2:Publication)-[r4:written_by]->(a1)
2 where p1.title <> p2.title and a1.author <> a2.author and id(a1) > id(a2) merge (a1)-
  [r:friend_with]-(a2) return a1, a2
```

Figure 3.7: Query 3.

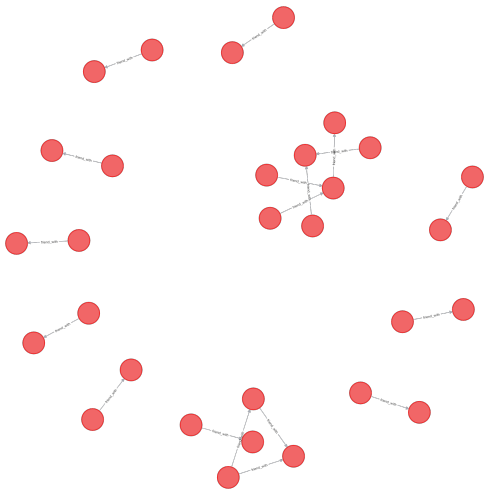

Figure 3.8: Output of query 3.

Query 4

This query creates the relationship `part of` between `Book part` and `Book`, the relationship `part of editorship` between `Conference paper` and `Editorship`.

```
1 match(c:Conference_paper), (e:Editorship)
2 where e.key = c.crossref
3 merge (c)-[:part_of_editorship]→(e)
4 match(bp:Book_part) match (b:Book)
5 where bp.crossref = b.key
6 merge (bp)-[:part_of]→(b)
7 return bp, b
```

Figure 3.9: Query 4.

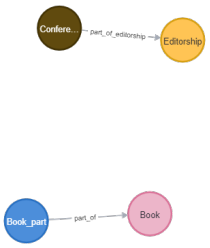

Figure 3.10: Output of query 4.

Query 5

This query creates some `Authors` and an `Article` written by them.

```
create(a1:Author {author: "Leonardo Cesani"})
create(a2:Author {author: "Matteo Paraboschi"})
create(a3:Author {author: "Fausto Lasca"})
create(a4:Author {author: "Alessio Buda"})
create(a5:Author {author: "Gabriele Munafò"})

create(a1)-[:friend_with]→(a2)
create(a1)-[:friend_with]→(a3)
create(a1)-[:friend_with]→(a4)
create(a1)-[:friend_with]→(a5)

create(a2)-[:friend_with]→(a3)
create(a2)-[:friend_with]→(a4)
create(a2)-[:friend_with]→(a5)

create(a3)-[:friend_with]→(a4)
create(a3)-[:friend_with]→(a5)

create(a4)-[:friend_with]→(a5)

create(art:Article {article: 25942,
authors: ["Leonardo Cesani", "Matteo Paraboschi", "Fausto Lasca", "Alessio Buda", "Gabriele Munafò"],
title: "Project Work 2022/2023, System and Methods for Big and Unstructured Data", year: 2022}ù)
create(art)-[:written_by]→(a1)
create(art)-[:written_by]→(a2)
create(art)-[:written_by]→(a3)
create(art)-[:written_by]→(a4)
create(art)-[:written_by]→(a5)
return a1, a2, a3, a4, a5, art
```

Figure 3.11: Query 5.

Figure 3.12: Output of query 5.

3.2.2. Request queries

Query 1

This query finds the shortest path in the graph between two given `Authors`.

```
1 match (a:Author {author: "Javier Gozálvez"}),
2 match(b:Author {author : "Harvey Glickenstein"}),
3 p = shortestPath((a)-[*]-(b))
4 return p
```

Figure 3.13: Query 1.

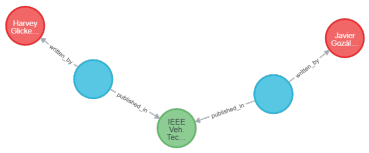

Figure 3.14: Output of query 1.

Query 2

This query finds the `Author` with the most number of publications published in a `Journal`.

```
1 match (p:Person)←[:written_by]-(c:Publication)-[r1:published_in]→(j:Journal)
2 with p, j, count(r1) as in_journal, collect(c) as publication
3 RETURN p, j, publication, in_journal order by in_journal desc limit 1
```

Figure 3.15: Query 2.

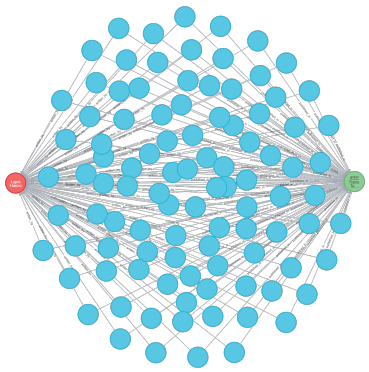

Figure 3.16: Output of query 2.

Query 3

This query extracts the `Publications` of two `Authors` who are colleagues and have the highest sum of `Publications`.

```
match (p1:Person)<-[w1:written_by]-(pub1:Publication)
with p1, count(w1) as n_pub1, collect(pub1) as publication1
where n_pub1 > 30
match (p2:Person)<-[w2:written_by]-(pub2:Publication)
with p1, n_pub1, publication1, p2, count(w2) as n_pub2, collect(pub2) as publication2
where id(p1) > id(p2) and n_pub2 > 30
match (p1)-[:friend_with]-(p2)
return p1, p2, publication1, publication2, n_pub1 + n_pub2 order by n_pub1 + n_pub2 desc limit 1
```

Figure 3.17: Query 3.

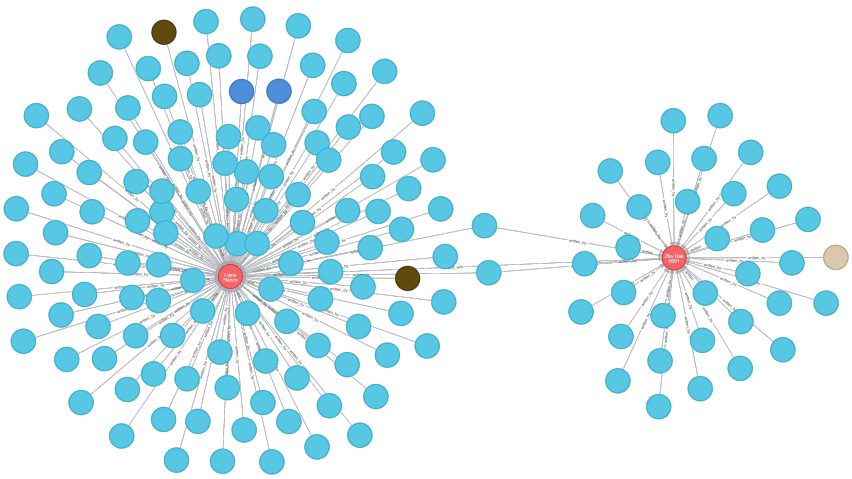

Figure 3.18: Output of query 3.

Query 4

This query finds the school with the most number of `Publications` authored by students whose name starts with the letter 'X'.

```
match (per:Person)<-[]-(pub:Publication)-[s:submitted_at]->(school:School)
where per.author starts with 'X'
with school, count(s) as submissions, collect(pub) as publications, collect(per) as people
return school, people, publications, submissions order by submissions desc limit 1
```

Figure 3.19: Query 4.

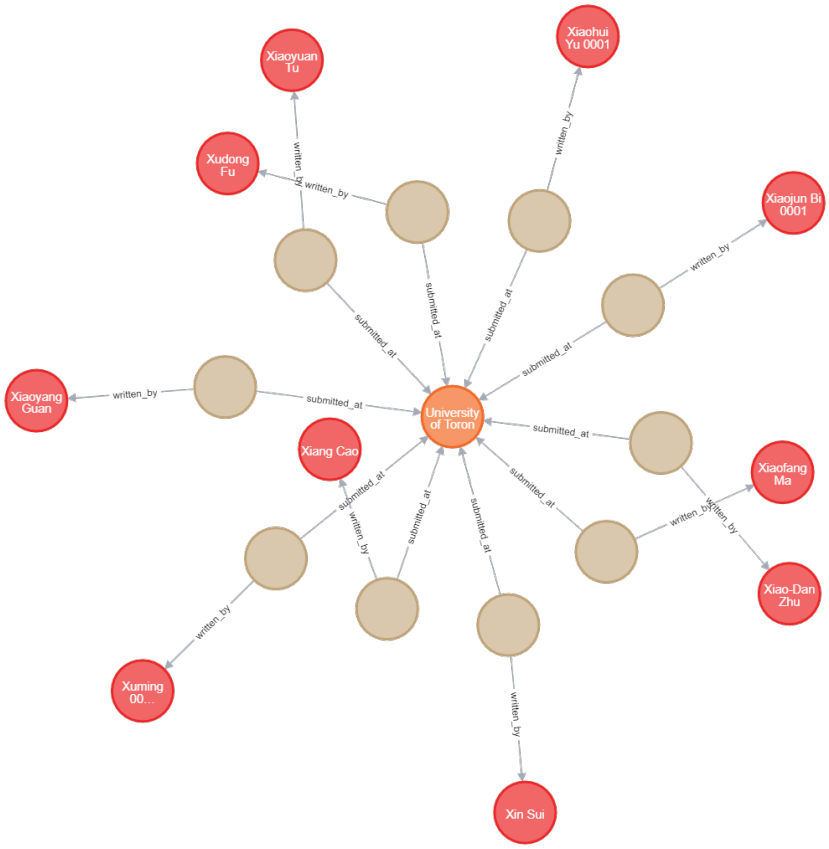

Figure 3.20: Output of query 4.

Query 5

This query extract all the 2-level colleagueship of the `Author` named "Lajos Hanzo".

```
match(p1:Person {author : "Lajos Hanzo"})-[:friend_with*..2]-(p2:Person)
where id(p1) <> id(p2)
return p1, p2
```

Figure 3.21: Query 5.

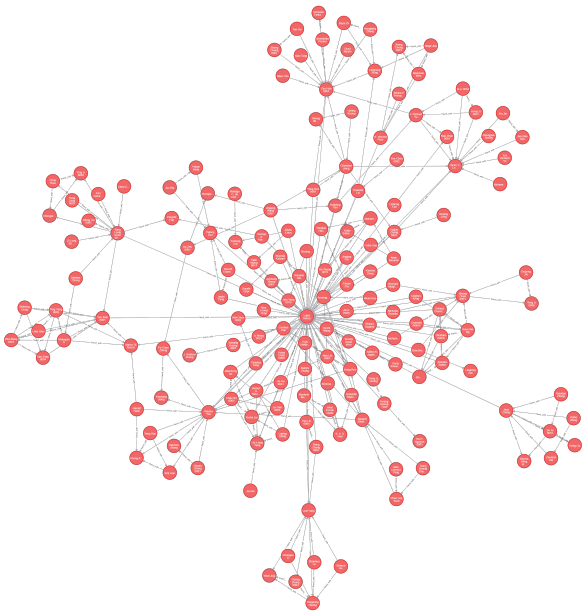

Figure 3.22: Output of query 5.

Query 6

This query finds the book, its chapters and the `Authors`, that has been written by the most number of people.

```
1 match (p:Person)←[r:written_by]-(bp:Book_part)-[r1:part_of]→(b:Book)
2 with b, count(distinct(p)) as authors,  collect(bp) as book_part, collect(p) as people
3 return b, book_part, people, authors order by authors desc limit 1
```

Figure 3.23: Query 6.

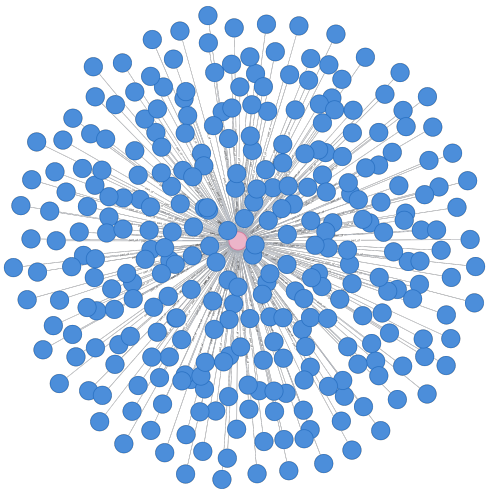

Figure 3.24: Output of query 6.

Query 7

This query finds the `Authors`, the `Conference paper` and the `Editorship`, have published a `Conference paper` in 2006.

```
match (p:Person)←[:written_by]-(c:Conference_paper)-[:part_of_editorship]→(e:Editorship)
where e.year = 2006
return p, c, e
```

Figure 3.25: Query 7.

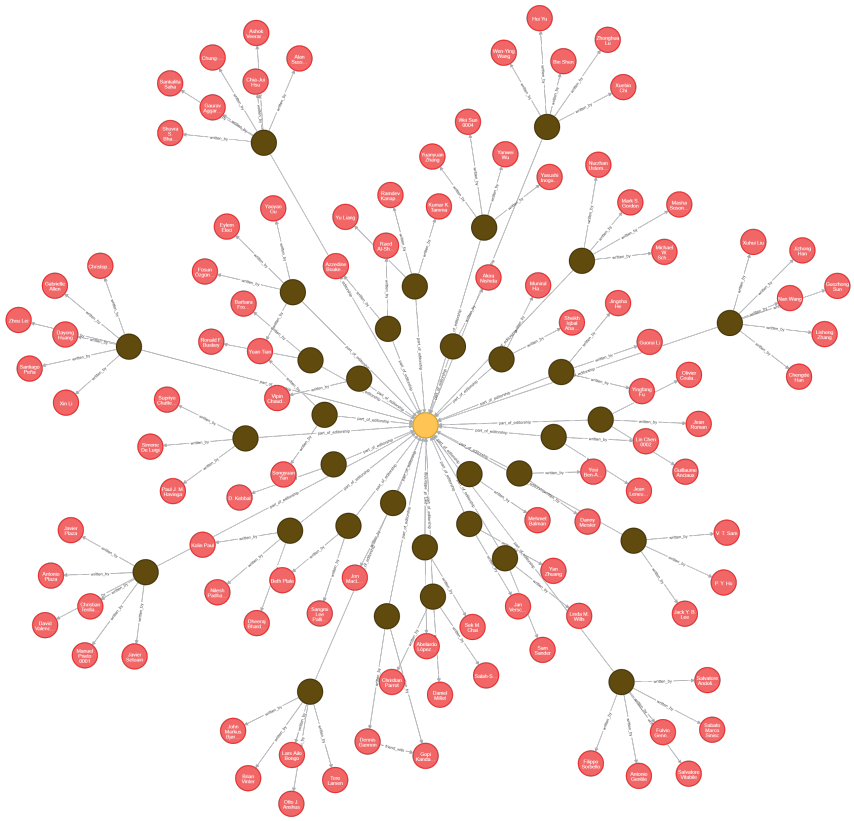

Figure 3.26: Output of query 7.

Query 8

This query search for the `Author` with the most `Publications` published in a `Journal` in a single year.

```
1 MATCH (a:Author)←[:written_by]-(p:Publication)-[:published_in]→(Journal)
2 return a.author,  p.year, count(p) as c  order by c desc limit 1
```

	a.author	p.year	c
1	"Lajos Hanzo"	2021	13

Figure 3.27: Query 8.

Query 9

This query returns the journal articles published by `Authors` that submitted a `Thesis` at the Polytechnic University of Milan.

```
1 match (s:School {school:"Polytechnic University of Milan, Italy"})←[:submitted_at]-(t:Thesis)-
  [:written_by]→(au:Author)←[:written_by]-(ar:Article)-[:published_in]→(j:Journal)
2 return s, t, au, ar
```

Figure 3.28: Query 9.

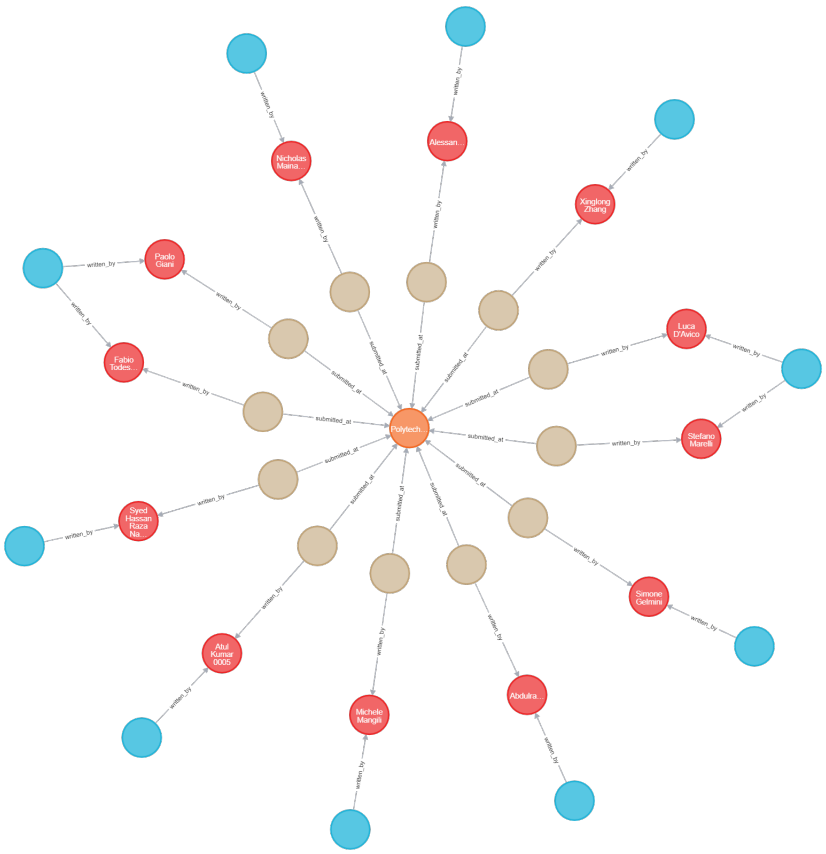

Figure 3.29: Output of query 9.

Query 10

This query returns a list of the Journals where `Articles`, written by `Authors` that submitted their thesis at the Polytechnic University of Milan, have been published. The list is ordered by the number of articles that has been published on each journal. `Journals` where

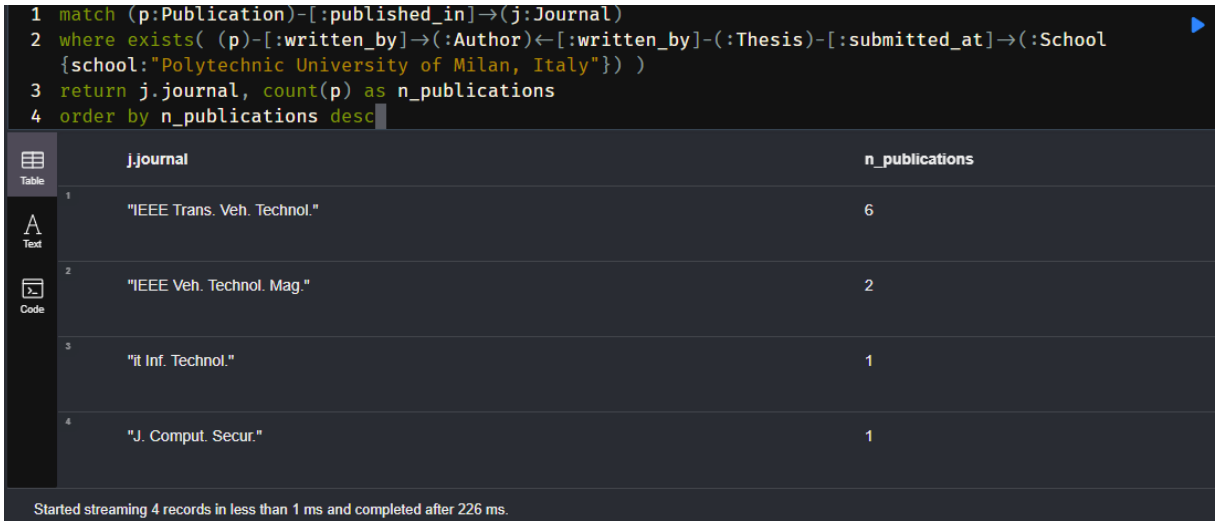

Figure 3.30: Query 10.