POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Simona Malegori**
Nicole Perrotta
Michele Simeone
Alberto Pirillo
Simone Tognocchi

Group Number: **38**

Academic Year: 2022-2023

Contents

Contents **i**

1 Problem description **1**

2 Hypothesis **3**

3 Data **5**

3.1 ER Diagram 5

3.2 Data set description 7

3.3 Data Pre-Processing 8

4 Neo4j **11**

4.1 Data Upload 11

4.2 Graph Diagram 13

4.3 Queries 16

4.4 Creation/Update Commands 26

5 References **29**

1 | Problem description

This project aims at building an Information System that manages a data set containing different type of scientific articles that can be used for: clustering with network and side information, studying influence in the citation network, finding the most influential papers and topics, modeling analysis.

The project is divided in the following steps.
At first it was made an ER Diagram that generalizes all the information gathered from different already existing data sets, then the most complete data set was chosen.
Afterwards the data set was pre-processed, transforming it from a JSON to a CSV format and then it was reduced in size.
After that it was uploaded on Neo4j and all the nodes, the relationships and the properties were edited to build the Graph Diagram.
At the end of the project 10 queries and 6 creation/update commands were initiated with a different level of complexity that was checked within the performance time.

2 | Hypothesis

In order to model the database we made some assumptions:

- authors can have zero, one or more papers associated, assuming that authors are inserted in the database before their paper/s is/are;
- authors can have zero, one or more affiliated organizations, assuming that there can be authors that didn't provide their organization;
- papers have at least one author;
- papers have at least one field of study;
- not all papers have keywords associated, assuming that they may not have been provided;
- papers can reference and be referenced by other papers;
- each paper has a venue, that is the place where it has been published/presented;
- a venue can host more than one paper;
- venues are of 4 types: Journal, Conference, Book and Patent;
- the volume n of a paper is the n-*th* published collection;
- the issue m of a paper is the m-*th* part of the volume in which it is published.

3 | Data

3.1. ER Diagram

The ER Diagram of the chosen model is characterized by the following entities with the respective attributes:

- **Paper** is a scientific article that is associated with the following attributes: *id*, *title*, *date* that corresponds to the publication date, *doi* that is the Digital Object Identifier, *volume* that corresponds to the n-th published collection, *issue* that corresponds to the m-th part of the volume, *language*, *issn* that is an identification code associated with the title of the publication, *isbn* that is a code that identifies printed or digital papers and it is used as inventory-tracking device, *n_citation* that is the number of citations, *page_start* and *page_end* that are the starting and the ending point of the collection from which the paper was extracted, *pdf_url* that is the source from which to recover the paper, *abstract* that is the summary of the paper, *publisher* and *external_url* that corresponds to the sitography of the paper;
- **Author** with the attributes: *id*, *name*, *surname*, *email*, *orcid* that is a unique and persistent identification number and *organization*;
- **Keyword** that represents a word that allows to define immediately the topic within the paper, its attributes are: *id* and *name*;
- **FoS** that corresponds to the field of studies with the attributes: *id*, *name*, *w* that is the weight of the fields of study;
- **Venue** that is the collection from which the paper was extracted, with the attributes: *id* and *name*. The venue of the paper can be of different types, in fact there is a total and exclusive generalization of the entity Venue that can be a:
  - **Journal** that has also the attribute *addressee* that is the type of audience of the paper;

  - **Book** that has also the attributes *category* and *edition*;
  - **Conference** that has also the attributes *type* that can be physical and online, and *location* that has cardinality one only if the type is physical and it represents the place in which the conference takes place;
  - **Patent** that has also the attributes *type* and *expiration*.

In the ER Diagram there are the following relationships with the respective cardinalities:

- **Writing** between the entities Paper and Author. The relationship means that a Paper can be written by at least 1 to a maximum of N authors and that an Author can write from 0 to N papers.
- **Containing** between the entities Paper and Keyword. The relationship means that a Paper can contain from 0 to N keywords and that a Keyword is contained into at least 1 to a maximum of N papers.
- **Dealing** between the entities Paper and FoS. The relationship means that a Paper can deal with at least 1 to a maximum of N field of studies and that a FoS can be dealt from 0 to N papers.
- **In** between the entities Paper and Venue. The relationship means that a Paper is extracted from exactly 1 venue and that a venue can be the collection in which at least one paper is contained.
- **Referencing** that is a relationship on the same entity of the Paper, in fact it contains the roles referencer and referenced. The relationship means that a Paper can or not reference other Papers and can be referenced or not from other Papers.

After all, in the ER Diagram there is an external constraint on the attributes of the entity Conference. In fact, the attribute *location* must have a cardinality of (1,1) if the type is *physical*.

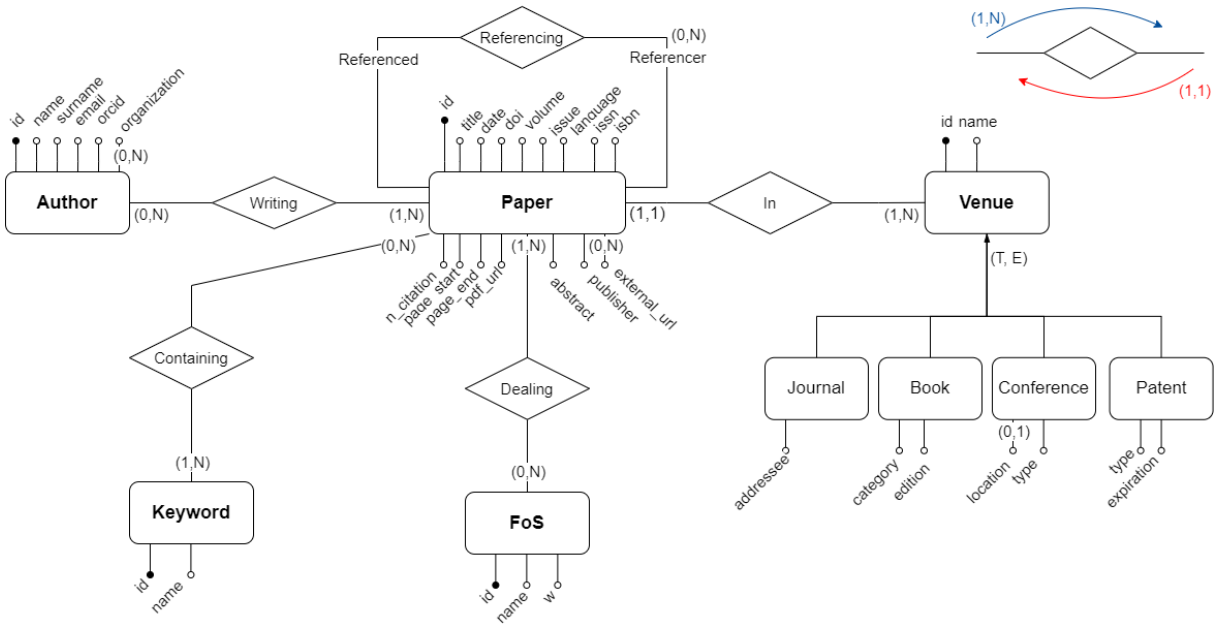


Figure 3.1: ER Diagram

3.2. Data set description

The data set that was chosen for the goal of the project contains 8335 papers, 730 venues, 27576 authors and 25005 field of studies. It is a reduced version of the more complex model on which the ER Diagram is based. In fact, it doesn't contain the entities Keyword and the sub-entities Journal, Book, Conference and Patent that, instead, are transformed into an attribute of the entity Paper that is called doc_type. All the respective attributes of these entities are eliminated. Moreover, the attribute *date* of the Paper, that is generalized in the ER Diagram, becomes the year. Furthermore, in the reduced version of the data set, the relationship Referencing becomes a list of string that are the references of the Paper. Lastly, the chosen data set contains just the more significant attributes. Summarizing, the chosen data set is composed by the following entities and attributes:

- **Paper** with the attributes: id (integer type), title (string type), year (integer type), n_citation (integer type), page_start (integer type), page_end (integer type), doc_type (string type), publisher (string type), volume (integer type), issue (integer type), doi (string type), references (list of string type), abstract (string type);
- **Author** with the attributes: paper_id (integer type), author_name (string type), author_id (integer type), author_org (string type);
- **FoS** with the attributes: paper_id (integer type), fos_name (string type), fos_weight (float type);

- **Venue** with the attributes: paper_id (integer type), venue_name (string type), venue_id (integer type).

3.3. Data Pre-Processing

The original data set can be downloaded *here*. Given that such data set was unnecessarily large for our purpose, we decided to reduce its size. We could have just cut it at a certain point, however we decided that it was better to work with consistent data, therefore we decided to carefully perform sub-sampling in an intelligent way. Such pre-processing was performed using multiple Python scripts and the Pandas library.

Here we provide a description of all the scripts used, together with the procedure to obtain the final data set starting from the initial one. All of these scripts can be found in the *scripts* folder of the *GitHub repository* of the project.

The notebook **dataset_exploration.ipynb** contains a short description of every operation for explanatory reasons. This notebook processes only a small chunk of the data set. The same operations are performed on the whole data set in the script **dataset_preprocessing.py.**

Here is a summary of the operations performed:

- Removal of samples with Null and NaN values
- Removal of samples with an empty string in a field

The operations above are required to work with consistent data. Notice that we can afford to simply drop the samples that do not respect such conditions since we dispose of a very large data set.

The following operations are not required but were performed to reduce even further the size of the data set, with the objective of keeping only the "most important" samples.

We kept the samples:

- With a number of citations greater than a threshold
- With a reference count greater than a threshold

We converted the *indexed_abstract* field from an inverted index to a string of text, to make it easier to query once inserted into the database. The field was also renamed to *abstract*.

We also processed the data set in order to remove some special characters not supported

by the import function of the database. An example of such characters is "\". This operation is performed in the script **remove_special_characters.py**.

Given that many samples were removed from the data set, we also had to fix up the *references* field to keep only the valid ones. We consider a reference valid when it points to a sample of the data set that was kept. Otherwise, we say that such reference is invalid and we remove it from the data set. This operation is performed in the script **deprecated_references.py**.

Lastly, when importing the data into the database, we realized that it was more practical to split the data set into multiple data sets to speed up the process and to produce cleaner code. This functionality was added in the **dataset_preprocessing.py** script. We split the data set following the structure of ER diagram, ending up with one separate data set for each entity present in the original data set. Thus, we ended up with 4 data sets: **Paper**, **Author**, **Venue** and **Fos**.

To obtain the final data set starting from the downloaded one, run the scripts in this order:

1. **dataset_preprocessing.py**
2. **deprecated_references.py**
3. **remove_special_characters.py**

The input of the first script is the initial data set. Only the paper data set requires to be processed by the second script, then only the paper and the venue data sets require to be processed by the third script. At the end, you will obtain 4 data sets which are identical to the ones that we imported into the database.

4 | Neo4j

4.1. Data Upload

To import the data into Neo4j, there is one last precaution needed: it is necessary to process the Paper data set to make the *references* field compatible with the import function of the database. To perform such action the script **preprocessing.py** can be used. The script is located in the *neo4j* folder of the *GitHub repository*. Now, the data sets can be correctly imported into the Neo4J database. In order to do that, you need to put the four data sets files in the program's import folder.

Thanks to the pre-processing that was performed, the four data sets are saved in the simple CSV format and it is possible to use the LOAD CSV command to easily load all the entities and relationships.

1. Clear the Database:

```
MATCH (x) DETACH DELETE x;
```

2. Load the Papers:

```
LOAD CSV WITH HEADERS FROM "file:///paper_dataset2.csv" AS csvLine
CREATE (p:Paper {id: toInteger(csvLine.id), title: csvLine.title,
   year: toInteger(csvLine.year), doi: csvLine.doi, volume: csvLine
   .volume, issue: csvLine.issue, abstract:csvLine.abstract,
   n_citation:toInteger(csvLine.n_citation), page_start:toInteger(
   csvLine.page_start), page_end:toInteger(csvLine.page_end),
   publisher:csvLine.publisher, doc_type:csvLine.doc_type})
```

3. Load the Authors:

```
LOAD CSV WITH HEADERS FROM "file:///author_dataset.csv" AS csvLine
CREATE (a:Author {id: toInteger(csvLine.author_id), name: csvLine.
   author_name, organization: csvLine.author_org})
```

4. Load the Fos:

```
LOAD CSV WITH HEADERS FROM "file:///fos_dataset_0.csv" AS csvLine
```

```
CREATE (f:Fos {weight: toFloat(csvLine.fos_weight), name: csvLine.
    fos_name})
```

5. Load the Venue:

```
LOAD CSV WITH HEADERS FROM "file:///venue_dataset.csv" AS csvLine
MERGE  (v:Venue {id: toInteger(csvLine.venue_id), name: csvLine.
    venue_name})
```

6. Create the `Writing` relationship between papers and authors:

```
LOAD CSV WITH HEADERS FROM "file:///author_dataset.csv" AS csvLine
MATCH (p:Paper),(a:Author)
WHERE (toInteger(csvLine.paper_id)=p.id) and (toInteger(csvLine.
    author_id) = a.id)
CREATE (p)-[w:writing]->(a)
```

7. Create the `Referring` relationship between papers and papers:

```
LOAD CSV WITH HEADERS FROM "file:///paper_dataset2.csv" AS csvLine
UNWIND split(csvLine.references,':') as ref
MATCH (p1:Paper),(p2:Paper)
WHERE (toInteger(csvLine.id)=p1.id) and (toInteger(ref)=p2.id)
CREATE (p1)-[r:referencing]->(p2)
```

8. Create the `in` relationship between papers and venues:

```
LOAD CSV WITH HEADERS FROM "file:///venue_dataset.csv" AS csvLine
MATCH (p:Paper),(v:Venue)
WHERE (toInteger(csvLine.paper_id)=p.id) and (toInteger(csvLine.
    venue_id) = v.id)
CREATE (p)-[i:in_]->(v)
```

9. Create the `dealing` relationship between papers and FoS:

```
LOAD CSV WITH HEADERS FROM "file:///fos_dataset_0.csv" AS csvLine
MATCH (p:Paper),(f:Fos)
WHERE (toInteger(csvLine.paper_id)=p.id) AND (csvLine.fos_name = f.
    name) AND (toFloat(csvLine.fos_weight) = f.weight)
CREATE (p)-[d:dealing]->(f)
```

4.2. Graph Diagram

Entities:

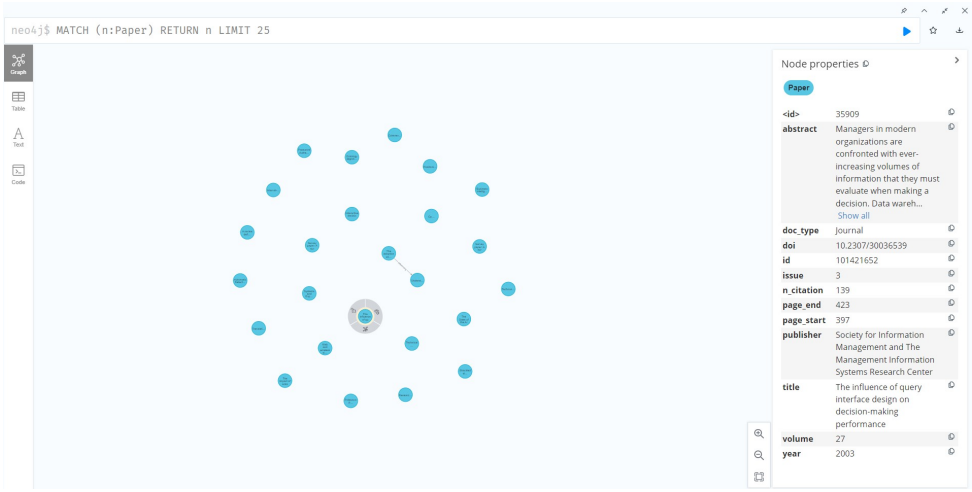


Figure 4.1: Paper

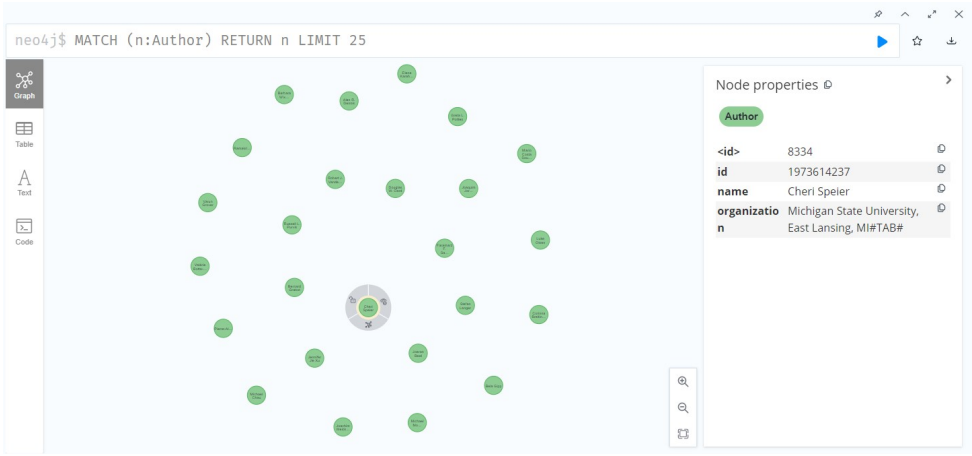


Figure 4.2: Author

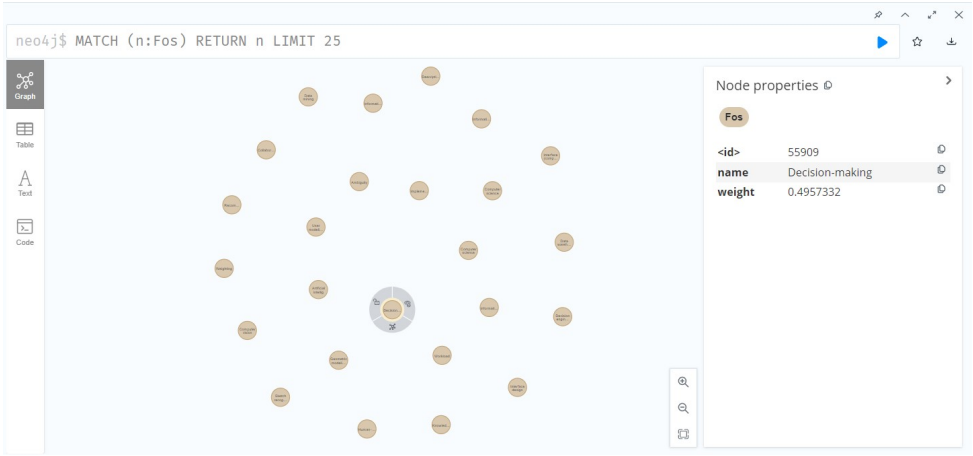


Figure 4.3: FoS

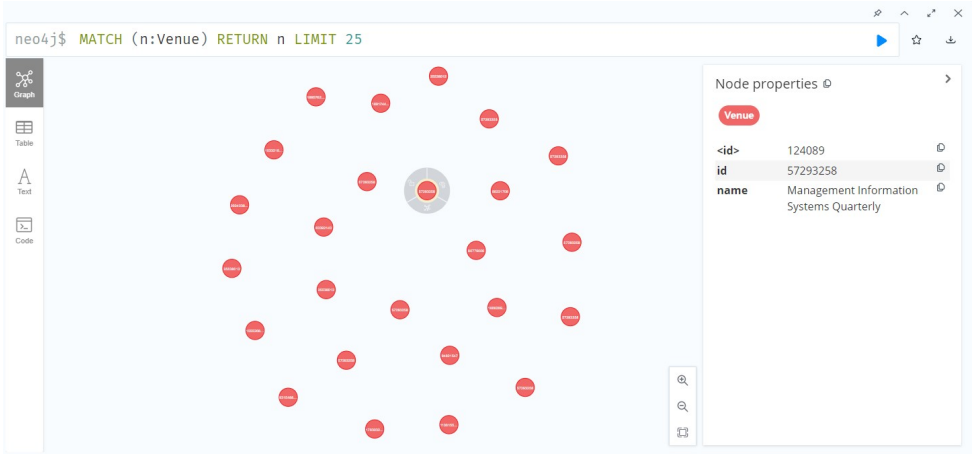


Figure 4.4: Venue

Relationships:

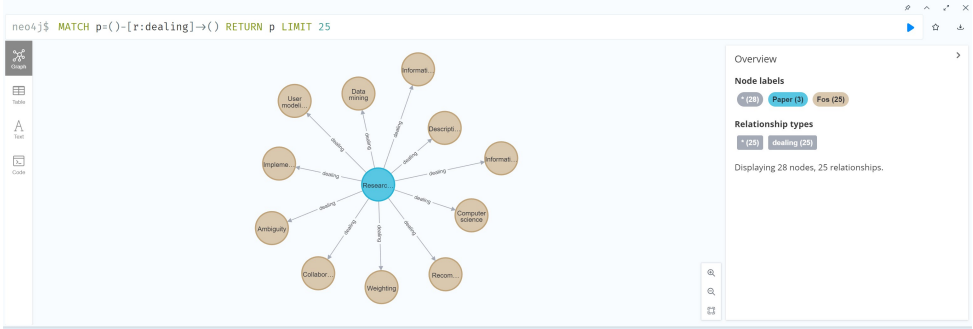


Figure 4.5: Paper-Dealing->FoS

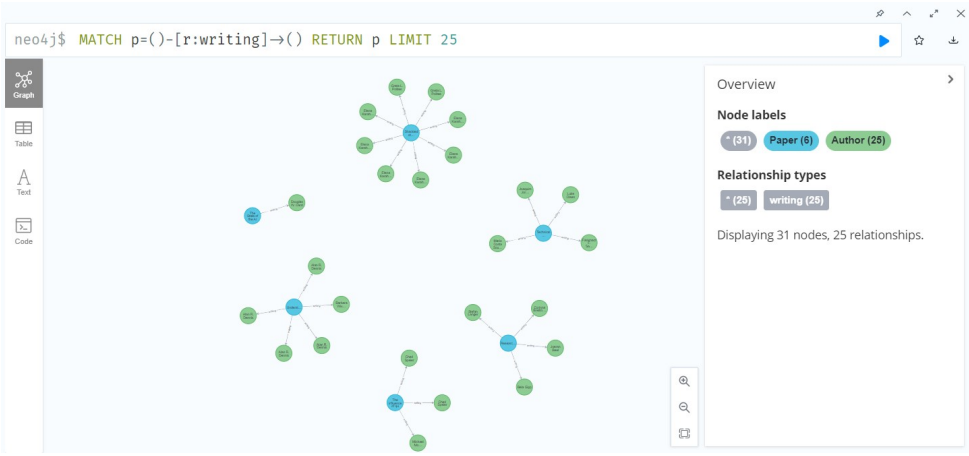


Figure 4.6: Paper-Writing->Author

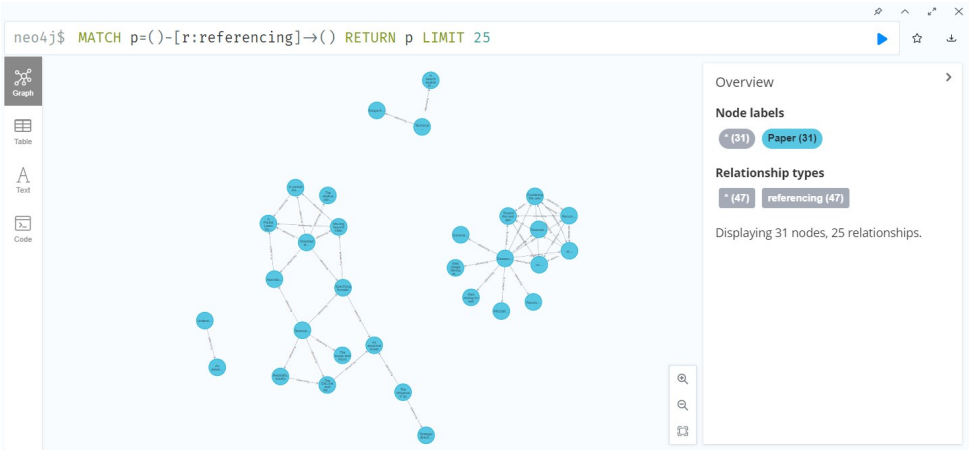


Figure 4.7: Paper-Referencing->Paper

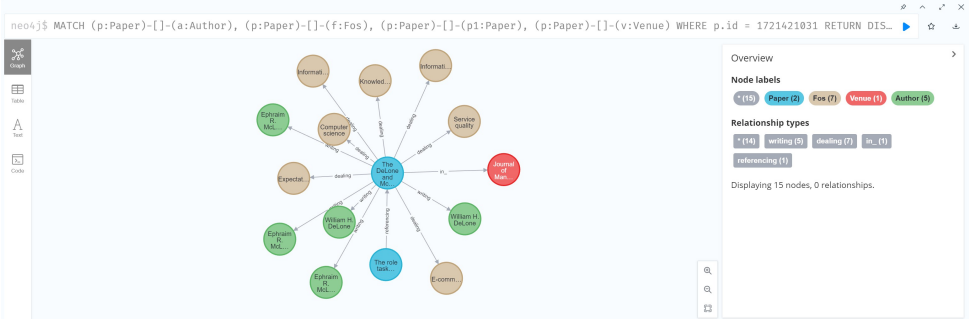
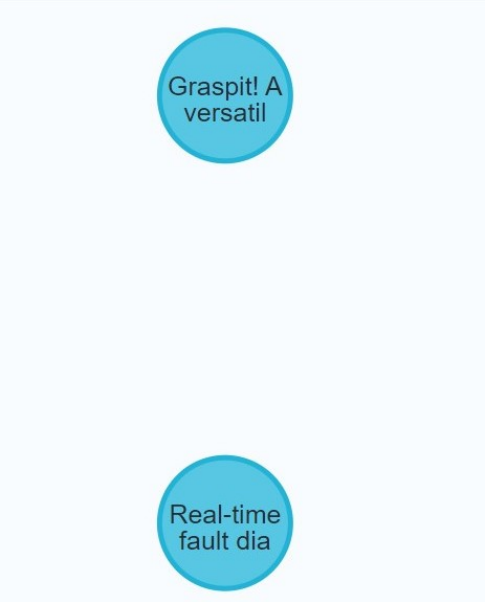


Figure 4.8: All the relationships

4.3. Queries

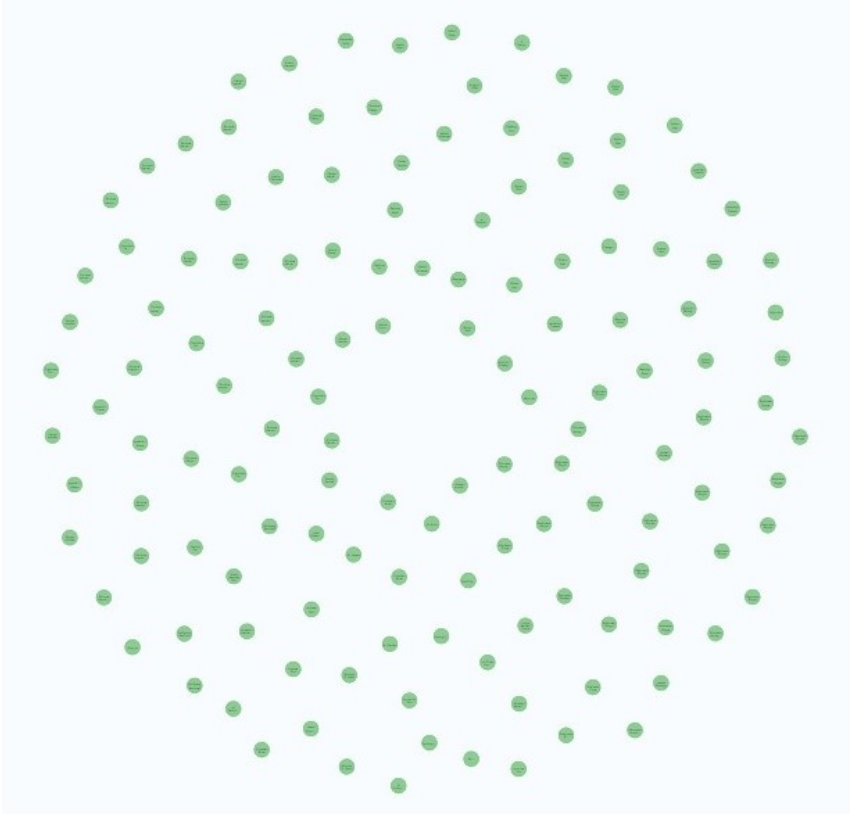
1. **Find papers of a determined venue, written after a certain date (Execution time 454ms)**:

```
MATCH (p: Paper)-[i:in_]->(v: Venue), (p: Paper)-[d:dealing]->(f:
    Fos)
WHERE (p.year > 2000) AND (v.name="IEEE Robotics & Automation
    Magazine")
RETURN p
```

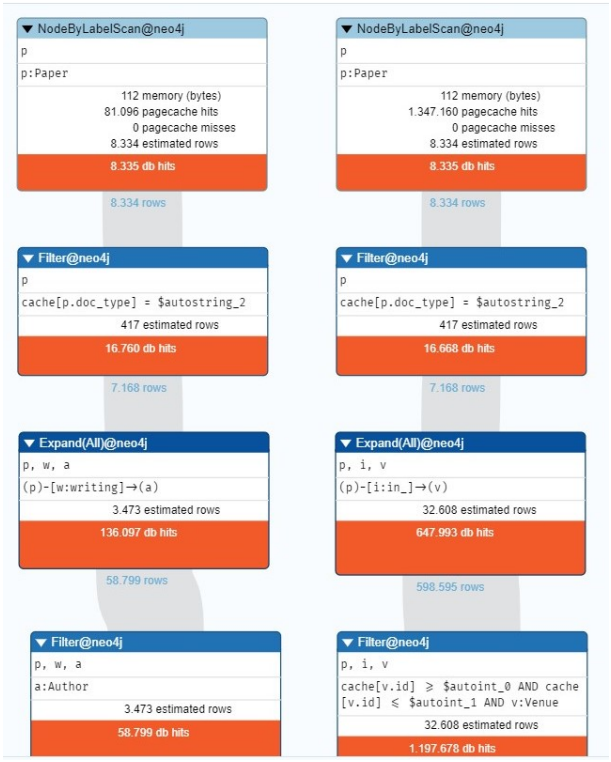


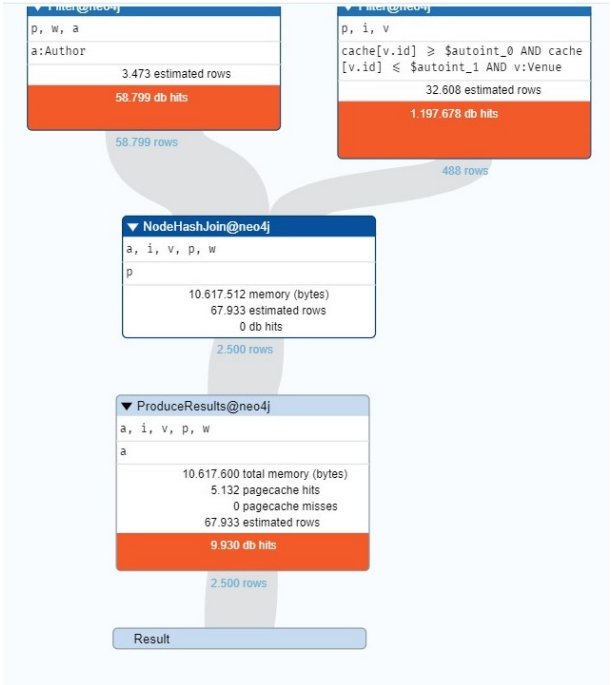
2. **Find papers in a set of venues, with a determined type (Execution time 2ms)**:

```
MATCH (p: Paper) -[w:writing]->(a:Author), (p: Paper) -[i: in_]->
    (v:Venue)
WHERE v.id >= 140000000 and v.id <= 140900000 AND p.doc_type='
    Journal'
RETURN a
```



Below we can see the output of the profile statement, that is used to track the query and the numbers of rows of each operation. It starts by scanning all the nodes with the papers, then it expands all the nodes with the 'writing' relationship on the left of the picture and the 'in' relationship on the right. Finally, it applies the filters on the venue id and on the paper's doc type and returns the results.

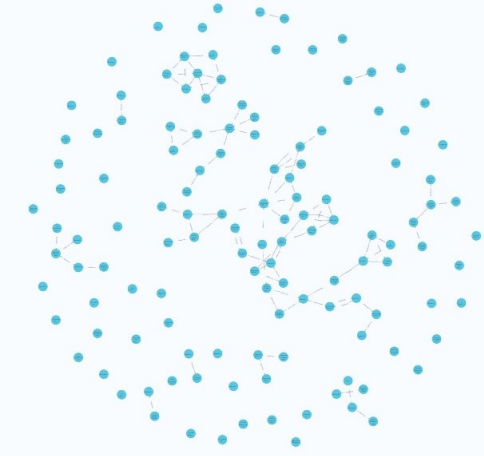




3. **Find papers of a determined main argument, written after a certain date (Execution time 30ms)**:

```
MATCH (p: Paper)-[r:referencing]->(p2: Paper), (p2: Paper)-[d:
    dealing]->(f: Fos)
```

```
WHERE (f.weight >= 0.45) AND (p2.year >2010) AND (f.name="
    Artificial intelligence")
RETURN p
```



4. **Count the authors that have written a famous paper of a determined argument (Execution time 38ms):**

```
MATCH (p: Paper)-[w:writing]->(a:Author), (p: Paper)-[d: dealing]->(f:
    Fos)
WHERE f.name = "Machine learning" AND p.n_citation>5000
RETURN COUNT(a.id) AS num_aut
```

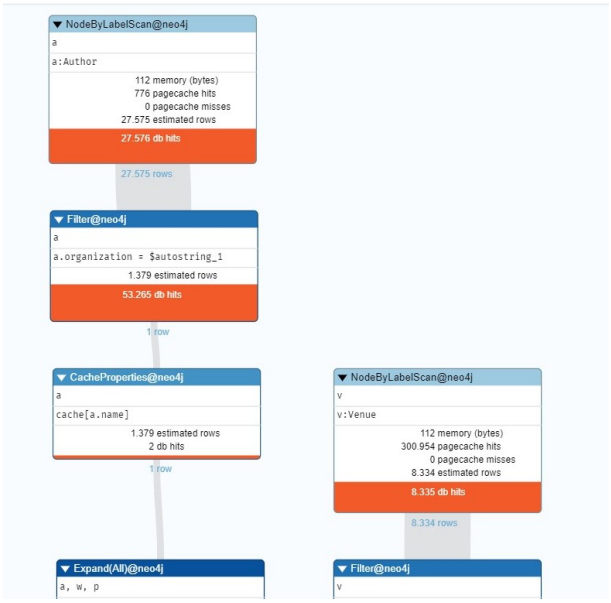
	num_aut
1	2120

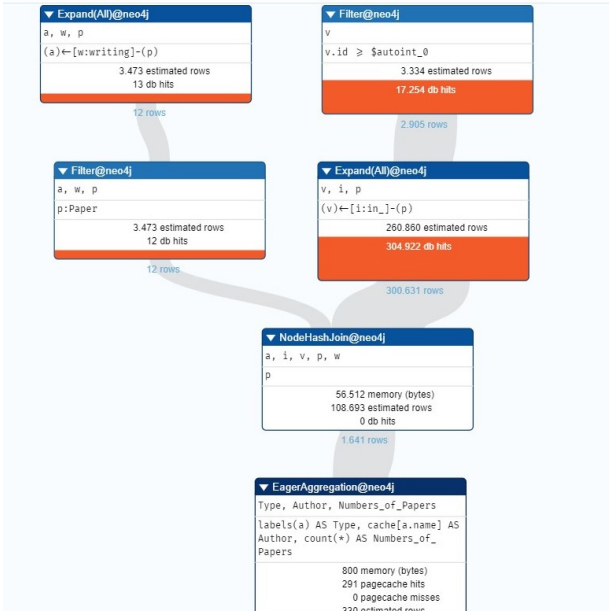
5. **Count the papers divided per author written in a set of venue by a determined authors' organization (Execution time 103ms):**

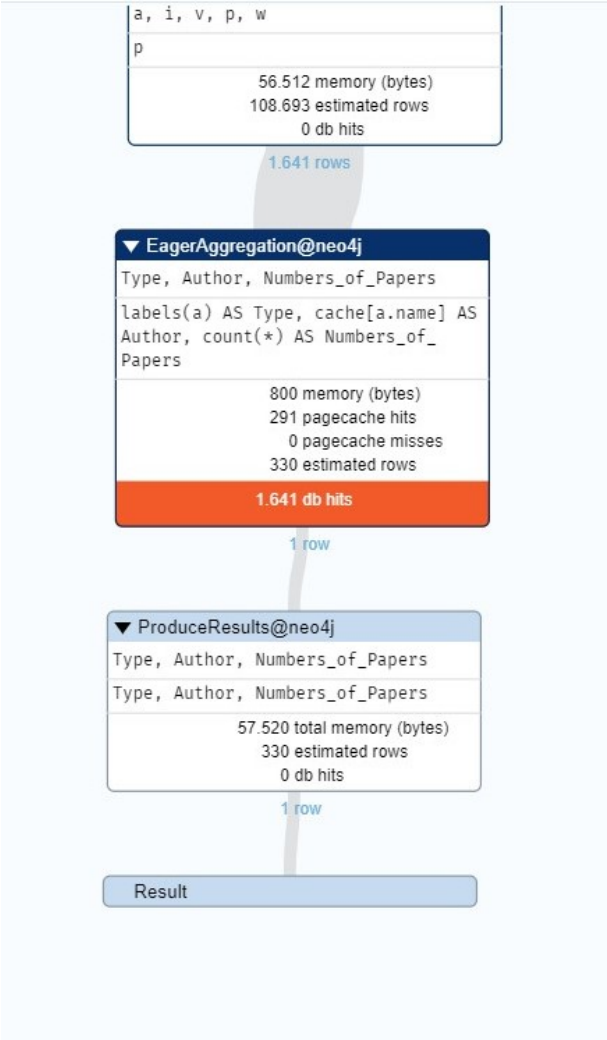
```
MATCH (p: Paper) -[w:writing]->(a:Author), (p: Paper) -[i: in_]->(v
    : Venue)
WHERE v.id >= 160000000 AND a.organization='Microsoft Research, One
    Microsoft Way, Redmond, WA 98052, USA. hshum@microsoft.com#TAB#
    '
RETURN labels(a) AS Type, a.name AS Author, COUNT(*) AS
    Numbers_of_Papers
```

	Type	Author	Numbers_of_Papers
1	["Author"]	"Heung-Yeung Shum"	1641

The profile statement scans all the authors, then it expands the 'writing' relationship and applies the organization's filter. At the same time it scans all the venues and filters their id. At the end the aggregation is applied and all the resulting values are returned.







6. **Calculate the weight keywords' average of papers that have a determined publisher and a reference to a famous paper (Execution time 27ms):**

```
MATCH (p: Paper) -[d: dealing]->(f: Fos), (p: Paper) -[r:
    referencing]->(p2: Paper)
WHERE p2.n_citation>2000 AND p.publisher='Society for Information
    Management and The Management Information Systems Research
    Center'
RETURN p.title, avg(f.weight)
```

	p.title	avg(f.weight)
1	"Shackled to the status quo: the inhibiting effects of incumbent system habit, switching costs, and inertia on new system acceptance"	0.449569698283237
2	"Technostress: technological antecedents and implications"	0.44386502606134975
3	"Business intelligence in blogs: understanding consumer interactions and communities"	0.41906192304436024
4	"Web and wireless site usability: understanding differences and modeling use"	0.42455533500106846
5	"Competing perspectives on the link between strategic information technology alignment and organizational agility: insights from a mediation model"	0.4178663593780648
6	"Reliability, mindfulness, and information systems"	0.42862394017677813

7. **Count the numbers of bilateral reference between two papers that have the same venue and a large number of citation (Execution time 119ms):**

```
MATCH (p: Paper)-[r:referencing]->(p2: Paper), (p2: Paper)-[r2:
    referencing]->(p: Paper), (p: Paper)-[w:writing]->(a: Author),(p
    : Paper)-[i:in_]->(v: Venue), (p2: Paper)-[i2:in_]->(v2: Venue)
WHERE p2.n_citation>500 AND p.n_citation>500 AND v.name=v2.name AND
     a.organization="Royal Institute of Technology"
RETURN type(r) AS Relation, COUNT(*)/2 AS
    Num_of_bilateral_referencingsame_venue, a.organization AS
    organization
LIMIT 1
```

	Relation	Num_of_bilateral_referencing_same_venue	organization
1	"referencing"	28900	"Royal Institute of Technology"

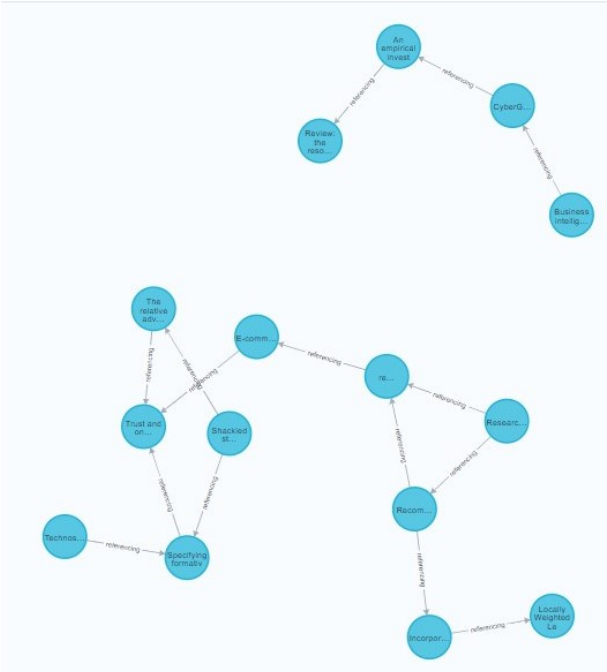
8. **Count the total citation of a paper that have references to two papers that deal of different field of study (Execution time 1849ms):**

```
MATCH (p: Paper)-[r: referencing]->(p2: Paper), (p: Paper)-[r2:
    referencing]->(p3: Paper), (p2: Paper)-[d: dealing]->(f: Fos), (
    p3: Paper)-[d2: dealing]->(f2: Fos)
WHERE p.page_end-p.page_start>10 AND p2 <> p3 AND f.name='
    Artificial Intelligence' AND f.weight>0.0 and f2.name = 'Machine
     learning' AND f2.weight>0.0
RETURN p.title AS Title, SUM(p.n_citation) AS Sum_n_citation
LIMIT 5
```

	Title	Sum_n_citation
1	"The SUN Attribute Database: Beyond Categories for Deeper Scene Understanding"	890400
2	"Composition in Distributional Models of Semantics"	4053440
3	"Wikipedia-based semantic interpretation for natural language processing"	2170880
4	"Knowledge derived from wikipedia for computing semantic relatedness"	737760
5	"A survey of paraphrasing and textual entailment methods"	4324800

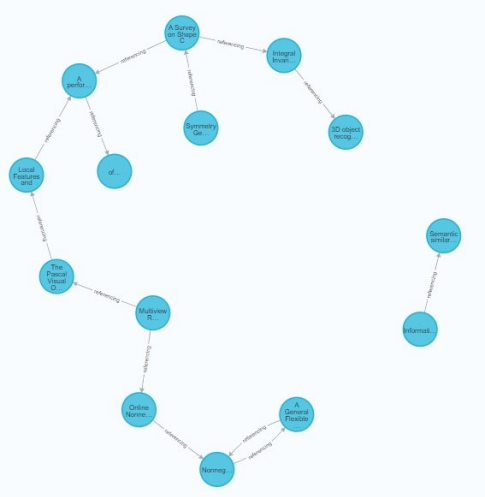
9. **Find the shortest path between two different paper that have a reference in common where the first is less famous than the second one (Execution time 61ms):**

```
MATCH s = shortestPath(
(p: Paper)-[r:referencing*]->(p2: Paper)
)
WHERE p2.n_citation>10*p.n_citation AND p <> p2
RETURN s
LIMIT 5
```

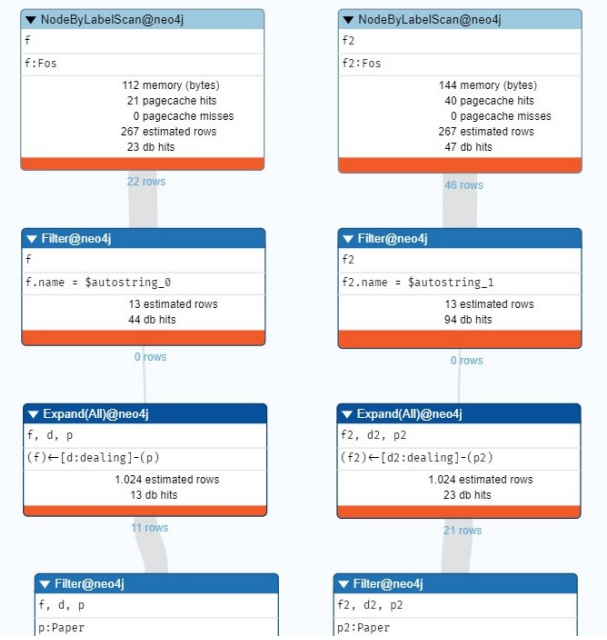


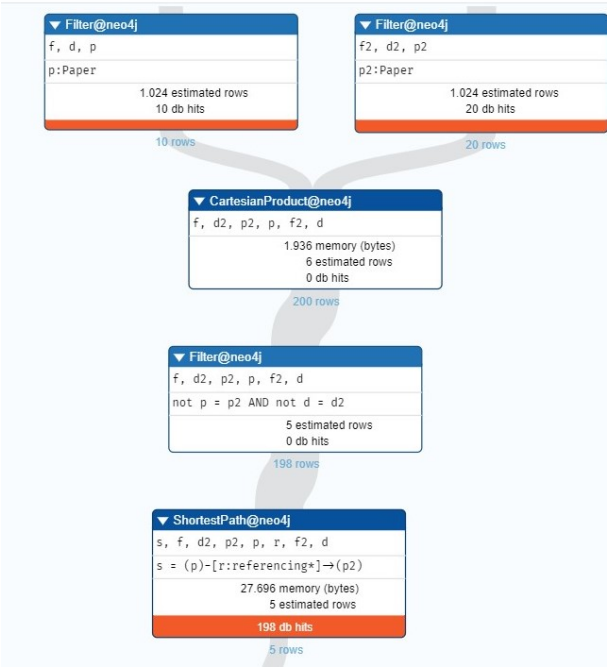
10. **Find the shortest path between two different paper that have a reference in common different FoS (Execution time 42ms)**:

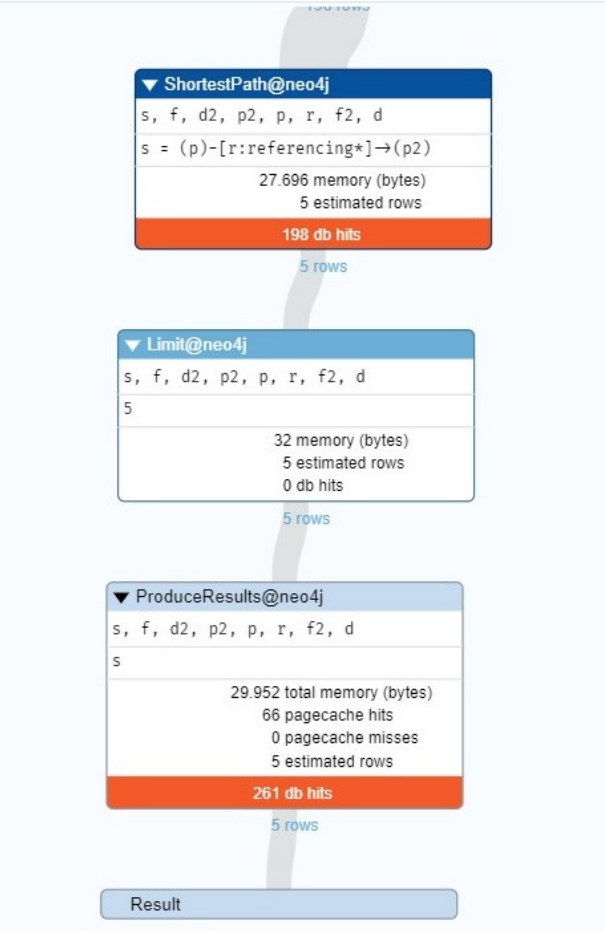
```
MATCH (p: Paper)-[d: dealing]->(f:Fos), (p2: Paper)-[d2:dealing]
   ->(f2:Fos), s = shortestPath( (p: Paper)-[r:referencing*]->(p2:
   Paper) )
WHERE f.name = "Artificial intelligence" AND f2.name = "Machine
   learning" AND p <> p2
RETURN s
LIMIT 5
```



In the following page we can see that the profile statement scans the FoS, applies the filters to them and expands the 'dealing' relationship. Then it checks that paper1 and paper2 are different and in the end it calls the function 'shortestPath' and returns the results.


NodeByLabelScan@neo4j
f
f:Fos
112 memory (bytes)
21 pagecache hits
0 pagecache misses
267 estimated rows
23 db hits
22 rows
NodeByLabelScan@neo4j
f2
f2:Fos
144 memory (bytes)
40 pagecache hits
0 pagecache misses
267 estimated rows
47 db hits
46 rows
Filter@neo4j
f
f.name = $autostring_0
13 estimated rows
44 db hits
0 rows
Filter@neo4j
f2
f2.name = $autostring_1
13 estimated rows
94 db hits
0 rows
Expand(All)@neo4j
f, d, p
(f)←[d:dealing]-(p)
1.024 estimated rows
13 db hits
11 rows
Expand(All)@neo4j
f2, d2, p2
(f2)←[d2:dealing]-(p2)
1.024 estimated rows
23 db hits
21 rows
Filter@neo4j
f, d, p
p:Paper
Filter@neo4j
f2, d2, p2
p2:Paper


Filter@neo4j
f, d, p
p:Paper
1.024 estimated rows
10 db hits
10 rows
Filter@neo4j
f2, d2, p2
p2:Paper
1.024 estimated rows
20 db hits
20 rows
CartesianProduct@neo4j
f, d2, p2, p, f2, d
1.936 memory (bytes)
6 estimated rows
0 db hits
200 rows
Filter@neo4j
f, d2, p2, p, f2, d
not p = p2 AND not d = d2
5 estimated rows
0 db hits
198 rows
ShortestPath@neo4j
s, f, d2, p2, p, r, f2, d
s = (p)-[r:referencing*]→(p2)
27.696 memory (bytes)
5 estimated rows
198 db hits
5 rows



4.4. Creation/Update Commands

1. **Insertion of a new author in the database:**

```
CREATE (a:Author {id: 3000000000, name:"Mario Rossi", organization:
    "Politecnico di Milano"})
```

2. **Insertion of a new field of study in the database:**

```
CREATE (f:Fos {weight:0.57640203, name:"Ambient intelligence" })
```

3. **Insertion of a new venue in the database:**

```
CREATE (v:Venue { id:3000000001, name:"Journal of Cloud Computing"
    })
```

4. **Insertion of a new paper in the database, paper's relationships with authors, venue, fields of study and other papers are created:**

```
MATCH (a:Author) WHERE a.id=3000000000
MATCH (f:Fos) WHERE f.name="Cloud computing" AND f.weight=0.6410412
```

```
MATCH (v:Venue) WHERE v.id=3000000001
MATCH (ref:Paper) WHERE ref.doi="10.1109/JPROC.2015.2483592"
MATCH (cit:Paper) WHERE cit.doi="10.1007/s10845-008-0158-5"
CREATE (p:Paper { id: 3000000000, title: "Application of
   deterministic, stochastic, and hybrid methods for cloud provider
    selection", year: 2022, doi: "10.1186/s13677-021-00275-1",
   volume: "11" , issue: "1" , abstract: "Cloud Computing
   popularization inspired ... requests.", n_citation: 5,
   page_start: 5, page_end: 10, publisher: "SpringerOpen", doc_type
   : "Journal" })
CREATE (p)-[w:writing]->(a), (p)-[r:referencing]->(ref), (cit)-[r:
   referencing]->(p), (p)-[i:in_]->(v), (p)-[d:dealing]->(f)
```

5. **Update of the organization of an author**:

```
MATCH (a:Author {name: 'Stefano Ceri'})
SET a.organization = "Politecnico di Milano"
```

6. **Update of the number of citations of a paper**:

```
MATCH (p:Paper {title: "Markov localization for mobile robots in
   dynamic environments"})
SET p.n_citation = 728
```

5 | References

- **Aminer (data set):** https://www.aminer.org/citation
- **Draw.io:** https://app.diagrams.net/
- **Neo4j:** https://neo4j.com/
- **Overleaf:** https://www.overleaf.com/
- **Python:** https://www.python.org/