

Data Mining 2nd Assignment - Neo4j Graph database

Nikolaos Tsoukalelis¹ and Georgios Chalkiopoulos²

¹p3352123 , ²p3352124

Emails: ntsoukalelis@aueb.gr , gchalkiopoulos@aueb.gr

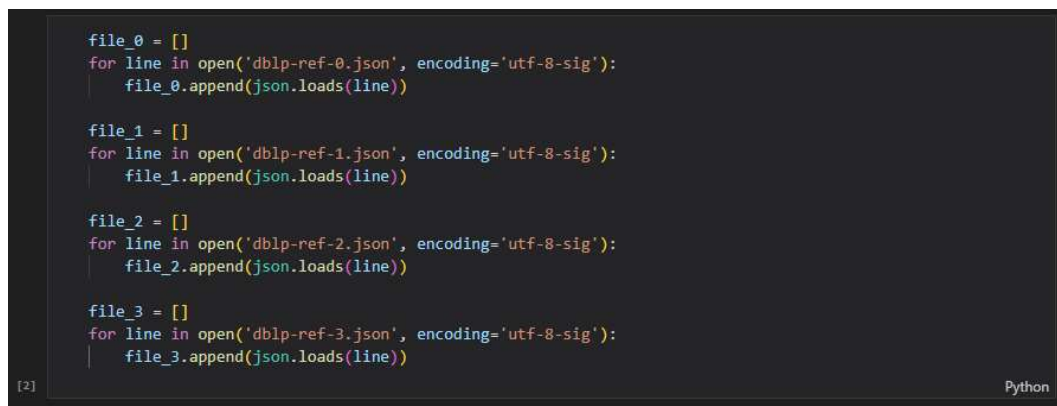
March 19, 2023

1 Assignment Description: Neo4j Graph database

You are given the DBLP citation network, which contains authors, articles, venues and citations between articles. In particular, the dataset contains 184313 articles with id, title, year and abstract (don't use the number of citations property), 80299 authors with names, 4 venues with names and 289908 citations among papers. You can download the dataset (Citation Network Dataset) from e-class in json format.

1.1 Dataset & pre-processing - from Python to Neo4j

In this assignment we decided to work in Python in order to pre-process our data and prepare them for import into the Neo4j software. The dataset was given in the form of 4 json files. These files contain the fields mentioned above. We firstly read these files using Python line by line, dropping them into lists (append). See below:



```
file_0 = []
for line in open('dblp-ref-0.json', encoding='utf-8-sig'):
    file_0.append(json.loads(line))

file_1 = []
for line in open('dblp-ref-1.json', encoding='utf-8-sig'):
    file_1.append(json.loads(line))

file_2 = []
for line in open('dblp-ref-2.json', encoding='utf-8-sig'):
    file_2.append(json.loads(line))

file_3 = []
for line in open('dblp-ref-3.json', encoding='utf-8-sig'):
    file_3.append(json.loads(line))
```

Figure 1: Reading the 4 json files and dropping all the lines of each file in a list.

Then we merged these 4 lists into 1 list, that contains our full data. We wanted as an ultimate goal to convert our data into csv files to import them in Neo4j. Next step was to create a csv file, containing all of our data in the form of a table and read this csv as dataframe in order to edit and perform data cleaning. See below the respective Python code:

```
# merging the 4 lists into 1
inout_data_list = file_0 + file_1 + file_2 + file_3

# saving the dataset into a csv - table format
with open('output.csv', 'w', encoding='utf-8-sig', newline='') as f:
    fieldnames = ['abstract', 'authors', 'n_citation', 'references', 'title', 'venue', 'year', 'id']
    writer = csv.DictWriter(f, fieldnames=fieldnames)

    writer.writeheader()
    for row in inout_data_list:
        writer.writerow(row)

# reading the csv files to create a dataframe for our preprocess
data_for_neo_4j = pd.read_csv('output.csv', encoding='utf-8')
data_for_neo_4j.head()
```

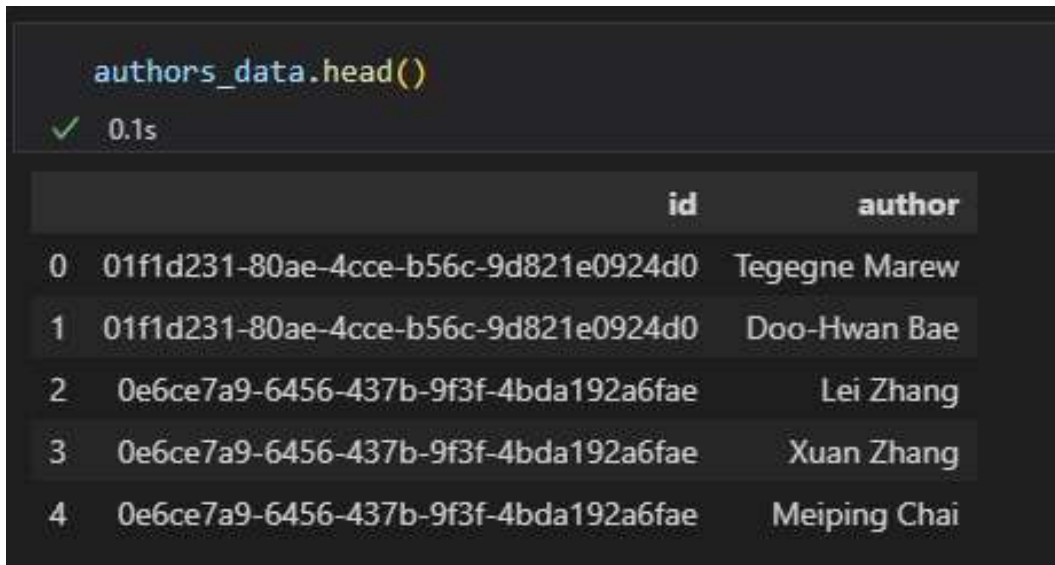
[8] ✓ 4.7s Python

	abstract	authors	n_citation	references	title	venue	year	id
0	NaN	['Tegegne Marew', 'Doo-Hwan Bae']	1	['2134bf3b-fd89-4724-90ce-5993b4fa3218', '906c...']	Using Classpects for Integrating Non-Functiona...	international conference on software engineering	2006	01f1d231-80ae-4cce-b56c-9d821e0924d0
1	NaN	['Lei Zhang', 'Xuan Zhang', 'Meiping Chai', 'Y...']	2	['3e3b524c-70c5-4008-b349-fd7ae950e655', '4929...']	Solution Proposals for Japan-Oriented Offshore...	international conference on software engineering	2009	0e6ce7a9-6456-437b-9f3f-4bda192a6fae
2	NaN	['Dongyun Liu', 'Hong Mei']	39	['4b837f17-7e38-4175-82bc-daa37f162933', '65ac...']	Mapping Requirements to Software Architecture ...	international conference on software engineering	2003	10c7185a-f2b7-4810-b1d6-1340c2949922
3	IEEE 802.11e Medium Access Control (MAC) is an...	['N. Sai Shankar', 'Sunghyun Choi']	50	NaN	QoS Signaling for Parameterized Traffic in IEE...	Lecture Notes in Computer Science	2002	11f0bd37-ae5a-43e6-b14a-a59bc00fdd90
4	The aim of this paper is to develop an executa...	['C. Graciani Díaz', 'Francisco-Jesús Martín-M...']	50	['c17481ca-9511-4793-8dad-a2486e0b2713']	Specification of Adleman's Restricted Model Us...	Lecture Notes in Computer Science	2002	155dec16-36d6-44f4-976b-1afb5d1924af

Figure 2: Converting our dataset into a table and loading it as a dataframe to edit it. It is obvious the format of the data is not in a good format.

As our data are in the table format, we proceed in the pre-processing phase. First thing to do is to remove from the columns 'authors' and 'references' the unnecessary square brackets, quotes, double quotes etc ([,],','). We also drop the n_citation column as it is not going to be needed. Then, as we wanted to create datasets, that correspond to the graph network we had in mind (2 nodes: Articles and Authors

with relationships between them WROTE & CITES) we split our main dataset into 3 smaller - articles_data, authors_data, references (5). Last thing we did was to create rows based on the delimiter of the columns 'authors' and 'references'. By doing that we created input datasets for Neo4j in the following format (every id that has more than one article authors or references is repeated in as many rows as the total of the authors or the references)

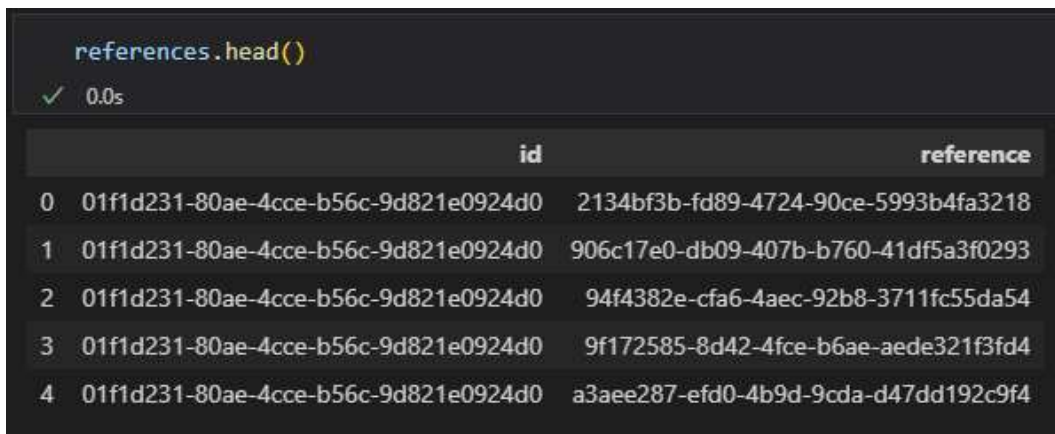


```
authors_data.head()
```

✓ 0.1s

	id	author
0	01f1d231-80ae-4cce-b56c-9d821e0924d0	Tegegne Marew
1	01f1d231-80ae-4cce-b56c-9d821e0924d0	Doo-Hwan Bae
2	0e6ce7a9-6456-437b-9f3f-4bda192a6fae	Lei Zhang
3	0e6ce7a9-6456-437b-9f3f-4bda192a6fae	Xuan Zhang
4	0e6ce7a9-6456-437b-9f3f-4bda192a6fae	Meiping Chai

Figure 3: As it can be seen the first article-id is repeated two times as it has 2 authors, the same with the second (3 times respectively).



```
references.head()
```

✓ 0.0s

	id	reference
0	01f1d231-80ae-4cce-b56c-9d821e0924d0	2134bf3b-fd89-4724-90ce-5993b4fa3218
1	01f1d231-80ae-4cce-b56c-9d821e0924d0	906c17e0-db09-407b-b760-41df5a3f0293
2	01f1d231-80ae-4cce-b56c-9d821e0924d0	94f4382e-cfa6-4aec-92b8-3711fc55da54
3	01f1d231-80ae-4cce-b56c-9d821e0924d0	9f172585-8d42-4fce-b6ae-aede321f3fd4
4	01f1d231-80ae-4cce-b56c-9d821e0924d0	a3aee287-efd0-4b9d-9cda-d47dd192c9f4

Figure 4: A view of references dataset that contains article-id as the first column and the references that appear in it as the second column.

Below is presented the code for the pre-process procedure stated in the paragraph above:

```
#pre-processing phase - remove square brackets, quotes, double quotes etc ([, ], ', ")
data_for_neo_4j['authors'] = data_for_neo_4j['authors'].str.strip('[]')
data_for_neo_4j['references'] = data_for_neo_4j['references'].str.strip('[]')

#pre-processing phase - drop n_citation as it's not needed
data_for_neo_4j = data_for_neo_4j.drop('n_citation', axis=1)
data_for_neo_4j['references'] = data_for_neo_4j['references'].astype(str).str.replace('[ ]', '', regex=True)
data_for_neo_4j['authors'] = data_for_neo_4j['authors'].astype(str).str.replace('[ ]', '', regex=True)
data_for_neo_4j['authors'] = data_for_neo_4j['authors'].astype(str).str.replace('["]', '', regex=True)

#save to csv
data_for_neo_4j.to_csv('C:/Users/cob_n/Neo4j/Desktop/relate-data/dbms/dbms-1fc8a5b8-6b7e-428c-b356-e52027d827bf/import/data_for_neo_4j.csv', header=False, index=False, encoding='utf-8')

#pre-processing phase - split data to 3 different datasets and continue
articles_data = data_for_neo_4j[['id', 'title', 'year', 'venue', 'abstract']]
authors_data = data_for_neo_4j[['id', 'authors']]
references_data = data_for_neo_4j[['id', 'references']]

#pre-processing phase - split based on delimiter and creating rows with authors names
authors_data = authors_data.assign(author=authors_data['authors'].str.split(','),).explode('author').reset_index(drop=True)
authors_data = authors_data.drop(columns='authors')

#pre-processing phase - split based on delimiter and creating rows with references
references = references.assign(reference=references['references'].str.split(','),).explode('reference').reset_index(drop=True)
references = references.drop(columns='references')
references['reference'] = references['reference'].fillna('unknown')

#pre-processing phase - remove whitespaces
articles_data['id'] = articles_data['id'].str.strip()
articles_data['title'] = articles_data['title'].str.strip()
articles_data['year'] = articles_data['year'].str.strip()
articles_data['venue'] = articles_data['venue'].str.strip()

references['id'] = references['id'].str.strip()
references['reference'] = references['reference'].str.strip()

authors_data['id'] = authors_data['id'].str.strip()
authors_data['author'] = authors_data['author'].str.strip()

#pre-processing phase - save to csv - there are the input data for Neo4j
articles_data.to_csv('C:/Users/cob_n/Neo4j/Desktop/relate-data/dbms/dbms-1fc8a5b8-6b7e-428c-b356-e52027d827bf/import/articles_data.csv', header=False, index=False)
authors_data.to_csv('C:/Users/cob_n/Neo4j/Desktop/relate-data/dbms/dbms-1fc8a5b8-6b7e-428c-b356-e52027d827bf/import/authors_data.csv', header=False, index=False)
references.to_csv('C:/Users/cob_n/Neo4j/Desktop/relate-data/dbms/dbms-1fc8a5b8-6b7e-428c-b356-e52027d827bf/import/references.csv', header=False, index=False)
```

Figure 5: The steps for the final preprocess of our data, so they can be input in Neo4j. The three datasets resemble the graph Network we designed as a raw plan - 2 nodes being Articles and Authors and 2 relationships between them WROTE & CITES (CITES is between articles based on reference dataset).

1.2 Creating the graph database

As mentioned above we firstly draw the graph database sketch, and then we moved to the creation of the datasets in accordance to the sketch we had in mind (7). After that we started importing our data in our database in Neo4j. We created our project database:

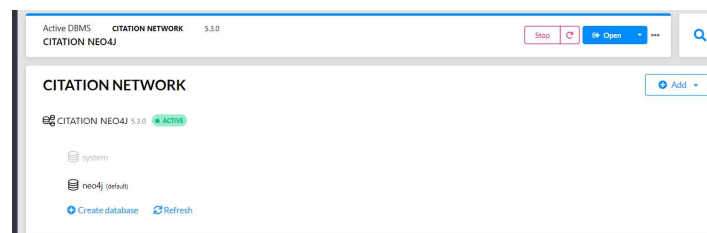


Figure 6: Neo4j desktop environment - the project we worked on.

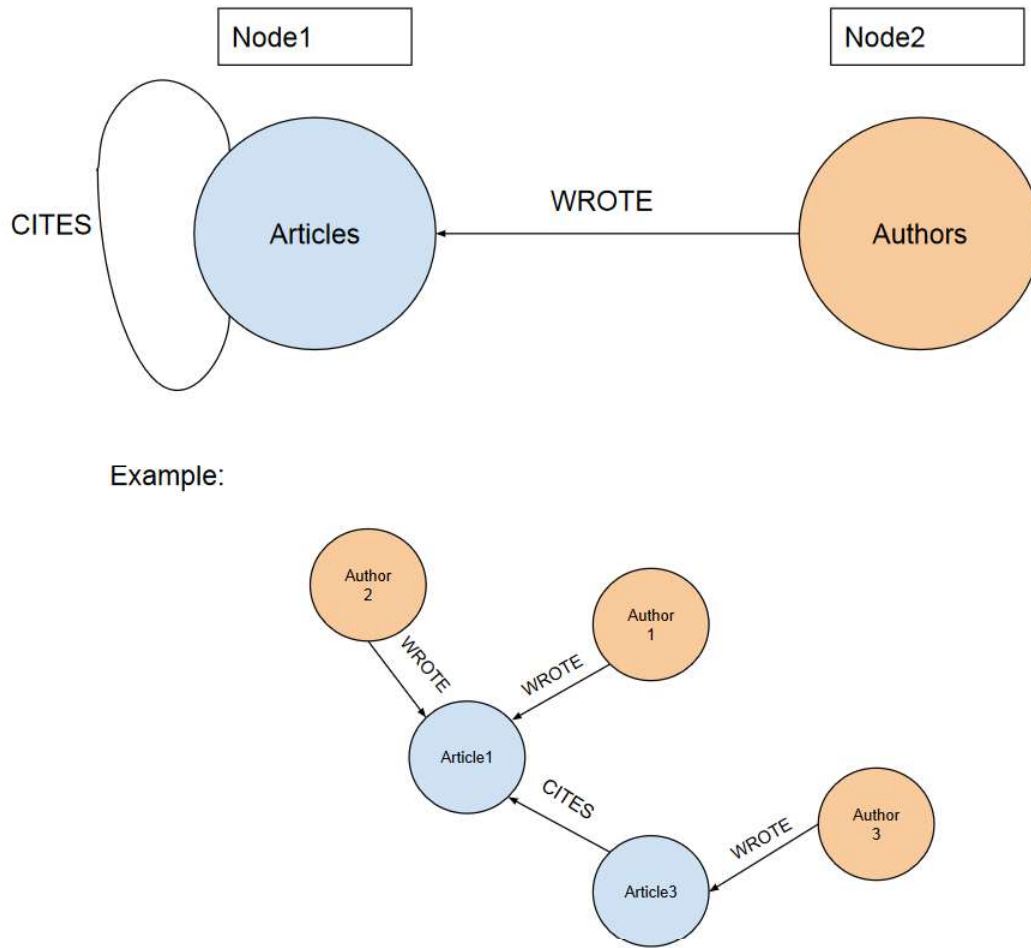


Figure 7: Graph database sketch - Nodes and relationships.

We then started our work in Cypher and we imported the Articles dataset, and consequently created the first node of our database.

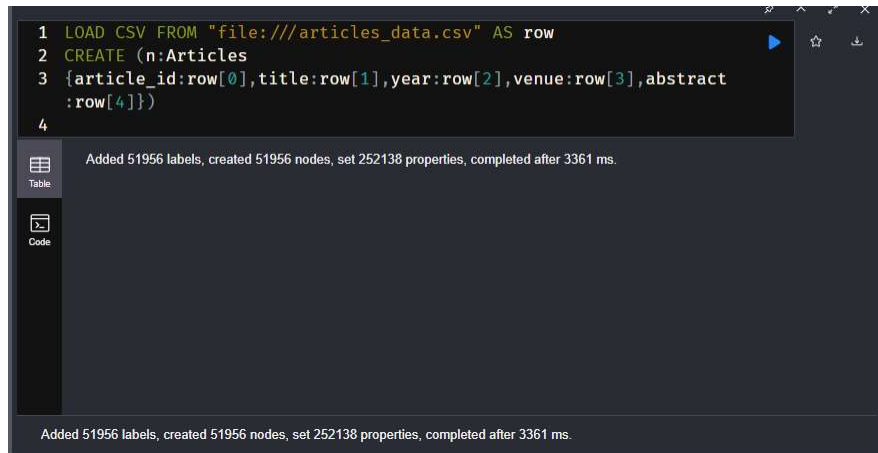
2 Querying the database

After having correctly input our data in the Neo4j we continue in querying the database to get insights and KPIs regarding our data.

2.1 Importing the data

Below we will showcase the queries and the results from our interaction with the Neo4j databases. Firstly we started by loading the data in our database.

For the Articles' node:



```
1 LOAD CSV FROM "file:///articles_data.csv" AS row
2 CREATE (n:Articles
3 {article_id:row[0],title:row[1],year:row[2],venue:row[3],abstract
:row[4]})
4
```

Added 51956 labels, created 51956 nodes, set 252138 properties, completed after 3361 ms.

Figure 8: Query for loading the data of the articles, in order to create the Articles' node.

We then loaded the second dataset, that is the authors_data.

Authors' node:



```
1 LOAD CSV FROM 'file:///authors_data.csv' AS row
2 WITH row[0] AS author_article_id, row[1] AS author_name
3 CREATE(b:Authors {author_article_id:author_article_id,
author_name:author_name})
4
```

Added 140602 labels, created 140602 nodes, set 281204 properties, completed after 2354 ms.

Figure 9: Query for loading the data of the authors, in order to create the Authors' node.

Third step in our process was to create a constraint especially for the creation of the relationships, which would take too much time, had not been for the constraint.

Constraint:



Figure 10: Constraint creation regarding the uniqueness of the article_id.

Fourth step in our process was to create the "WROTE" relationship.

Creation of the relationship between authors and article (Authors – WROTE→ Articles):

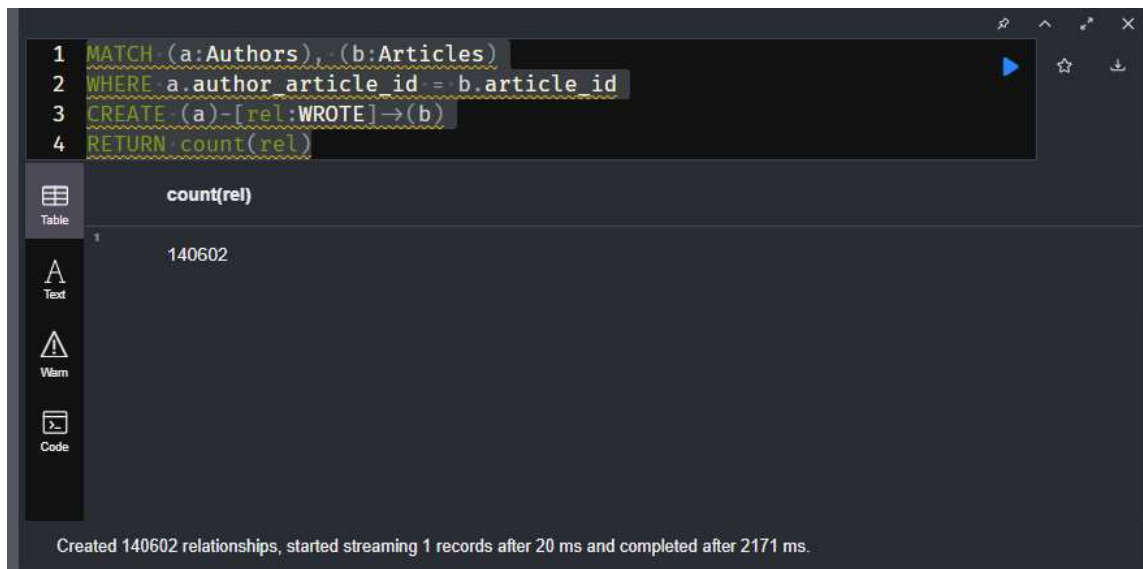


Figure 11: Query that describes the creation of the WROTE relationship.

After creating the relationship (WROTE) we wanted to visualize it by running the following query:

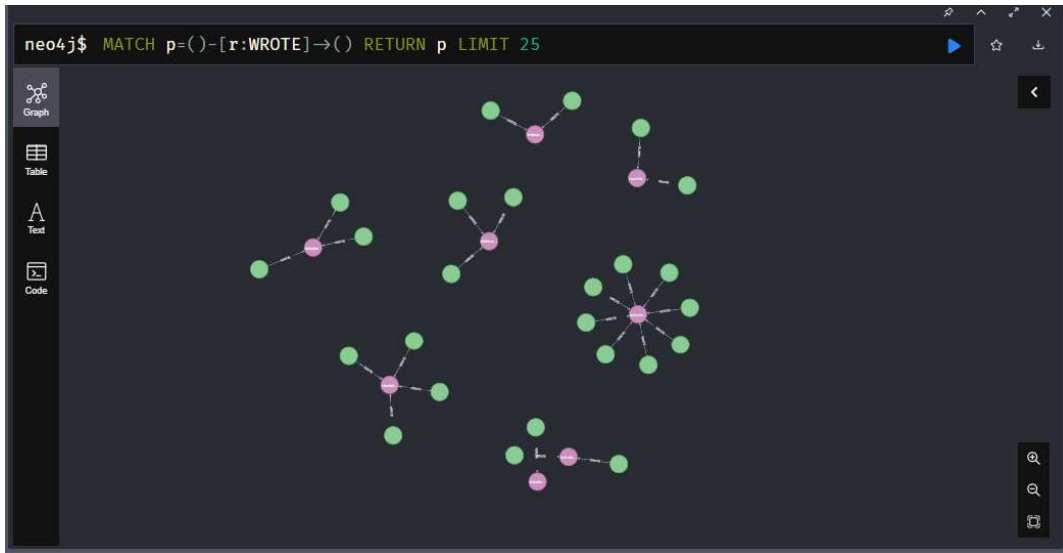


Figure 12: Query that visualizes the outcome after the creation of the relation between authors and articles. It is obvious that the green is the Authors and the pink represents the Articles.

Last thing to do was to upload the references dataset and create the relationship "CITES" between articles. This procedure was draining our resources when we first ran it, but with the addition of the constraint, it ran smoothly.

Creation of the relationship between authors and article (article_id – CITES → reference (another article_id):

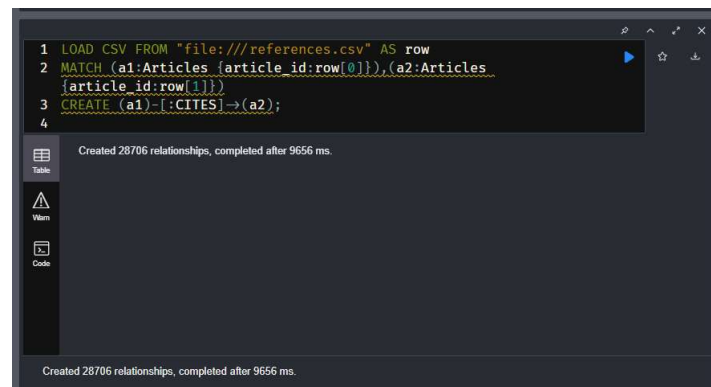


Figure 13: Query that creates the CITES relationship between articles.

Visualization of the CITES relationship was accomplished running the following query:

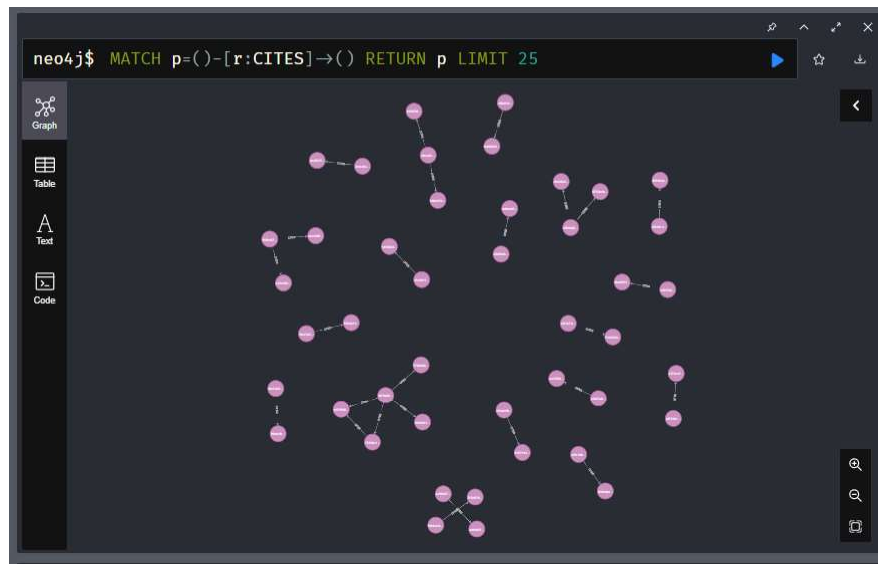


Figure 14: Query that creates the CITES relationship between articles.

We lastly reached to the point, that our database is created and by visualizing it, we notice that it resembles our original sketch.

Database created:

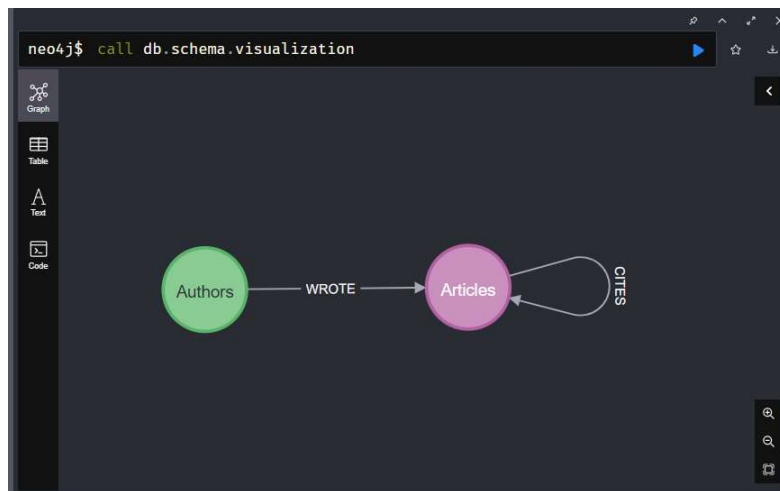
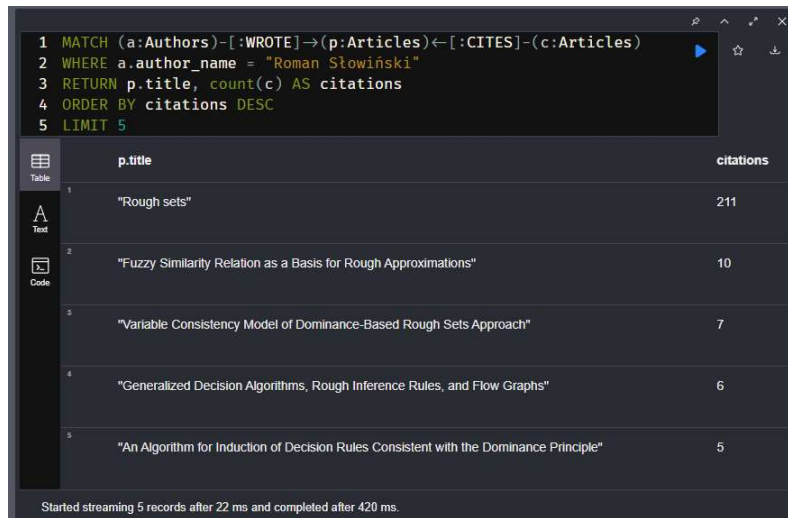


Figure 15: Graph database of Autors-Articles and the respective relationships WROTE and CITES.

2.2 Queries

After the creation of our database, we were asked to write and execute the following queries using the Cypher language.

1. Which are the top 5 papers with the most citations of the author “Roman Słowiński”. Return paper title and number of citations.



```

1 MATCH (a:Authors)-[:WROTE]-(p:Articles)-[:CITES]-(c:Articles)
2 WHERE a.author_name = "Roman Słowiński"
3 RETURN p.title, count(c) AS citations
4 ORDER BY citations DESC
5 LIMIT 5

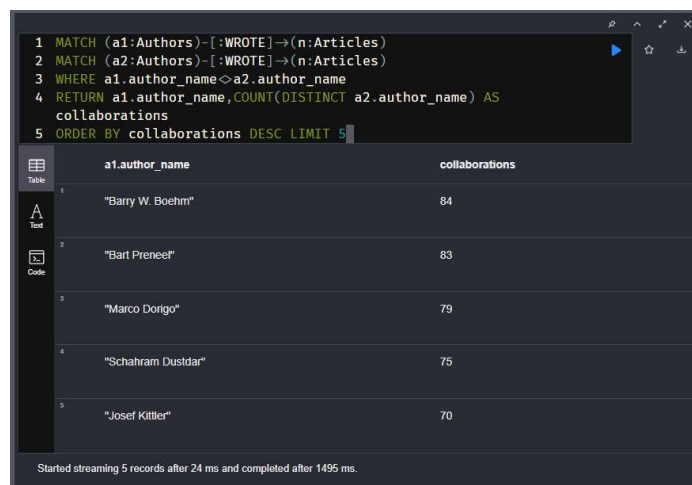
```

	p.title	citations
1	"Rough sets"	211
2	"Fuzzy Similarity Relation as a Basis for Rough Approximations"	10
3	"Variable Consistency Model of Dominance-Based Rough Sets Approach"	7
4	"Generalized Decision Algorithms, Rough Inference Rules, and Flow Graphs"	6
5	"An Algorithm for Induction of Decision Rules Consistent with the Dominance Principle"	5

Started streaming 5 records after 22 ms and completed after 420 ms.

Figure 16: Query 1 that returns the top 5 articles with the most citations.

2. Which are the top 5 authors with the most collaborations (with different authors). Return author name and number of collaborations.



```

1 MATCH (a1:Authors)-[:WROTE]-(n:Articles)
2 MATCH (a2:Authors)-[:WROTE]-(n:Articles)
3 WHERE a1.author_name <> a2.author_name
4 RETURN a1.author_name, COUNT(DISTINCT a2.author_name) AS collaborations
5 ORDER BY collaborations DESC LIMIT 5

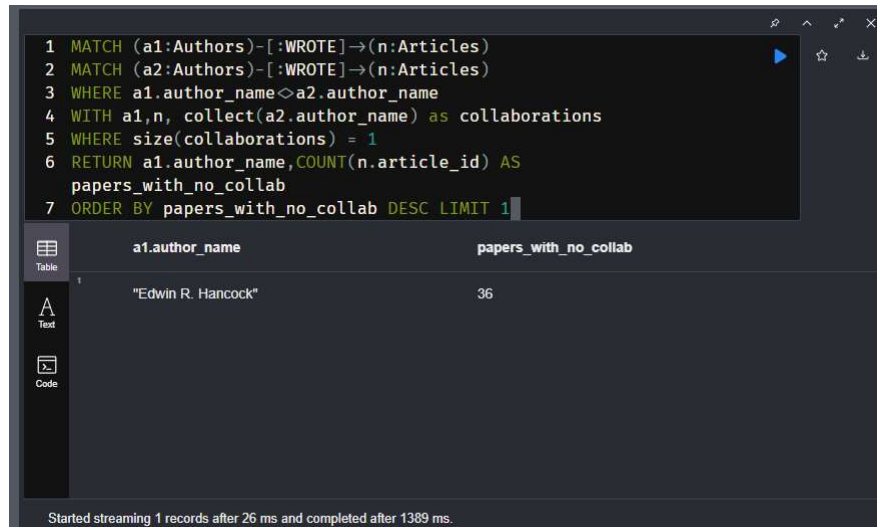
```

	a1.author_name	collaborations
1	"Barry W. Boehm"	84
2	"Bart Preneel"	83
3	"Marco Dorigo"	79
4	"Schahram Dustdar"	75
5	"Josef Kittler"	70

Started streaming 5 records after 24 ms and completed after 1495 ms.

Figure 17: Query 2 that returns the top 5 authors with the most collaborations.

3. Which is the author who has wrote the most papers without collaborations. Return author name and number of papers.



```
1 MATCH (a1:Authors)-[:WROTE]->(n:Articles)
2 MATCH (a2:Authors)-[:WROTE]->(n:Articles)
3 WHERE a1.author_name <> a2.author_name
4 WITH a1,n, collect(a2.author_name) as collaborations
5 WHERE size(collaborations) = 1
6 RETURN a1.author_name,COUNT(n.article_id) AS
papers_with_no_collab
7 ORDER BY papers_with_no_collab DESC LIMIT 1
```

a1.author_name	papers_with_no_collab
"Edwin R. Hancock"	36

Started streaming 1 records after 26 ms and completed after 1389 ms.

Figure 18: Query 3 that returns the top author without collaborations and his/her papers.

4. Which author published the most papers in 2009 to “Lecture Notes in Computer Science”? Return author name and number of papers.



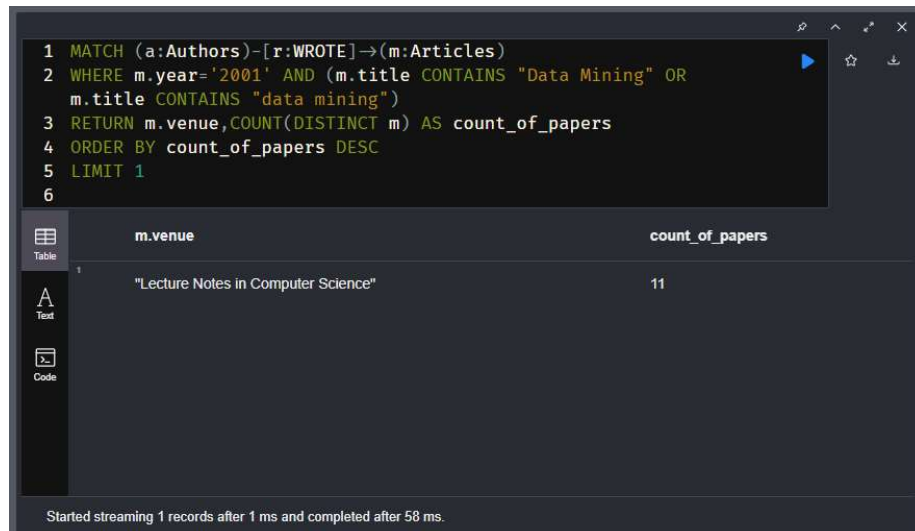
```
1 MATCH (a:Authors)-[r:WROTE]->(m:Articles)
2 WHERE m.venue CONTAINS 'Lecture Notes in Computer Science' AND
m.year="2009"
3 RETURN a.author_name, COUNT(*) AS papers
4 ORDER BY papers DESC
5 LIMIT 1
```

a.author_name	papers
"Florian Mender"	10

Started streaming 1 records after 1 ms and completed after 83 ms.

Figure 19: Query 4 that returns the top author with the most papers in 2009 to “Lecture Notes in Computer Science”.

5. Which is the venue with the most papers on the Data Mining field (derived from the paper title) in 2001. Return venue and number of papers.



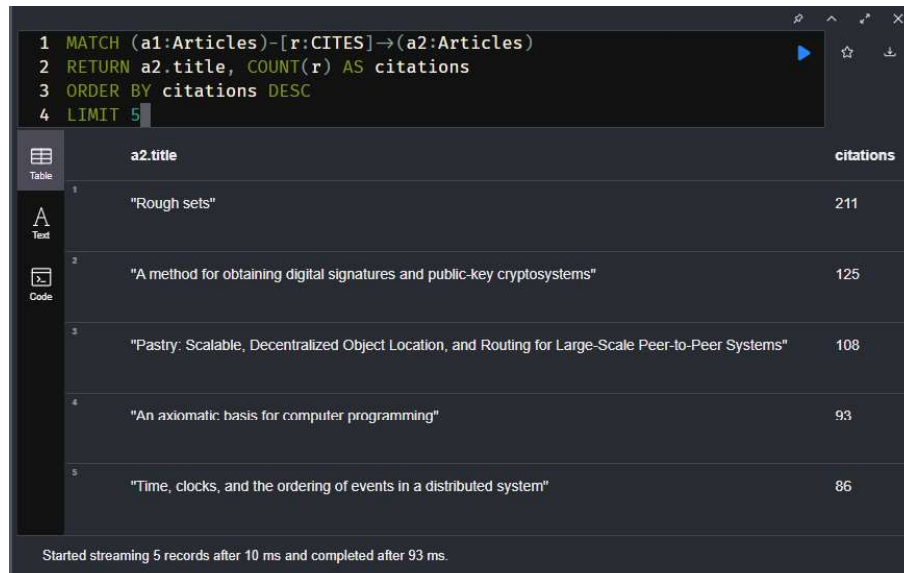
```
1 MATCH (a:Authors)-[r:WROTE]-(m:Articles)
2 WHERE m.year='2001' AND (m.title CONTAINS "Data Mining" OR
3 m.title CONTAINS "data mining")
4 RETURN m.venue,COUNT(DISTINCT m) AS count_of_papers
5 ORDER BY count_of_papers DESC
6 LIMIT 1
```

	m.venue	count_of_papers
1	"Lecture Notes in Computer Science"	11

Started streaming 1 records after 1 ms and completed after 58 ms.

Figure 20: Query 5 that returns the venue with the most papers about Data Mining.

6. Which are the top 5 papers with the most citations? Return paper title and number of citations.



```
1 MATCH (a1:Articles)-[r:CITES]-(a2:Articles)
2 RETURN a2.title, COUNT(r) AS citations
3 ORDER BY citations DESC
4 LIMIT 5
```

	a2.title	citations
1	"Rough sets"	211
2	"A method for obtaining digital signatures and public-key cryptosystems"	125
3	"Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems"	108
4	"An axiomatic basis for computer programming"	93
5	"Time, clocks, and the ordering of events in a distributed system"	86

Started streaming 5 records after 10 ms and completed after 93 ms.

Figure 21: Query 6 that returns the top 5 papers with the most citations.

7. Which were the papers that use “collaborative filtering” in “recommendation system” (derived from the paper abstract). Return authors, title.

```

1 MATCH (a:Authors)-[r:WROTE]->(m:Articles)
2 WHERE (m.abstract CONTAINS 'collaborative filtering' OR m.abstract CONTAINS 'Collaborative Filtering' OR m.abstract CONTAINS 'collaborative' ) AND (m.abstract CONTAINS 'recommend' OR m.abstract CONTAINS 'Recommend')
3 RETURN collect(a.author_name) AS Authors, m.title

```

	Authors	m.title
1	["Robin D. Burke"]	"A Case-Based Reasoning Approach to Collaborative Filtering"
2	["Stefano Aguzzoli", "Paolo Massa", "Paolo Avesani"]	"Collaborative Case-Based Recommender Systems"
3	["Mathias Bauer", "Dietmar Dengler", "Christian Schmitt"]	"Multivariate preference models and decision making with the MAUT machine"
4	["Joaquin Delgado", "Tomoki Ura", "Naohiro Ishii"]	"Intelligent Collaborative Information Retrieval"
5	["José David Martín-Guerrero"]	"A pseudo-supervised approach to improve a recommender based on collaborative filtering"
6	["Yong-Tae Woo", "Young-Ji Kim", "Soo-Ho Ok"]	"A Case-Based Recommender System Using Implicit Rating Techniques"

Started streaming 58 records in less than 1 ms and completed after 222 ms.

Figure 22: Query 7 that returns the papers that contain collaborative filtering in recommendation systems in their abstract.

1	Authors	m.title
2	["Robin D. Burke"]	"A Case-Based Reasoning Approach to Collaborative Filtering"
3	["Stefano Aguzzoli", "Paolo Massa", "Paolo Avesani"]	"Collaborative Case-Based Recommender Systems"
4	["Mathias Bauer", "Dietmar Dengler", "Christian Schmitt"]	"Multivariate preference models and decision making with the MAUT machine"
5	["Joaquin Delgado", "Tomoki Ura", "Naohiro Ishii"]	"Intelligent Collaborative Information Retrieval"
6	["José David Martín-Guerrero"]	"A pseudo-supervised approach to improve a recommender based on collaborative filtering"
7	["Yong-Tae Woo", "Young-Ji Kim", "Soo-Ho Ok"]	"A Case-Based Recommender System Using Implicit Rating Techniques"
8	["Patrick R. Paulson", "Aimilia Tzanavari"]	"Combining collaborative and Content-Based Filtering using Conceptual graphs"
9	["Josephine Griffith", "Colin O'Riordan", "Humphrey Sorensen"]	"A constrained spreading activation approach to collaborative filtering"
10	["Gustavo R. Alves", "Francisco de A. T. de Carvalho", "Byron L. D. Bezerra"]	"Collaborative filtering based on modal symbolic user profiles: Knowing you in the first meeting"
11	["Bary Smyth", "Keith Bradley", "Raehael Rafter"]	"Automated Collaborative Filtering Applications for Online Recruitment Services"
12	["Emilia Bakak", "Janusz Sobczyk", "Maria Slanina"]	"Application of hybrid recommendation in web-based cooking assistant"
13	["Jill Freyne", "Bary Smyth"]	"Further experiments in case-based collaborative web search"
14	["Paolo Avesani", "Paolo Massa", "Angelo Susi", "Michele Nori"]	"Collaborative Radio Community"
15	["Enrico Blandini", "Paolo Giorgini"]	"Implicit culture for information agents"
16	["Georg Lausen", "Car-Nicolas Ziegler"]	"Analyzing Correlation between Trust and User Similarity in Online Communities"
17	["Osmar R. Zaiane", "Alexander Sotile"]	"Finding Similar Queries to Satisfy Searches Based on Query Traces"
18	["Amaud De Bruyn", "David M. Pennock", "C. Lee Giles"]	"Offering Collaborative-Like Recommendations When Data Is Sparse: The Case of Attraction-Weighted Information Filtering"
19	["Nicholas Kushmerick"]	"Robustness Analyses of Instance-Based Collaborative Recommendation"
20	["Derek G. Bridge", "John Paul Kelly"]	"Ways of computing diverse collaborative recommendations"
21	["David C. Wilson", "Derry O'Sullivan", "Bary Smyth"]	"Improving Case-Based Recommendation: A Collaborative Filtering Approach"
22	["Jerome Kelleher", "Derek G. Bridge"]	"Experiments in Sparsity Reduction: Using Clustering in Collaborative Recommenders"
23	["Tzvi Kuflik", "Shlomo Berkovsky", "Francesco Ricci"]	"Cross-technique mediation of user models"
24	["Syed Sibte Raza Abidi", "Zeina Chedrawy"]	"An adaptive personalized recommendation strategy featuring context sensitive content adaptation"
25	["Menou Papagelis", "Dimitris Plexousakis", "Thomazakis Koutsouras"]	"Alleviating the sparsity problem of collaborative filtering using trust inferences"
26	["A. Negro", "Vittorio Scarano", "Maria Barra", "Paul P. Maglio"]	"GAS: Group Adaptive System"
27	["Jennifer Golbeck"]	"Generating predictive movie recommendations from trust in social networks"
28	["Derry O'Sullivan", "David C. Wilson", "Bary Smyth"]	"Data Mining Support for Case-Based Collaborative Recommendation"
29	["Rohit Balasubramanian", "Rachamalla Rahul Reddy", "K. Sridharan", "M. Sridhar", "D. Venkataramanan", "V. Kavitha"]	"A hybrid approach for recommendation system with added feedback component"
30	["Bary Smyth", "Maria Angela Ferraro"]	"Collaborative Maintenance - A Distributed, Interactive Case-Base Maintenance Strategy"
31	["Pyungsuek Choi", "Meehee Lee", "Yong-Tae Woo"]	"A Hybrid Recommender System Combining Collaborative Filtering with Neural Network"
32	["Adriano Venturini", "Francesco Ricci", "Nader Mirzadeh", "Bora Arslan"]	"ITR: A Case-Based Travel Advisory System"
33	["Elizabeth Gray", "Jean-Marc Seigneux", "Christian Damsgaard Jensen", "Yong Chen"]	"Trust propagation in small worlds"
34	["Thomas Holmann"]	"Learning What People (Don't) Want"
35	["Bilal Vaporiady", "Alexandra L. Utidenbogerd"]	"Combining demographic data with collaborative filtering for automatic music recommendation"
36	["Frank McCarey", "Nicholas Kushmerick", "Mel O'Connell"]	"Recommending library methods: an evaluation of the vector space model (VSM) and latent semantic indexing (LSI)"
37	["Mikael Sollenborn", "Peter Funk"]	"Category-Based Filtering in Recommender Systems for Improved Performance in Dynamic Domains"
38	["Jong-Hun Kim", "Kyoung-Yong Jung", "Jung-Hyun Lee"]	"Hybrid music filtering for recommendation based ubiquitous computing environment"
39	["Hania Mehra", "Veer San Dixi", "Punam Bedi"]	"Weighted difference entropy based similarity measure at two levels in a recommendation framework"
40	["Ali A. Abbasi", "Amin Javari", "Mahdi Jalil", "Hamid R. Rabiee"]	"Enhancing precision of Markov-based recommenders using location information"
41	["Ayse Bazar Bener", "Bora Caglayan", "Andriy V. Miranskyy"]	"Factors affecting team evolution during software projects"
42	["Yehuda Koren"]	"Collaborative filtering with temporal dynamics"
43	["Fabio Abbattista", "Hans H. K. Andersen", "Verner Andersen", "Pasquale Lops", "Giovanni Semeraro"]	"Evaluation and validation of a conversational agent embodied in a bookstore"
44	["Kyoung-Yong Jung"]	"User preference through learning user profile for ubiquitous recommendation systems"
45	["Mikael Sollenborn", "Peter Funk"]	"Category-Based Filtering and User Stereotype Cases to Reduce the Latency Problem in Recommender Systems"
46	["Amanda Rodda", "Tyler, Collin McMillan"]	"Collaborative software engineering education between college seniors and blind high school students"
47	["Margaret-Anne D. Storey", "Joseph Feliciano", "Aleksy Zagalski"]	"Student experiences using GitHub in software engineering courses: a case study"
48	["Gerhard Veltum", "Julia Luebenberger"]	"Query-log based authority analysis for Web information search"
49	["Masanori Sugimoto", "Yuchiro Takeuchi"]	"CityVoyager: An outdoor recommendation system based on user location history"
50	["Yoonseon Zhou", "Xin Jin", "Bomchod Mebsachei"]	"Semantically enhanced Collaborative Filtering on the Web"
51	["Yusuke Hayashi", "Mitsuru Ikeda"]	"Intellectual reputation to find an appropriate person for a role in creation and inheritance of organizational intellect"
52	["Mohammed Wasid", "Vibhor Kant", "Rashid Ali"]	"Frequency-based similarity measure for context-aware recommender systems"
53	["Anjali Gautam", "Punam Bedi", "Kunal Sindhwani", "Partha Chaudhary"]	"DSOARS: Content boosted context-aware recommendations using tensor factorization"
54	["Richa Punam Bedi", "Veeru Bhasin", "Sumit Kumar Agarwal"]	"ELM based imputation-boosted proactive recommender systems"
55	["Richa Punam Bedi"]	"Parallel context-aware recommender system using GPU and CUDA"
56	["Qing Li", "Byeong Man Kim"]	"Constructing user profiles for collaborative recommender system"
57	["Kyoung-Yong Jung", "Jung-Hyun Lee", "Aia Youngjoo"]	"FORAS: Fashion Design Recommender Agent System using the extraction of representative sensibility and the two-way combined filtering on textile"
58	["Peng Han", "Ramin Shen", "Bo Xie", "Fan Yang"]	"An adaptive spreading activation scheme for performing more effective collaborative recommendation"
59	["Barani Selvaraj", "Gopi Krishna Durbhaka"]	"Predictive maintenance for wind turbine diagnostics using vibration signal analysis based on collaborative recommendation approach"

Figure 23: The results of Query 7 in a csv.

3 Link Prediction


For this section of the assignment we will use the created neo4j graph database in order to predict future collaborations between authors. We will do that, by using link prediction algorithms as discussed in lecture “Link Prediction”.

3.1 Preparing the data - train & test splits

Before reaching to the point in which we could implement the Link Prediction algorithms we had to create the **CoAuthor** relation. From that point we can split our data to train and test dataset by creating 2 relations as **CoAuthorEarly** (train data, until 2005) and **CoAuthorLate** (test data, after 2006).

Following on, we will showcase the process of implementing the aforementioned steps:

We create the **CoAuthor** relation by collecting the year from the Articles node as well. We notice that the graph database is able of processing loads of data in just a few seconds.



```
1 MATCH (a1:Authors)-[:WROTE]->(p:Articles)<-[:WROTE]-(a2:Authors)
2 WITH a1, a2, p.year AS year
3 | ORDER BY a1, year
4 WITH a1,a2, collect(year)[0] AS year, COUNT(*) AS count_
5 MERGE (a1)-[e:CoAuthor {year:year}]->(a2)
6 SET e.count_=count_
```

Set 539796 properties, created 179932 relationships, completed after 4164 ms.

Table

Code

Set 539796 properties, created 179932 relationships, completed after 4164 ms.

Figure 24: Creation of CoAuthor relation

We then proceeded in the implementation of the other relations and realized that when we filtered data using year as integer (e.g < 2006), the query returned no changes. This is happening because Neo4j imports everything as String. So to avoid future problems, next step was to convert the field year to integer, something that would help in

the split between train and test.



Figure 25: Converting year to integer.

For the next step we created a subgraph as we wanted to work in it by creating a link prediction pipeline. Although this could be of better use if we would build a custom prediction model. Still we show the procedure below:

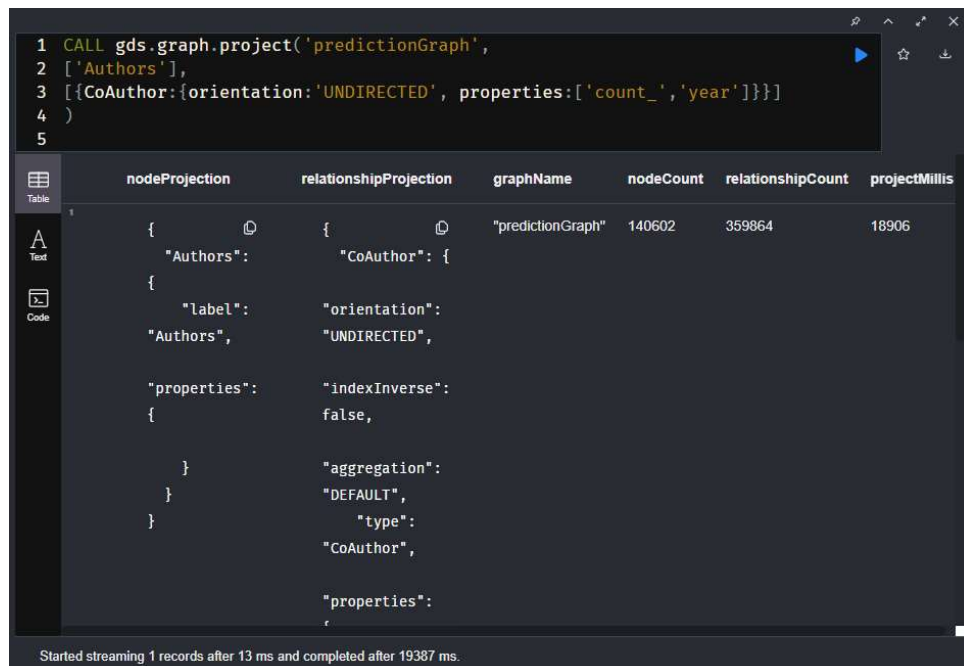
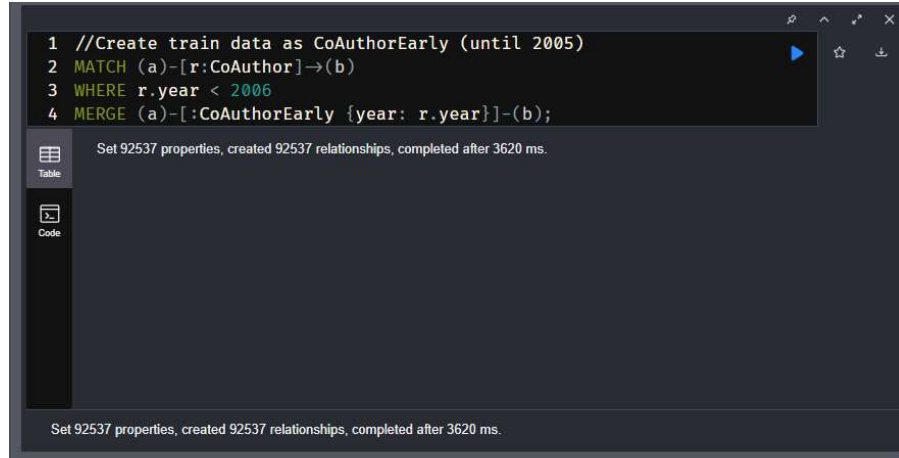


Figure 26: Creating subgraph 'predictionGraph'.

Splitting our data in train and test datasets. Train dataset consists of data before 2005 (until 2005) and test dataset from 2006 onwards. To distinguish them we create two new relations called **CoAuthorEarly** and **CoAuthorLate** respectively.



```
1 //Create train data as CoAuthorEarly (until 2005)
2 MATCH (a)-[r:CoAuthor]-(b)
3 WHERE r.year < 2006
4 MERGE (a)-[:CoAuthorEarly {year: r.year}]- (b);
```

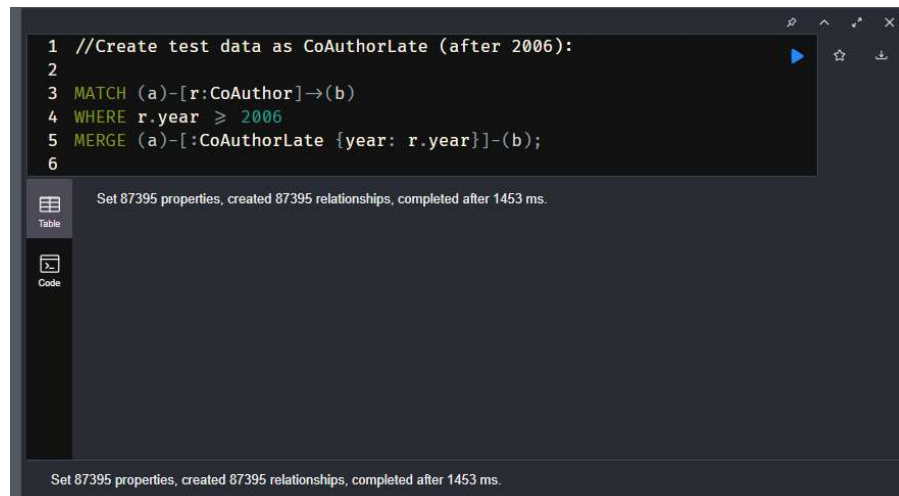
Set 92537 properties, created 92537 relationships, completed after 3620 ms.

Table

Code

Set 92537 properties, created 92537 relationships, completed after 3620 ms.

Figure 27: Creation of train dataset.



```
1 //Create test data as CoAuthorLate (after 2006):
2
3 MATCH (a)-[r:CoAuthor]-(b)
4 WHERE r.year ≥ 2006
5 MERGE (a)-[:CoAuthorLate {year: r.year}]- (b);
6
```

Set 87395 properties, created 87395 relationships, completed after 1453 ms.

Table

Code

Set 87395 properties, created 87395 relationships, completed after 1453 ms.

Figure 28: Creation of test dataset.

The results of the creations above can be visualized as a graph network with Neo4j and can be seen below:

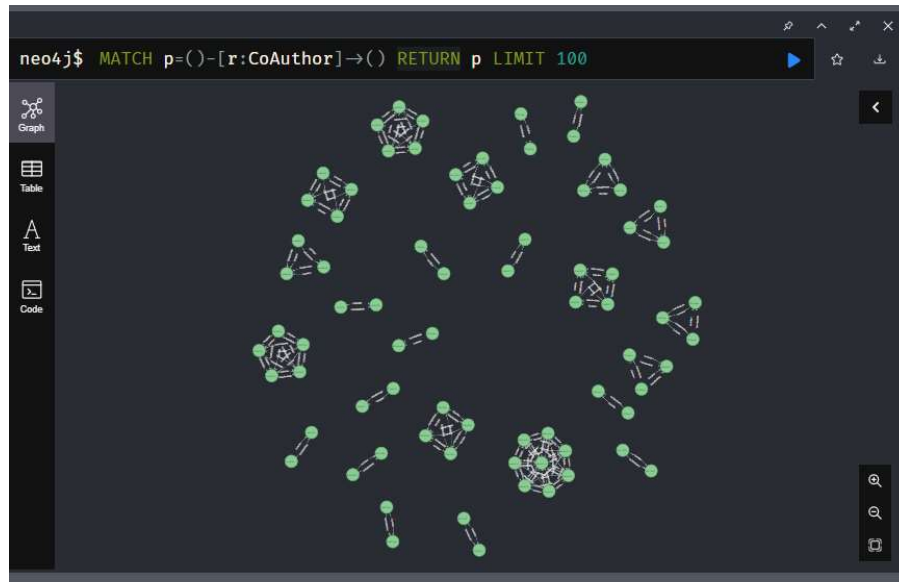


Figure 29: CoAuthor relation alongside CoAuthorEarly and CoAuthorLate.

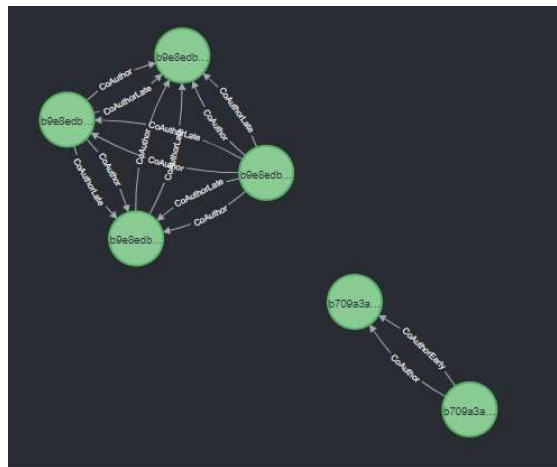


Figure 30: Zoom in to show relations.

3.2 Running Link Prediction algorithms

At last we are ready to run our Link Prediction algorithms for the specific authors that are given to us. We also have to calculate evaluation scores:

True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN), Accuracy, Precision and Recall metrics for each author. Finally we calculate Adoption Rate.

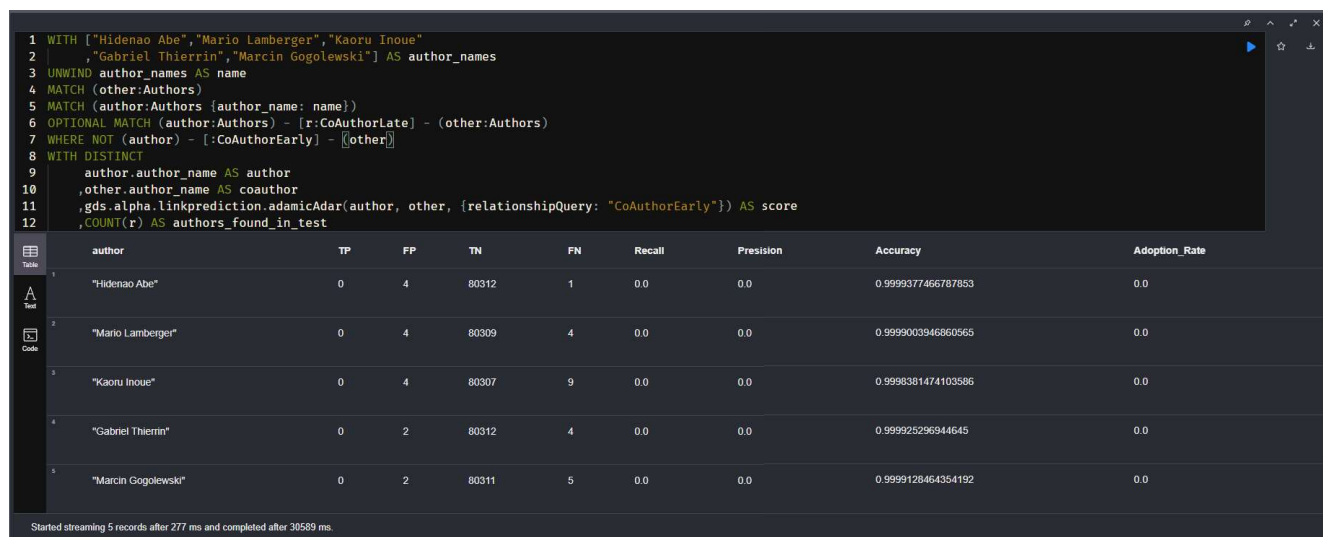
Author Names
Hidenao Abe
Mario Lamberger
Gabriel Thierrin
Marcin Gogolewski
Kaoru Inoue

Table 1: Author names for which we will run Link Predictions.

The queries we ran for the procedure above can be seen below. We ran the queries three times for the 3 different algorithmic approaches (Adamic Adar, Common Neighbors, Preferential Attachment) and printed the results of the authors in question. We also calculated the scores that were requested and we added some **CASE WHEN** commands to make sure we do not divide with zero (safe divide feature in SQL).

Note that due to Neo4j window showing only 12 rows, only a fragment of the cypher code is available in the image attached (the whole query is almost 30 lines). Please find the whole code in our txt file:

Adamic Adar



```

1 WITH ["Hidenao Abe","Mario Lamberger","Kaoru Inoue"
2      ,"Gabriel Thierrin","Marcin Gogolewski"] AS author_names
3 UNWIND author_names AS name
4 MATCH (other:Authors)
5 MATCH (author:Authors {author_name: name})
6 OPTIONAL MATCH (author:Authors) - [r:CoAuthorLate] - (other:Authors)
7 WHERE NOT (author) - [:CoAuthorEarly] - ([other])
8 WITH DISTINCT
9     author.author_name AS author
10    ,other.author_name AS coauthor
11    ,gds.alpha.linkprediction.adamicAdar(author, other, {relationshipQuery: "CoAuthorEarly"}) AS score
12    ,COUNT(r) AS authors_found_in_test

```

	author	TP	FP	TN	FN	Recall	Precision	Accuracy	Adoption_Rate
1	"Hidenao Abe"	0	4	80312	1	0.0	0.0	0.9999377466787853	0.0
2	"Mario Lamberger"	0	4	80309	4	0.0	0.0	0.9999003946860565	0.0
3	"Kaoru Inoue"	0	4	80307	9	0.0	0.0	0.9998381474103586	0.0
4	"Gabriel Thierrin"	0	2	80312	4	0.0	0.0	0.999925296944645	0.0
5	"Marcin Gogolewski"	0	2	80311	5	0.0	0.0	0.9999128464354192	0.0

Started streaming 5 records after 277 ms and completed after 30589 ms.

Figure 31: Link Prediction results for Adamic Adar implementation

Common Neighbors

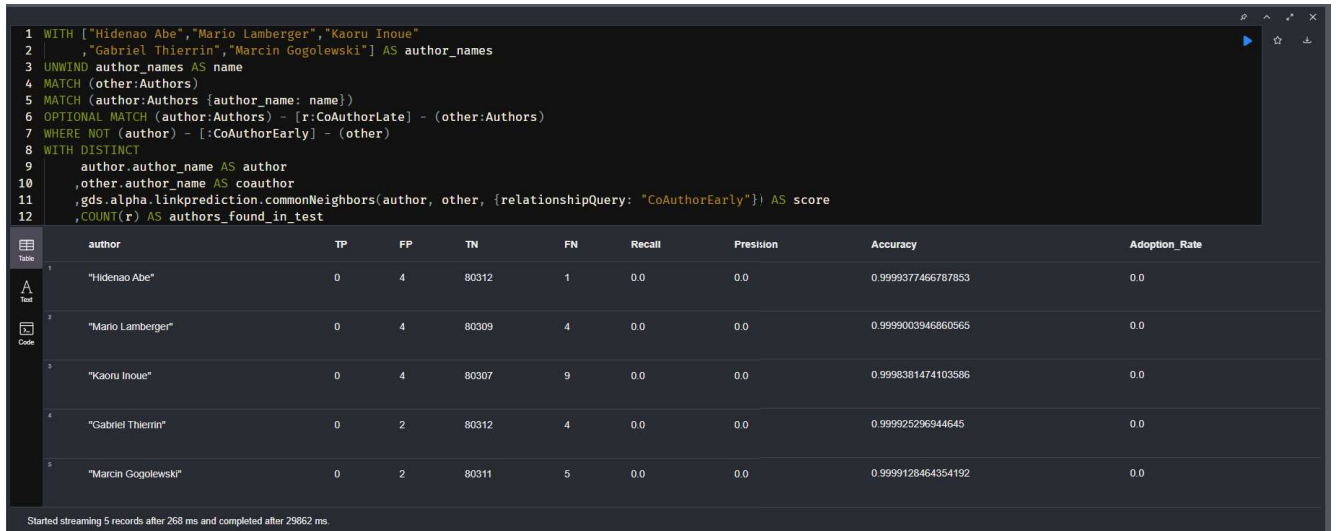


Figure 32: Link Prediction results for Common Neighbors implementation

Preferential Attachment

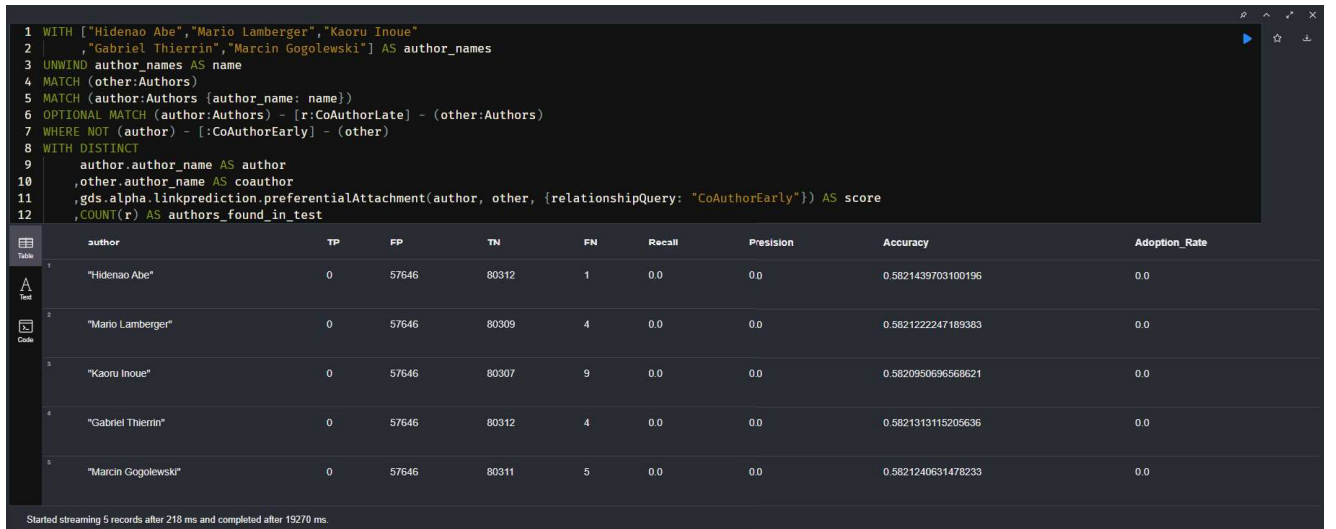


Figure 33: Link Prediction results for Preferential Attachment implementation

4 Results - conclusion

After implementing the three algorithmic approaches, we understand that Common Neighbors and Adamic Adar give similar scores, while the Preferential Attachment seems to have some differences. The first two models return better results than Preferential Attachment.

Checking the data manually from our csv dataset we see that the results might be reasonable, as we see that these specific Authors take part in a few "different" co-authoring schemes. That means that they usually co-author with more or less the same authors.

Diving deeper in the three algorithms we used, we can find more reasoning behind these results. Preferential Attachment works behind the intuition that nodes with lots of relationships will gain more relationships. With that in mind and the above manual check we can understand the low accuracy results.

Adamic Adar and Common Neighbors are "close neighbor - algorithms". Adamic Adar builds the common neighbors, but rather than just counting those neighbors, it computes the sum of the inverse log of the degree of each of the neighbors. The degree of a node is the number of neighbors it has. Lastly Common Neighbors work capturing the notion that nodes with common related nodes could be connected eventually.