Neo4j Graph Database

ASSIGNMENT 2

DATA MINING
MSc in Business Analytics Part Time (2022-2024)
Athens University of Economics and Business
Professor: Y. Kotidis
Assignment's Assistant: I. Filippidou
02/07/2023

Vaidomarkakis Panagiotis | p2822203
Gioupai Katerina | p2822208

Table of Contents

Introduction

Graph databases have gained significant popularity in recent years due to their ability to efficiently store and query highly connected data. One prominent graph database is Neo4j, known for its robust features and scalability. In this assignment, we are presented with the task of working with a subset of the high energy physics theory citation network and leveraging Neo4j to model and analyze the dataset.

The provided dataset encompasses crucial components of the citation network, including articles, authors, journals, and citations between articles. With a total of 29,555 articles, 15,420 authors, 836 journals, and 352,807 citations, the dataset offers a rich source of information for understanding the relationships and dynamics within the high energy physics theory domain. The dataset is conveniently available for download in CSV format, allowing for easy access and utilization.

To effectively represent and analyze the dataset, we are required to model the data as a property graph using Neo4j. Property graphs are composed of nodes, representing entities, and edges, representing the relationships between these entities. By assigning appropriate labels, types, and properties to nodes and edges, we can capture the essential attributes of each entity and relationship within the citation network.

Data Import

EXPLANATION

The first step in the modeling process is to thoroughly examine the details of each dataset file. We start with the "ArticleNodes.csv" file, which contains information about articles, including their unique identifiers, titles, publication years, associated journals, and abstracts. These attributes will be incorporated as properties of the article nodes in the graph. The "AuthorNodes.csv" file provides the article IDs and corresponding author names. This information enables us to establish connections between authors and articles through edges. Lastly, the "Citations.csv" file specifies the citations between articles, allowing us to create edges that represent these relationships in the graph.

When designing the property graph model, it is important to ensure that we include only the necessary attributes for each node and edge type. This eliminates redundant information and enhances the overall clarity and efficiency of the graph. By avoiding unnecessary repetitions of elements, we can maintain a concise and well-structured representation of the citation network.

Having formulated the property graph model, the next step is to import the dataset into Neo4j. Neo4j provides various options for data loading, including the Neo4j browser, the Neo4j import tool, and programming languages supported by Neo4j. By leveraging these tools, we can efficiently load the article nodes, author nodes, citation edges, and their associated attributes into the Neo4j graph database. To optimize the loading process and improve query response times, it is advisable to create appropriate indexes on the model properties. Indexing allows for faster data retrieval and enhances the performance of subsequent queries on the dataset.

Through this assignment, we aim to gain practical experience in working with Neo4j and understanding the intricacies of graph databases. By modeling and analyzing the high energy physics theory citation network, we can explore the relationships between articles, authors, and journals, and gain valuable insights into the dynamics of scholarly research within this domain. The use of Neo4j as a powerful tool for graph analysis equips us with the ability to navigate and extract meaningful information from complex, interconnected datasets, furthering our understanding of the underlying patterns and structures present in the high energy physics theory domain.

DATA MODEL

The available dataset comprises several CSV files that provide valuable information for constructing the graph model:

- ArticleNodes.csv: This file includes details such as the article ID, title, publication year, journal name, and abstract for each article.
- AuthorNodes.csv: This file contains the article IDs and the corresponding author name(s) for each article.
- Citations.csv: This file captures the citation relationships between articles.

We will create the following entities as nodes to be created within the graph model:

- Article: This node represents an article and has the following attributes:
    - Article ID
    - Title
    - Abstract
    - Year
- Author: This node represents an author and has the following attribute:
    - Author Name
- Journal: This node represents a journal and has the following attribute:
    - Name

The decision to model the journal as a separate node, rather than an attribute of the article node, is based on the understanding that the journal in which an article is published is not an inherent characteristic of the article itself. Instead, it is a relationship that the article has with the journal (via a "published" relationship). Therefore, the journal can be considered a distinct entity associated with articles through a specific kind of relationship in the model.

After defining the node entities, we proceeded to model the appropriate relationships to achieve a comprehensive graph model. The following relationships were established:

- CITES: This relationship connects two article nodes and represents the citation from one article to another. It is a directed relationship, originating from the article citing another article.
- AUTHORED: This relationship is incoming to the article node from the author node and signifies that the author(s) wrote the article.

- PUBLISHED IN: This relationship is incoming to the journal node from the article node and indicates that the article was published in the respective journal.

VISUALIZATION OF THE MODEL

Figure 1 presents an illustrative example of the graph model that will be created:

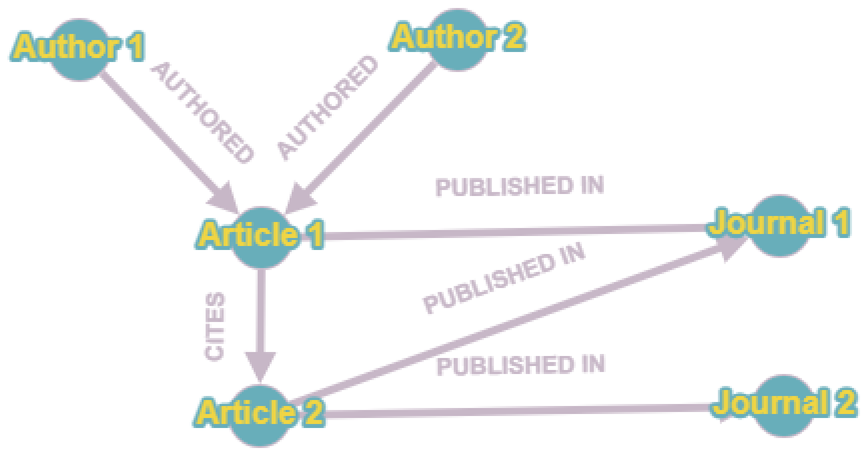


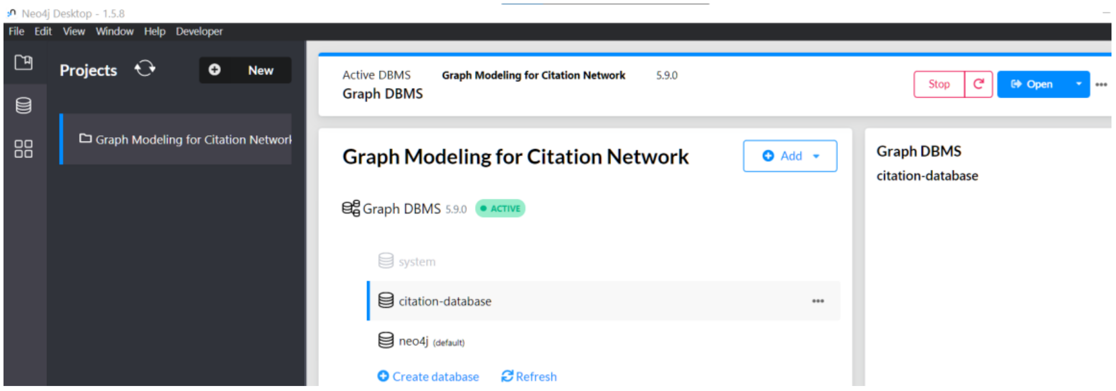
Figure 1

The example diagram showcases the interconnected nodes representing articles, authors, and journals, along with the relationships between them. The "CITES" relationship demonstrates the directed citation connections between articles, while the "AUTHORED" relationship indicates the authorship association. Lastly, the "PUBLISHED IN" relationship signifies the publication of articles in specific journals.

This graph model provides a visual representation of the dataset, allowing for a comprehensive understanding of the relationships and connections between articles, authors, and journals within the high energy physics theory citation network.

Database Initialization

First of all, we have downloaded Neo4j Desktop – 1.5.8 in order to create the database and execute the queries. Then, we create the cached database 'Graph Modeling for Citation Network' and we start the execution of it. Before going any further, we need to put our csv files in the import folder of the database in order to be easier to access them later on. After that, we created the 'citation-database' using Graph DBMS Community Edition 5.9.0. Below is a screenshot until these steps:



Now, we need to open the Active DBMS (Graph Modeling for Citation Network) in Neo4j Browser in order to create the graph model and execute the Cypher Queries.

To create the database based on the schema described in the Data Modeling section, the following steps were undertaken:

- Constraint Creation: Two constraints were created to ensure data integrity and improve query performance:

    1. An Article ID constraint was established using the query:

```
CREATE CONSTRAINT artIdConstr for (a:Article) REQUIRE a.articleId  IS
UNIQUE
```

This constraint not only enforces the uniqueness of article IDs but also automatically creates an index, which is beneficial for efficient querying based on the article ID.

    2. An Author Name index was created using the query:

```
CREATE INDEX FOR (a:Author) ON (a.name)
```

This index improves the matching of author nodes based on their names and enhances data retrieval and query performance.

- Data Loading and Node/Edge Creation:
  The dataset was loaded into the database, and the necessary nodes and edges were created using the following queries:
  1. Article Node Creation:

```
LOAD CSV FROM "file:///ArticleNodes.csv" AS row
CREATE(a:Article{
articleId:toInteger(row[0]),
title:row[1],
year:toInteger(row[2]),
abstract:row[4]})
```

  2. Journal Node Creation:

```
LOAD CSV FROM "file:///ArticleNodes.csv" AS row
WITH row WHERE row[3] IS NOT NULL
MERGE (j:Journal{journalName:row[3]})
```

  The merge command was used to avoid duplicate journal nodes, and rows with null journal names were filtered out.

  3. Creation of Relationships between Article Nodes and Journal Nodes:

```
LOAD CSV FROM "file:///ArticleNodes.csv" AS row
WITH row WHERE row[3] IS NOT NULL AND row[3] <> ''
MATCH (a:Article {articleId: toInteger(row[0])})
MATCH (j:Journal {journalName: row[3]})
MERGE (a)-[:PUBLISHEDIN]->(j)
```

  Match commands were utilized to find each pair of article and journal nodes, and the merge command ensured that duplicate relationships were not created.

  4. Creation of Citation Relationships between Articles:

```
LOAD CSV FROM "file:///Citations.csv" AS row FIELDTERMINATOR '\t'
MATCH (m:Article {articleId: toInteger(row[0])}),
(n:Article {articleId: toInteger(row[1])})
MERGE (m)-[:CITES]->(n)
```

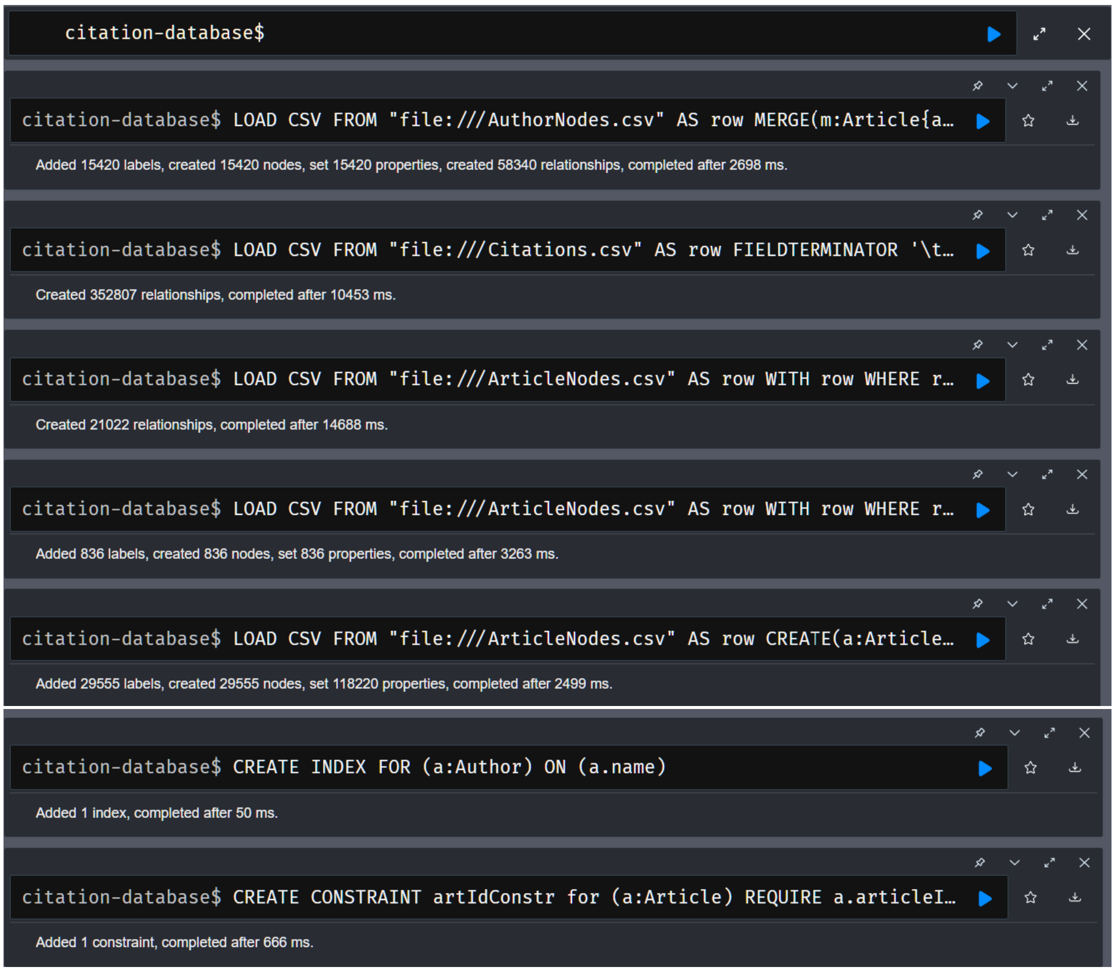
  For each line in the Citations.csv file, the corresponding article nodes participating in the citation relationship were matched based on their article IDs. The merge command prevented the creation of duplicate relationships, and the direction of the relationship was from the citing article to the cited article.

5. Creation of Author Nodes and the Relationship between Authors and Articles:

```
LOAD CSV FROM "file:///AuthorNodes.csv" AS row
MERGE(m:Article{articleId:toInteger(row[0])})
MERGE(a:Author{name:row[1]})
MERGE(a)-[:AUTHORED]->(m)
```

This query created author nodes and established the "AUTHORED" relationship between authors and articles. The merge command ensured that duplicate nodes and relationships were avoided.

Below is a screenshot of the current situation:

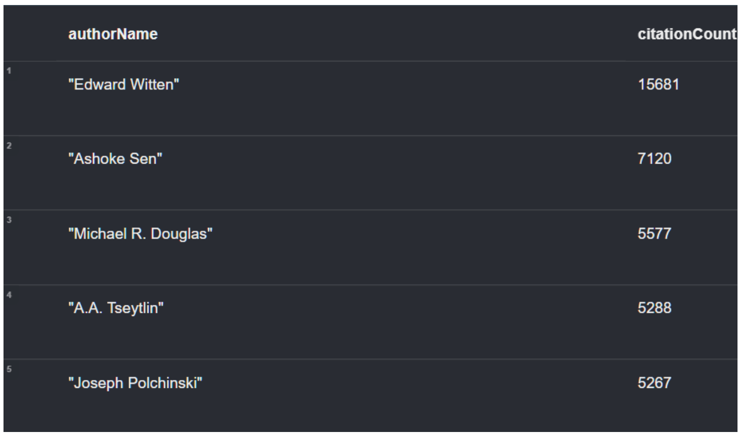


```
citation-database$

citation-database$ LOAD CSV FROM "file:///AuthorNodes.csv" AS row MERGE(m:Article{a…
Added 15420 labels, created 15420 nodes, set 15420 properties, created 58340 relationships, completed after 2698 ms.

citation-database$ LOAD CSV FROM "file:///Citations.csv" AS row FIELDTERMINATOR '\t…
Created 352807 relationships, completed after 10453 ms.

citation-database$ LOAD CSV FROM "file:///ArticleNodes.csv" AS row WITH row WHERE r…
Created 21022 relationships, completed after 14688 ms.

citation-database$ LOAD CSV FROM "file:///ArticleNodes.csv" AS row WITH row WHERE r…
Added 836 labels, created 836 nodes, set 836 properties, completed after 3263 ms.

citation-database$ LOAD CSV FROM "file:///ArticleNodes.csv" AS row CREATE(a:Article…
Added 29555 labels, created 29555 nodes, set 118220 properties, completed after 2499 ms.

citation-database$ CREATE INDEX FOR (a:Author) ON (a.name)
Added 1 index, completed after 50 ms.

citation-database$ CREATE CONSTRAINT artIdConstr for (a:Article) REQUIRE a.articleI…
Added 1 constraint, completed after 666 ms.
```

Querying the database

1. Which are the top 5 authors with the most citations (from other papers). Return author names and number of citations.

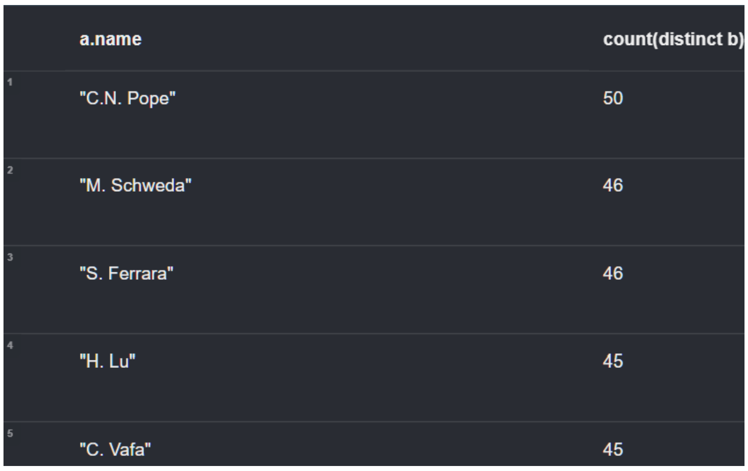
```
MATCH (a:Author)-[:AUTHORED]->(:Article)<-[:CITES]-(:Article)
WITH a, COUNT(*) AS citationCount
RETURN a.name AS authorName, citationCount
ORDER BY citationCount DESC
LIMIT 5;
```



authorName	citationCount
"Edward Witten"	15681
"Ashoke Sen"	7120
"Michael R. Douglas"	5577
"A.A. Tseytlin"	5288
"Joseph Polchinski"	5267

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

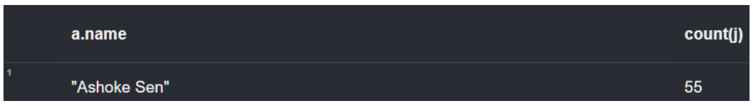
```
MATCH(a:Author)-[:AUTHORED]->(:Article)<-[:AUTHORED]-(b:Author)
WHERE a.name<>b.name
RETURN a.name, count(distinct b) AS
ORDER BY count(distinct b) DESC
LIMIT 5;
```



a.name	count(distinct b)
"C.N. Pope"	50
"M. Schweda"	46
"S. Ferrara"	46
"H. Lu"	45
"C. Vafa"	45

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

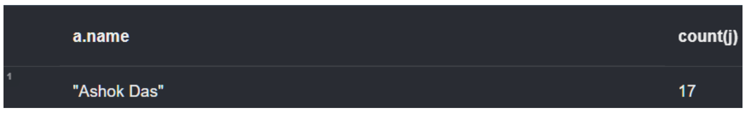
```
MATCH (a:Author)-[:AUTHORED]->(j:Article)
WHERE NOT EXISTS {
  MATCH (a:Author)-[:AUTHORED]->(j:Article)<-[:AUTHORED]-
(b:Author)
  WHERE a.name <> b.name}
RETURN a.name, count(j)
ORDER BY count(j) DESC
LIMIT 1;
```



```
   a.name                                              count(j)
1  "Ashoke Sen"                                        55
```

4. Which author published the most papers in 2001? Return author name and number of papers.

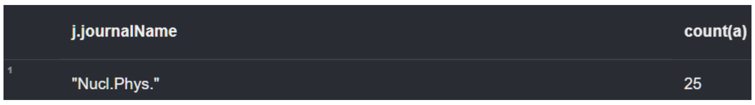
```
MATCH(a:Author)
MATCH(j:Article{year:2001})
MATCH(a:Author)-[:AUTHORED]->(j:Article)
RETURN a.name, count(j)
ORDER BY count(j) DESC
LIMIT 1;
```



```
   a.name                                              count(j)
1  "Ashok Das"                                         17
```

5. Which is the journal with the most papers about "gravity" (derived only from the paper title) in 1998. Return name of journal and number of papers.

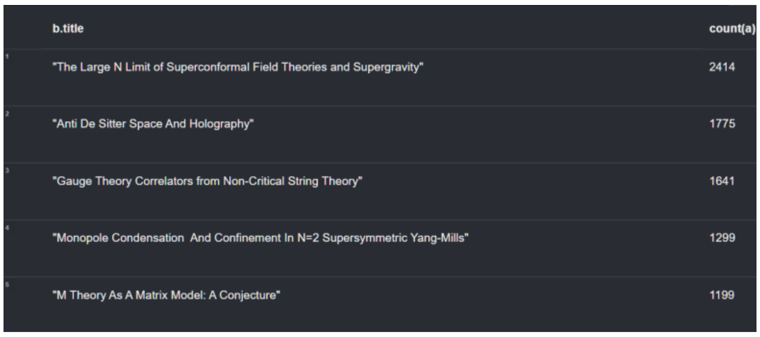
```
MATCH(a:Article)
MATCH(j:Journal)
WHERE a.year=1998 AND a.title CONTAINS 'gravity'
MATCH(a:Article)-[:PUBLISHEDIN]->(j:Journal)
RETURN j.journalName, count(a)
ORDER BY count(a) DESC
LIMIT 1;
```



```
   j.journalName                                       count(a)
1  "Nucl.Phys."                                        25
```

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

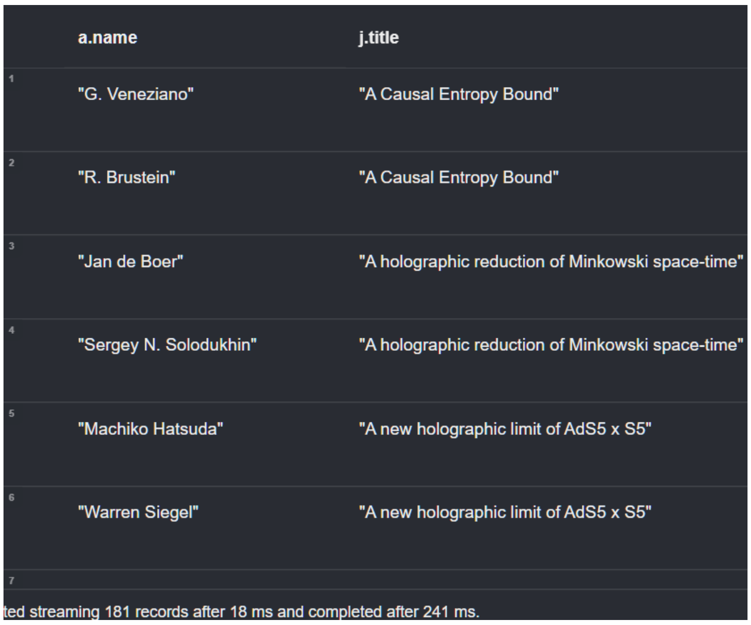
```
MATCH(a:Article)-[:CITES]->(b:Article)
RETURN b.title, count(a)
ORDER BY count(a) DESC
LIMIT 5;
```



	b.title	count(a)
1	"The Large N Limit of Superconformal Field Theories and Supergravity"	2414
2	"Anti De Sitter Space And Holography"	1775
3	"Gauge Theory Correlators from Non-Critical String Theory"	1641
4	"Monopole Condensation And Confinement In N=2 Supersymmetric Yang-Mills"	1299
5	"M Theory As A Matrix Model: A Conjecture"	1199

7. Which were the papers that use "holography" and "anti de sitter" (derived only from the paper abstract). Return authors and title.

```
MATCH(a:Author)-[:AUTHORED]->(j:Article)
WHERE j.abstract CONTAINS 'holography' OR j.abstract CONTAINS 'anti d
e sitter'
RETURN a.name, j.title
ORDER BY j.title, a.name;
```



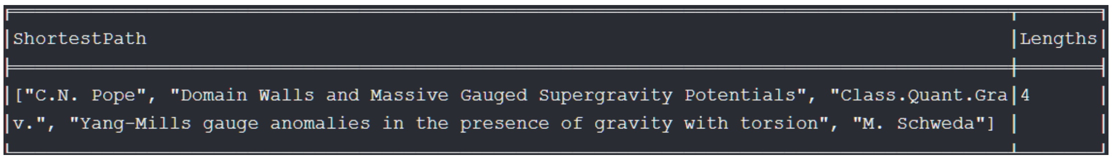
	a.name	j.title
1	"G. Veneziano"	"A Causal Entropy Bound"
2	"R. Brustein"	"A Causal Entropy Bound"
3	"Jan de Boer"	"A holographic reduction of Minkowski space-time"
4	"Sergey N. Solodukhin"	"A holographic reduction of Minkowski space-time"
5	"Machiko Hatsuda"	"A new holographic limit of AdS5 x S5"
6	"Warren Siegel"	"A new holographic limit of AdS5 x S5"
7		

ted streaming 181 records after 18 ms and completed after 241 ms.

(Not all 181 records appear here)

8. Find the shortest path between ‘C.N. Pope’ and ‘M. Schweda’ authors (use any type of edges). Return the path and the length of the path. Comment about the type of nodes and edges of the path.

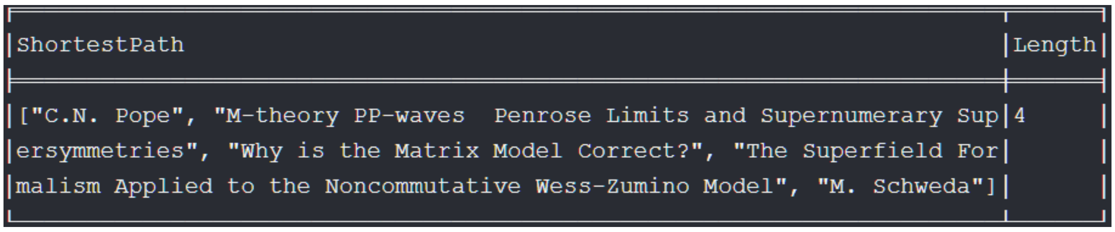
```
MATCH p=shortestPath((a:Author{name:'C.N. Pope'})-[*]-
(f:Author{name:'M. Schweda'}))
RETURN [n in nodes(p) |
 CASE
   WHEN n:Article THEN n.title
   WHEN n:Author THEN n.name
   WHEN n:Journal THEN n.journalName
   ELSE "Unknown"
 END
] AS ShortestPath, length(p) as Lengths
```



```
|ShortestPath                                                                       |Lengths|
|["C.N. Pope", "Domain Walls and Massive Gauged Supergravity Potentials", "Class.Quant.Gra|4      |
|v.", "Yang-Mills gauge anomalies in the presence of gravity with torsion", "M. Schweda"] |       |
```

9. Run again the previous query (8) but now use only edges between authors and papers. Comment about the type of nodes and edges of the path. Compare the results with query 8.

```
MATCH p=shortestPath((a:Author{name:'C.N. Pope'})-[*]-
(f:Author{name:'M. Schweda'}))
WHERE all(node in nodes(p) WHERE node:Author OR node:Article)
RETURN [n in nodes(p) |
 CASE
   WHEN n:Article THEN n.title
   WHEN n:Author THEN n.name
 END
] AS ShortestPath, length(p) as Length
```



```
|ShortestPath                                                    |Length|
|["C.N. Pope", "M-theory PP-waves  Penrose Limits and Supernumerary Sup|4     |
|ersymmetries", "Why is the Matrix Model Correct?", "The Superfield For|      |
|malism Applied to the Noncommutative Wess-Zumino Model", "M. Schweda"]|      |
```

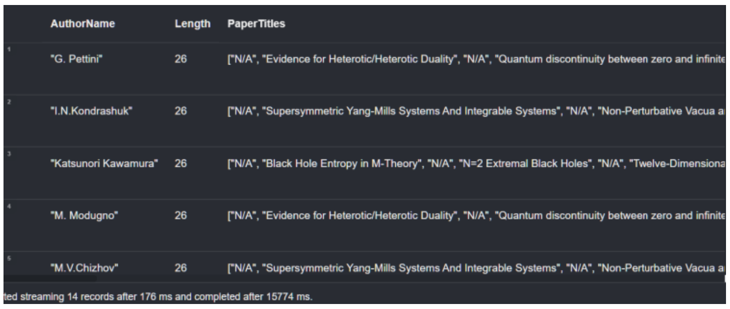
10. Find all authors with shortest path lengths > 25 from author 'Edward Witten'. The shortest paths will be calculated only on edges between authors and articles. Return author name, the length and the paper titles for each path.

(When we tried to filter the query based on the shortest path's length, we encountered a problem. The issue was that we had to check all the paths and select only the ones with a length greater than 25. This created a difficulty because it would require a lot of time and resources to evaluate all the paths.

To overcome this problem, we took a different approach. Instead of filtering the paths based on length right away, we decided to retrieve all the paths without any length restrictions. Then, using a "WITH" clause, we further refined the results to only include the paths that had a length of more than 25. It's similar to having a step-by-step process, where we first gather all the relevant information and then narrow down the results based on the desired length criterion.

By doing this, we avoided the challenge of evaluating the length of all paths during the initial query, which would have been time-consuming and resource-intensive. Now, the query retrieves all the paths and then filters them based on the desired length of over 25, allowing us to achieve our goal without the initial difficulty.)

```
MATCH (a:Author {name:'Edward Witten'})
MATCH p=shortestPath((a)-[:AUTHORED*]-(m:Author))
WHERE a.name <> m.name AND ALL(node IN nodes(p) WHERE (node:Aut
hor OR node:Article))
WITH m.name AS AuthorName, length(p) AS Length, [n IN nodes(p) |
  CASE
    WHEN n:Article THEN n.title
    WHEN n:Author THEN 'N/A'
  END
] AS PaperTitles
WHERE Length > 25
RETURN AuthorName, Length, PaperTitles
ORDER BY Length, AuthorName, PaperTitles;
```



	AuthorName	Length	PaperTitles
1	"G. Pettini"	26	["N/A", "Evidence for Heterotic/Heterotic Duality", "N/A", "Quantum discontinuity between zero and infinite
2	"I.N.Kondrashuk"	26	["N/A", "Supersymmetric Yang-Mills Systems And Integrable Systems", "N/A", "Non-Perturbative Vacua a
3	"Katsunori Kawamura"	26	["N/A", "Black Hole Entropy in M-Theory", "N/A", "N=2 Extremal Black Holes", "N/A", "Twelve-Dimensiona
4	"M. Modugno"	26	["N/A", "Evidence for Heterotic/Heterotic Duality", "N/A", "Quantum discontinuity between zero and infinite
5	"M.V.Chizhov"	26	["N/A", "Supersymmetric Yang-Mills Systems And Integrable Systems", "N/A", "Non-Perturbative Vacua a

```
ted streaming 14 records after 176 ms and completed after 15774 ms.
```

(Not all 14 records appear here)