M A S T E R   I N   B U S I N E S S   A N A L Y T I C S

MINING BIG DATASETS

2ND ASSIGNMENT
NEO4J GRAPH DATABASE

C H R I S T O S   K A L L A R A S ,   P 2 8 2 2 0 0 9
A T H E N S   2 0 2 1

C O N T E N T S

1. GRAPH MODEL

The datasets consist of 3 files:

- ArticleNodes.csv: Contains info about Article nodes.
- AuthorNodes.csv: Contains article id and the name of the author(s).
- Citations.csv: Contains info about citations between articles.

Based on that we created a directed graph that consists of 2 types of nodes and 2 types of edges. Specifically, we used nodes to represent Articles and Authors. Each article consists of an id, title, journal and abstract. Each author consists of the id of the article he wrote and his/her name. The writing of an article was represented by a directed edge, the same as the citation between articles. A property diagram of the graph before importing the datasets into Neo4j can be seen in Figure 1 .

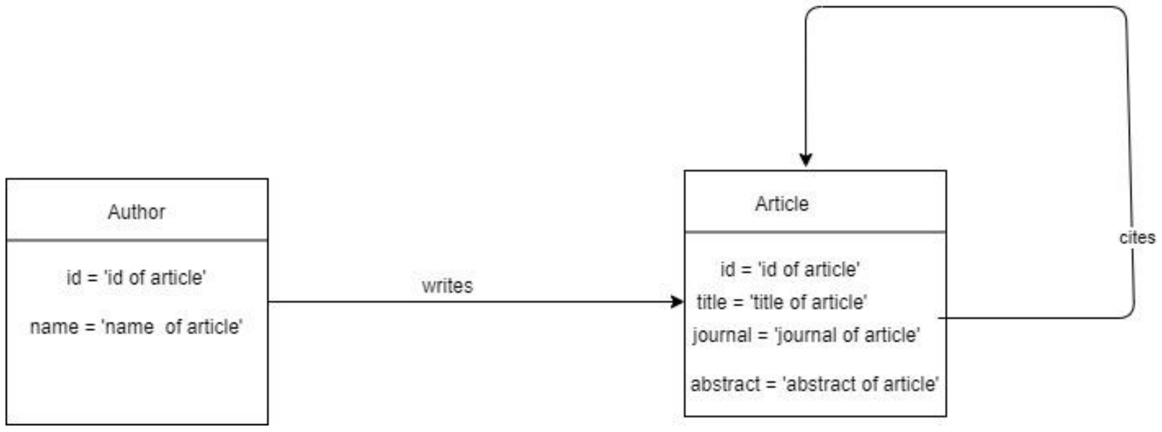

Author
id = 'id of article'
name = 'name of article'
writes
Article
id = 'id of article'
title = 'title of article'
journal = 'journal of article'
abstract = 'abstract of article'
cites

Figure 1: Property Graph diagram before importing into neo4j

2. IMPORT IN DATABASE

First, we created the database as we can see in Figure 2 , and then we stored the 3 given files using the import neo4j tool.

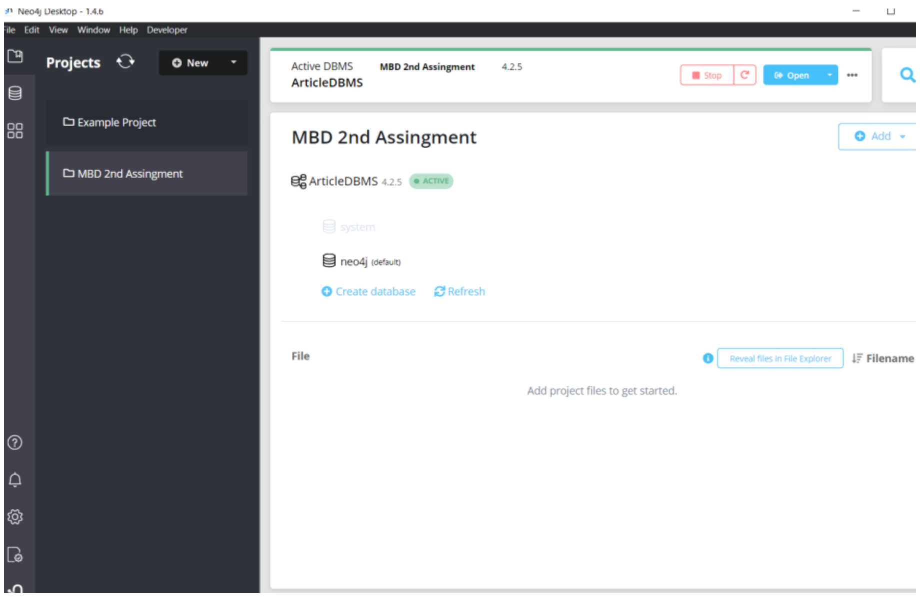


Figure 2: Database created.

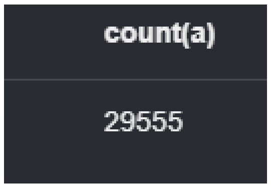
After that we used cypher command to load the files in the database.

First for the Article nodes we created a constraint to the id of the article in order to speed up query execution time and avoid duplicates (using merge command). The command can be seen bellow.

```
CREATE CONSTRAINT ON (a:Article) ASSERT a.articleId IS UNIQUE

LOAD CSV FROM 'file:///ArticleNodes.csv' AS row with toInteger(row[0]) as articleId,row[1] as title,toInteger(row[2]) as year,row[3] as journal,row[4] as abstract MERGE (a:Article {articleId:articleId}) SET a.title = title,a.year = year,a.journal = journal,a.abstract = abstract return count(a)
```

The return result (number of inserted nodes) are the following:



count(a)
29555

The graph at this moment can be seen in Figure 3.

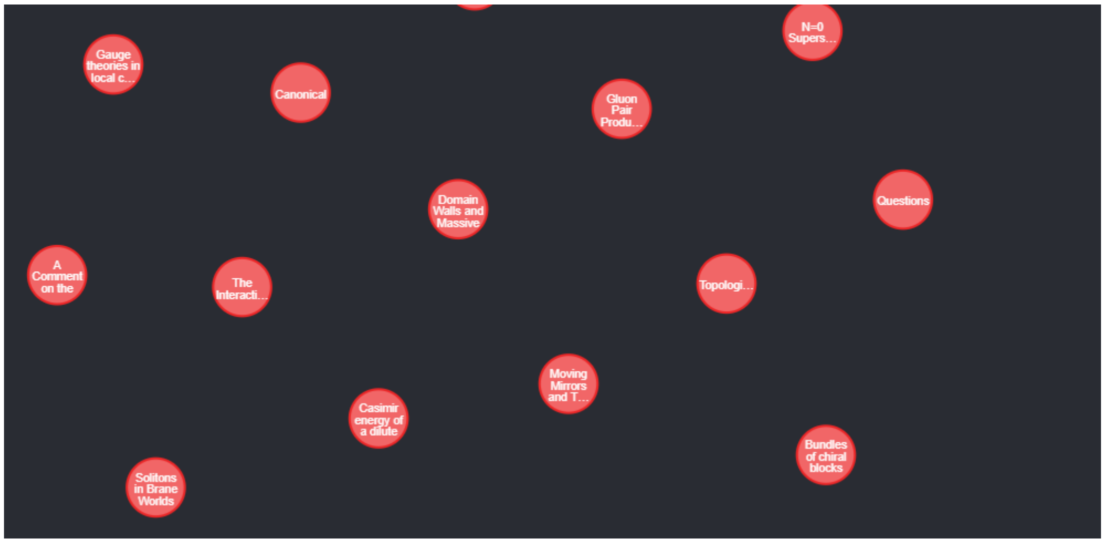


Figure 3: Article graph

Then for Authors we execute the following query. We do not use a constraint here since we can have multiple ids and names.

```
LOAD CSV FROM 'file:///AuthorNodes.csv' AS row with toInteger(row[0]) as author_articleId, row[1] as author_name CREATE(b:Author {author_articleId:author_articleId, author_name:author_name})
```

The graph for the author nodes can be seen in Figure 4.

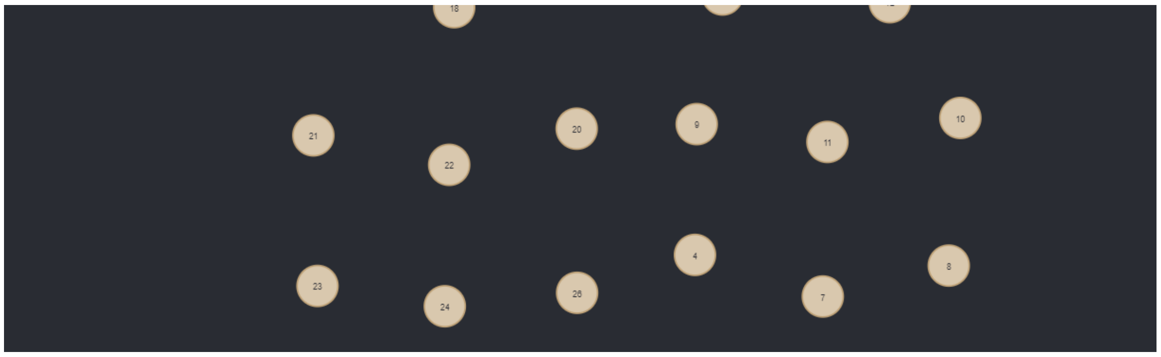


Figure 4: Author Graph

Then we created the relationships. We have 2 relationships :

1. From article to article nodes, that is the CITES edge
2. From author node to article node, that is the WROTE edge

The command for the WROTE edge is the following:

```
MATCH
  (a:Author),
  (b:Article)
WHERE a.author_articleId = b.articleId
CREATE (a)-[rel:WROTE]->(b)
RETURN count(rel)
```

The graph with this relationship can be seen in Figure 5 .

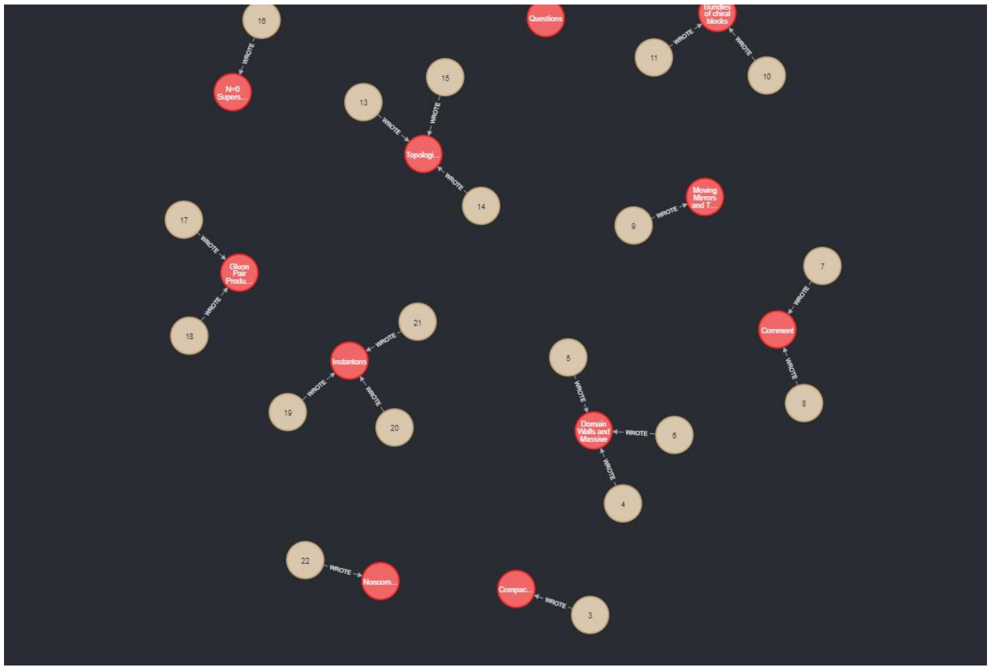


Figure 5: Graph with WROTE relationship

Finally, we created the second relationship, relationship CITES from an article to article.

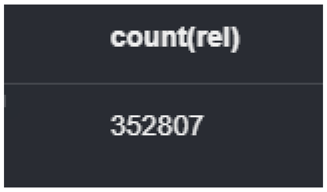
For that we tried to load the citation file using space as delimiter (bellow querry), but the query never finished execution.

```
:auto USING PERIODIC COMMIT
LOAD CSV FROM "file:/Citations.csv" AS row FIELDTERMINATOR '\t'
MATCH (articleid1:Article {id: toInteger(row[0])})
MATCH (articleid2:Article {id: toInteger(row[1])})
CREATE (articleid1)-[:Cites]->(articleid2)
```

So, we created a python script to read the Citation file and transform it into a comma separated file called Citation_comma.csv. The script can be seen in *create_citation_file.py* and the cypher command bellow.

```
:auto USING PERIODIC COMMIT
LOAD CSV FROM 'file:///Citations.csv' AS row
WITH toInteger(row[0]) AS firstId, toInteger(row[1]) AS secondId
MATCH (c:Article { articleId:firstId})
MATCH (a:Article {articleId:secondId})
MERGE (c)-[rel:CITES]->(a)
RETURN count(rel);
```

Since importing large amounts of data using LOAD CSV with a single Cypher query may fail due to memory constraints, we to used PERIODIC COMMIT. PERIODIC COMMIT processes specific number of rows at a time, while preventing running out of memory. The return results are the following:



count(rel)
352807

The graph with that relationship can be seen in.

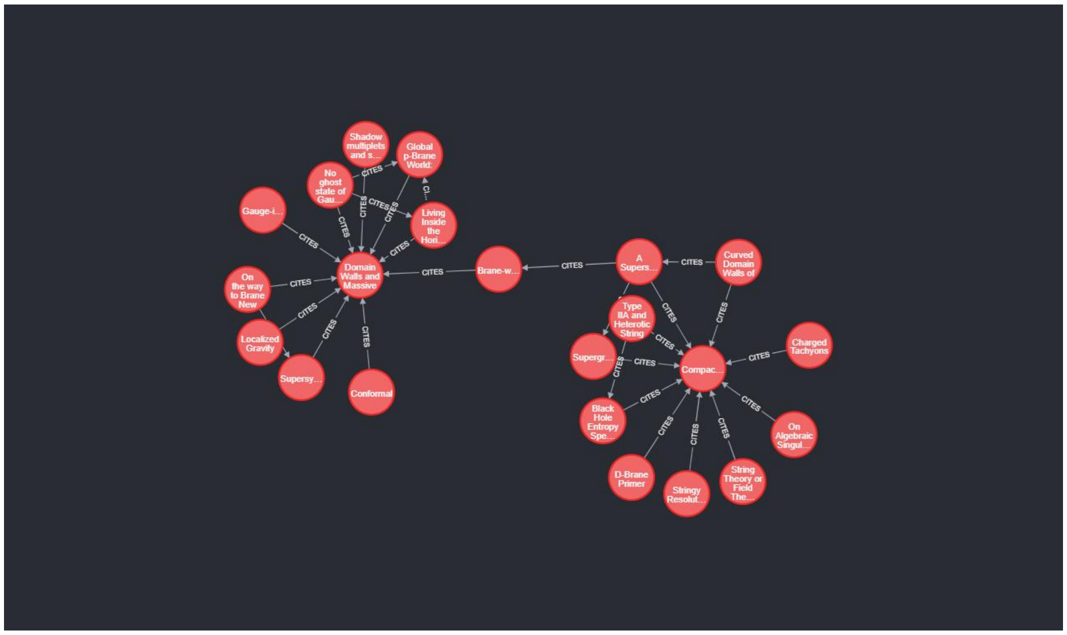
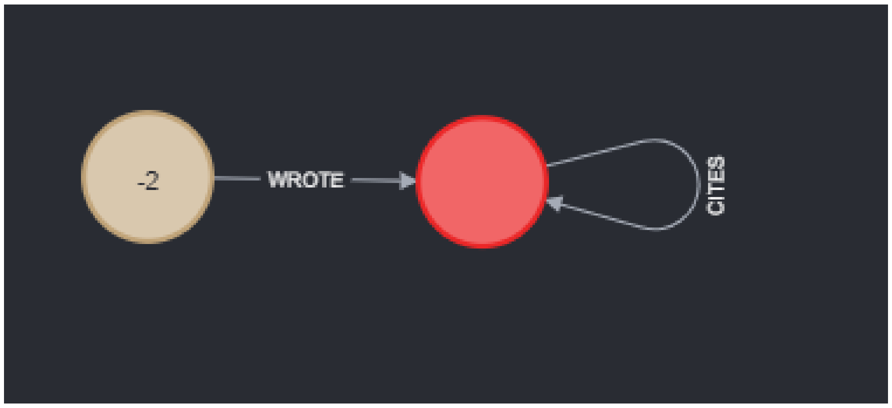


Figure 6: Article graph with CITES relationship.

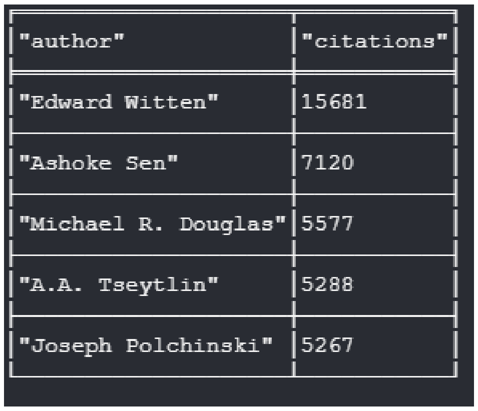
The graph schema as expected is the same as the one we designed and can be seen bellow:



3. QUERIES

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

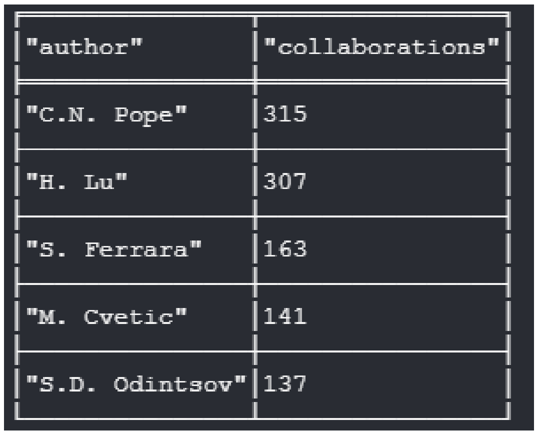
```
MATCH (a:Author) - [rel:WROTE] -> (b:Article) <- [c:CITES] - (n:Article)
      RETURN a.author_name AS author, COUNT(c) AS citations
      ORDER BY citations DESC
      LIMIT 5
```



```
"author"             |"citations"
"Edward Witten"      |15681
"Ashoke Sen"         |7120
"Michael R. Douglas" |5577
"A.A. Tseytlin"      |5288
"Joseph Polchinski"  |5267
```

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

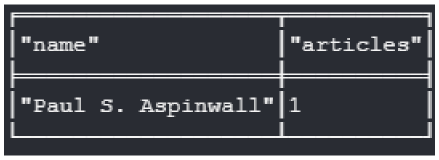
```
      RETURN a.author_name AS author, COUNT(DISTINCT d) AS collaborations
      ORDER BY collaborations DESC
      LIMIT 5
```



```
"author"        |"collaborations"
"C.N. Pope"     |315
"H. Lu"         |307
"S. Ferrara"    |163
"M. Cvetic"     |141
"S.D. Odintsov" |137
```

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

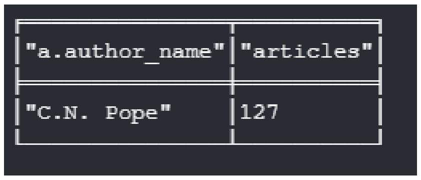
```
MATCH (a:Author) - [rel:WROTE] -> (b:Article)
     OPTIONAL MATCH (collaborator:Author) - [rel1:WROTE] -> (b:Article)
    WITH a, COUNT(b) AS articles, COUNT(DISTINCT collaborator) AS collaborators
     WHERE collaborators = 1
     RETURN a.author_name as name, articles
     ORDER BY articles DESC
     LIMIT 1
```



```
"name"             |"articles"
"Paul S. Aspinwall"|1
```

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

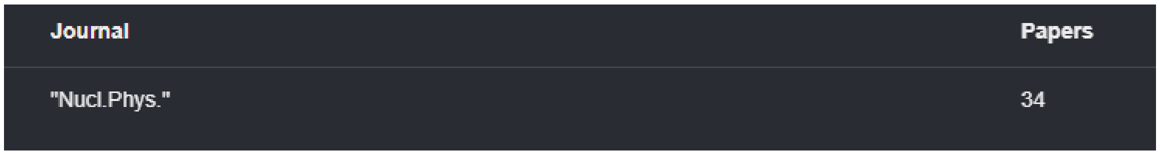
```
MATCH (a:Author) - [:WROTE] -> (b:Article)
RETURN a.author_name, count(b) AS articles
ORDER BY articles DESC
LIMIT 1
```



```
╒═══════════════╤══════════╕
│"a.author_name"│"articles"│
╞═══════════════╪══════════╡
│"C.N. Pope"    │127       │
└───────────────┴──────────┘
```

3.5. 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.

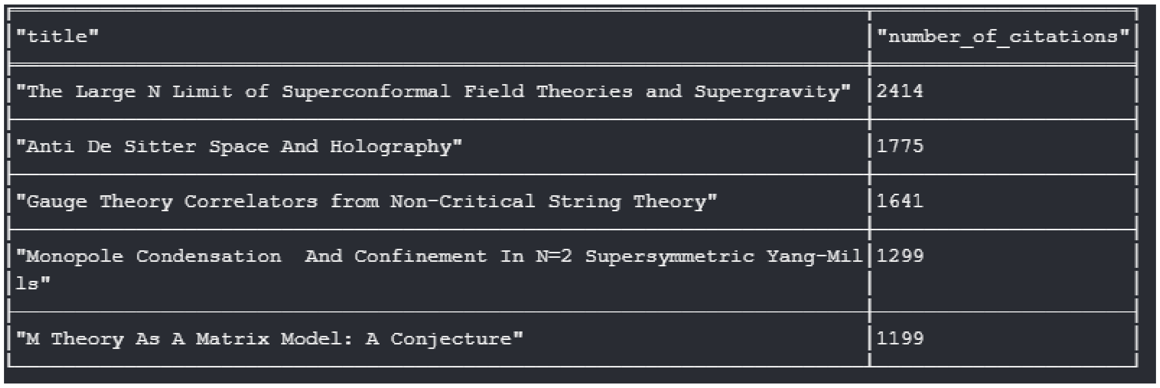
```
MATCH(a:Article) WHERE a.year =1998 AND toLower(a.title) CONTAINS "gravity"
RETURN a.journal as Journal,COUNT(a) AS Papers ORDER BY Papers DESC LIMIT
1
```



Journal	Papers
"Nucl.Phys."	34

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

```
MATCH (a:Article)-[rel:CITES]->(b:Article)
RETURN b.title AS title, COUNT(rel) AS number_of_citations
ORDER BY number_of_citations DESC
LIMIT 5
```



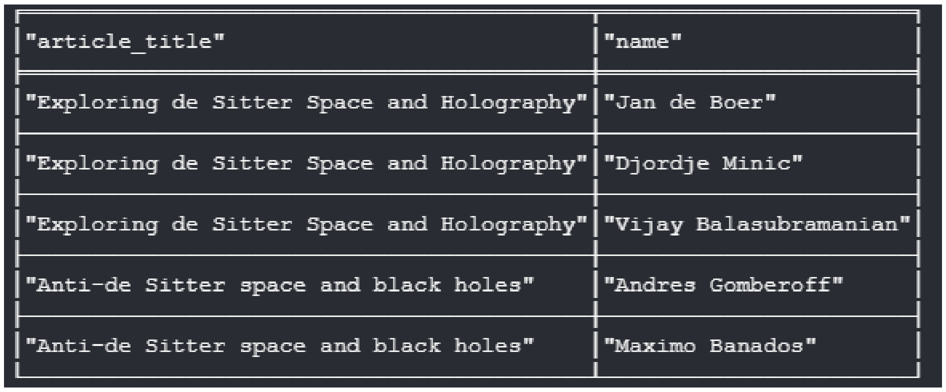
```
╒══════════════════════════════════════════════════════════════════════╤═════════════════════╕
│"title"                                                               │"number_of_citations"│
╞══════════════════════════════════════════════════════════════════════╪═════════════════════╡
│"The Large N Limit of Superconformal Field Theories and Supergravity" │2414                 │
├──────────────────────────────────────────────────────────────────────┼─────────────────────┤
│"Anti De Sitter Space And Holography"                                 │1775                 │
├──────────────────────────────────────────────────────────────────────┼─────────────────────┤
│"Gauge Theory Correlators from Non-Critical String Theory"            │1641                 │
├──────────────────────────────────────────────────────────────────────┼─────────────────────┤
│"Monopole Condensation  And Confinement In N=2 Supersymmetric Yang-Mil│1299                 │
│ls"                                                                   │                     │
├──────────────────────────────────────────────────────────────────────┼─────────────────────┤
│"M Theory As A Matrix Model: A Conjecture"                            │1199                 │
└──────────────────────────────────────────────────────────────────────┴─────────────────────┘
```

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

We will create full text index on the title of the article which will allow us search for the existence of a key phrase in all the titles much more quickly with the benefit of searching for permutations of the phrase.

```
CALL db.index.fulltext.createNodeIndex("AbstractTitle",
["Article"], ["title", "abstract"])
```

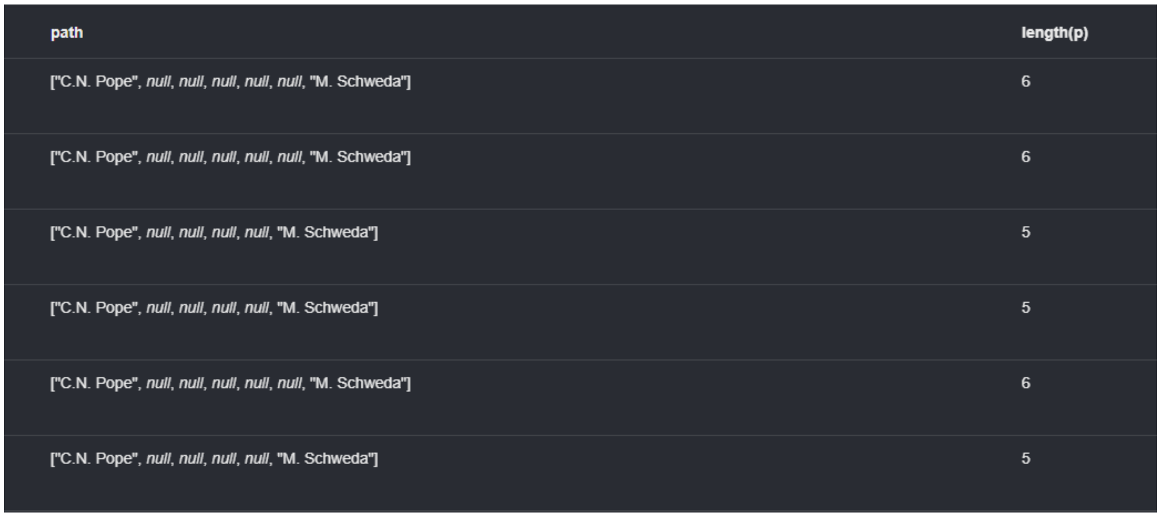
```
CALL db.index.fulltext.queryNodes("AbstractTitle", "holography anti de sitter")
    YIELD node AS article
    MATCH (author:Author)-[:WROTE]->(article)
    RETURN article.title AS article_title, author.author_name as name
    LIMIT 5
```



```
"article_title"                                  "name"
"Exploring de Sitter Space and Holography"       "Jan de Boer"
"Exploring de Sitter Space and Holography"       "Djordje Minic"
"Exploring de Sitter Space and Holography"       "Vijay Balasubramanian"
"Anti-de Sitter space and black holes"           "Andres Gomberoff"
"Anti-de Sitter space and black holes"           "Maximo Banados"
```

3.8. 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 (a1:Author {author_name: 'C.N. Pope'}),
  (a2:Author {author_name: 'M. Schweda'}),
  p = shortestPath((a1)-[*]-(a2))
 RETURN [n in nodes(p) | n.author_name] AS path, length(p)
```



path	length(p)
["C.N. Pope", *null*, *null*, *null*, *null*, *null*, "M. Schweda"]	6
["C.N. Pope", *null*, *null*, *null*, *null*, *null*, "M. Schweda"]	6
["C.N. Pope", *null*, *null*, *null*, *null*, "M. Schweda"]	5
["C.N. Pope", *null*, *null*, *null*, *null*, "M. Schweda"]	5
["C.N. Pope", *null*, *null*, *null*, *null*, *null*, "M. Schweda"]	6
["C.N. Pope", *null*, *null*, *null*, *null*, "M. Schweda"]	5

We can see that there are multiple shortest paths. The *shortestpath* function finds the single shortest path between two specific nodes. But in our case multiple Author nodes have the same author_name, that is why we get many shortest paths.

3.9. 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.

```
MATCH (a:Author{author_name:"Edward Witten"}),((a)-[:WROTE]-
>(article:Article)) MATCH p= ShortestPath((article)-[*]-
(b:Author)) WHERE length(p) >25 AND a<>b return b.name as author_Name, length(p)
as length , article.title as title
```

Query fails with "not enough memory" error after almost 1 hour of run time.