CS 5423: Principles of Database Systems

Part II

Neo4J Queries

Name: Raghu Medarametla

CWID: A20386265

Node Data:

	A	B	C
1	class	Name of class	Size of class
2	class1	Database	40
3	class2	Programming	35
4	class3	Datastructures	45
5	class4	Statistics	50
6	class5	Calculus	25
7	class6	Discrete maths	38
8	class7	Sets and Relations	47
9	class8	Operating System	36
10	class9	Computer Architecture	38
11	class10	Compiler Design	36
12	class11	Physics	24
13	class12	Quantam Computing	30
14	class13	Cloud Computing	29
15	class14	Machine Learning	33
16	class15	Formal Language Theory	31
17	class16	Artificial Intelligence	43
18	class17	Deep Learning	27
19	class18	Neural Network	37
20	class19	Natural Language Processing	43
21	class20	CyberSecurity	37

Edge Data:

	A	B	C	D	E	F
1	Edge	Node1	Node2	Connected by	common assignments	
2	Edge1	Sets and Relations	Database	Mathematics	12	
3	Edge2	Sets and Relations	statistics	Mathematics	15	
4	Edge3	statistics	Discrete Maths	Mathematics	20	
5	Edge4	Discrete Maths	Calculus	Mathematics	24	
6	Edge5	Database	Programming	Computer Science	27	
7	Edge6	Programming	Data Structures	Computer Science	18	
8	Edge7	Data Structures	Operating system	Computer Science	9	
9	Edge8	Operating System	Computer Architecti	Computer Science	23	
10	Edge9	Calculus	Physics	Physics	13	
11	Edge10	Physics	Quantam Computin	Physics	16	
12	Edge11	Cloud Computing	Quantam Computin	Applied Science	11	
13	Edge12	Cloud Computing	Machine Learning	Applied Science	14	
14	Edge13	Compiler Design	Formal Language Th	Design Systems	23	
15	Edge14	Machine Learning	Artificial Intelligence	Artifical Intelligence	32	
16	Edge15	Artifical Intelligence	Deep Learning	Artifical Intelligence	22	
17	Edge16	Deep Learning	Neural Network	Computer Science	26	
18	Edge17	Neural Network	Natural Language Pr	Computer Science	14	
19	Edge18	Natural Language Pro	CyberSecurity	Computer Security	21	
20	Edge19	Computer Architectur	Compiler Design	Computer Science	19	
21	Edge20	Formal Language Ther	CyberSecurity	Computer Science	17	

Query to load the Node data csv file:

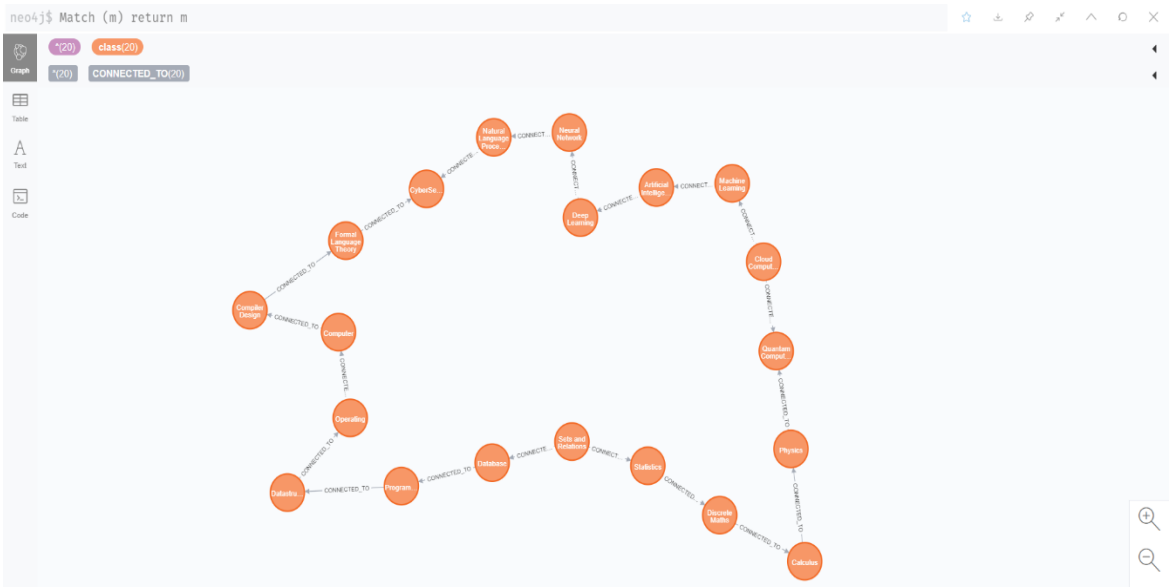
```
LOAD CSV WITH HEADERS FROM "file:///node-data.csv" AS row

MERGE (c:class {name: row.Nameofclass, classSize: toInteger(row.Sizeofclass)})
```

Query to load the Edge data csv file:

```
LOAD CSV WITH HEADERS FROM "file:///edge-data.csv" AS csvLine

MERGE (source:class {name: csvLine.Node1})

MERGE (destination:class {name: csvLine.Node2})

MERGE (source)-[:CONNECTED_TO {connectedBy: csvLine.ConnectedBy,
commonAssignment : toInteger(csvLine.commonAssignments)}]->(destination)
```

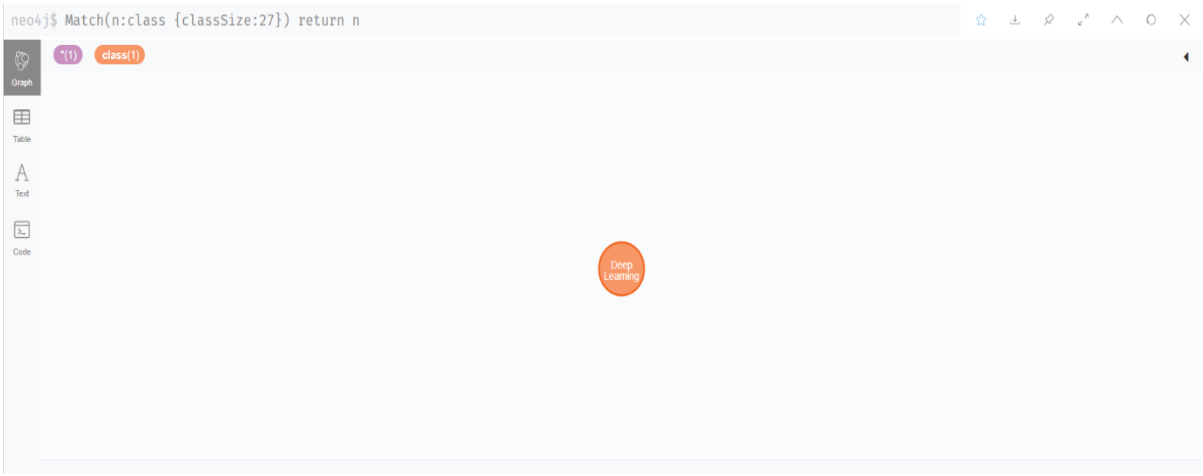
1.Display the graph created in Part I

Query: Match (m) return m



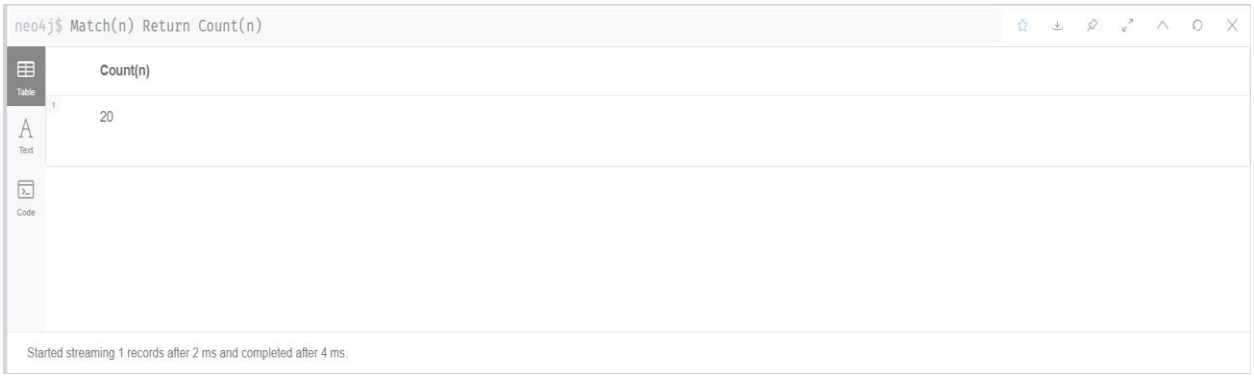
2. Find a specific node

Query: Match (n:class {classSize:27}) return n



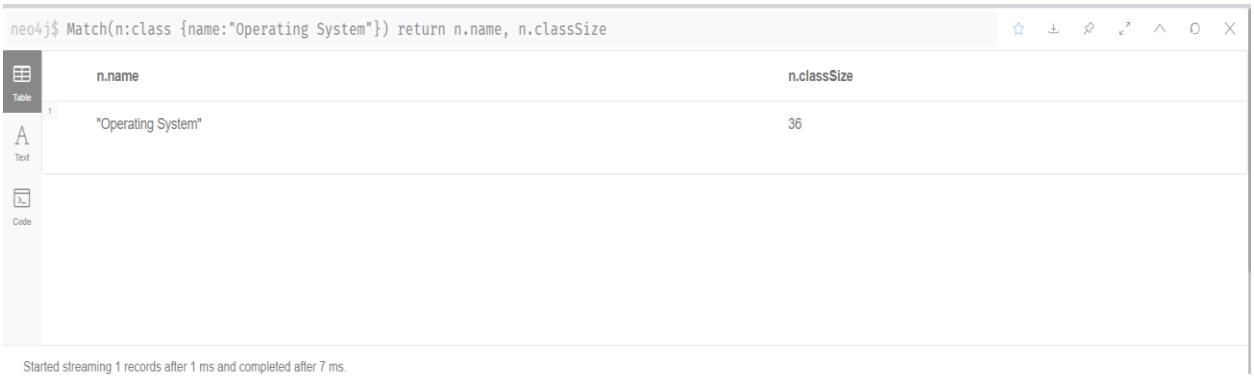
3. Find total number of nodes

Query: Match(n) Return Count(n)



4. Find a specific node by name or title

Query: Match(n:class {name:"Operating System"}) return n.name, n.classSize



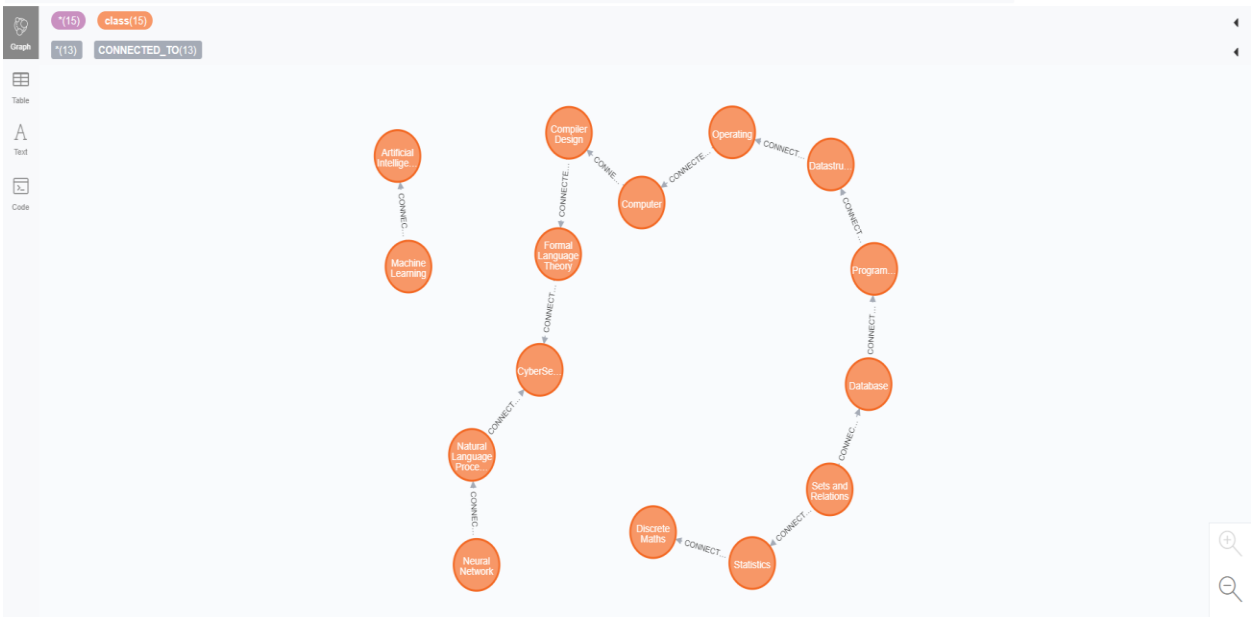
5. List the nodes that satisfy one property, for example, population > 20000 or position = 'goal keeper'

Query: Match(c:class) where c.classSize > 30 return c.name, c.classSize

```
neo4j$ Match(c:class) where c.classSize > 30 return c.name, c.classSize
```

	c.name	c.classSize
1	"Database"	40
2	"Programming"	35
3	"Datastructures"	45
4	"Statistics"	50
5	"Discrete Maths"	38
6	"Sets and Relations"	47
7	"Operating System"	36
8	"Computer Architecture"	38
9	"Compiler Design"	36

```
neo4j$ Match(c:class) where c.classSize > 30 return c
```



6. Find total number of edges

Query: match (n:class) - [r]->(p:class) return count(r)

```
neo4j$ match (n:class) - [r]→(p:class) return count(r)
```

	count(r)
1	20

Started streaming 1 records after 1 ms and completed after 4 ms.

7. **List relationships by type**

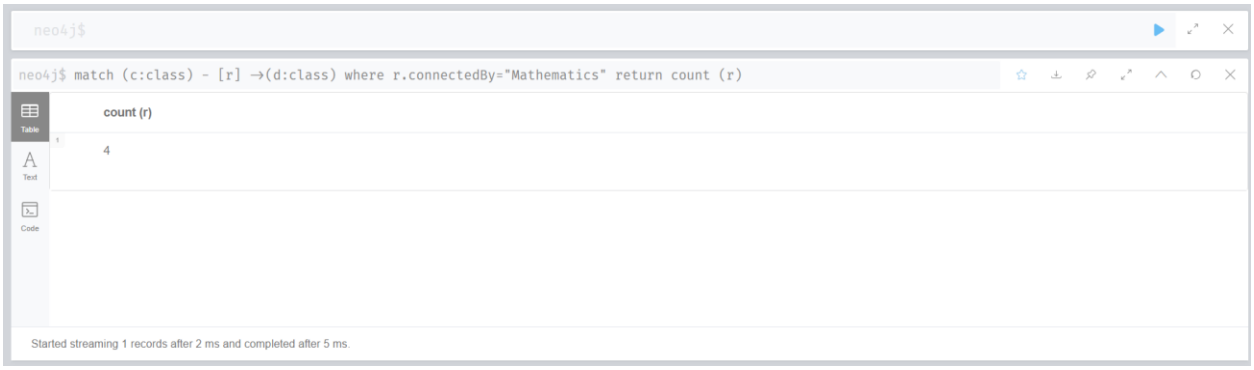
Query: match (c:class) - [r] ->(d:class) return distinct r.connectedBy

```
neo4j$ match (c:class) - [r] →(d:class) return distinct r.connectedBy
```

	r.connectedBy
1	"Computer Science"
2	"Mathematics"
3	"Physics"
4	"Design Systems"
5	"Applied Science"
6	"Artificial Intelligence"
7	"Computer Security"

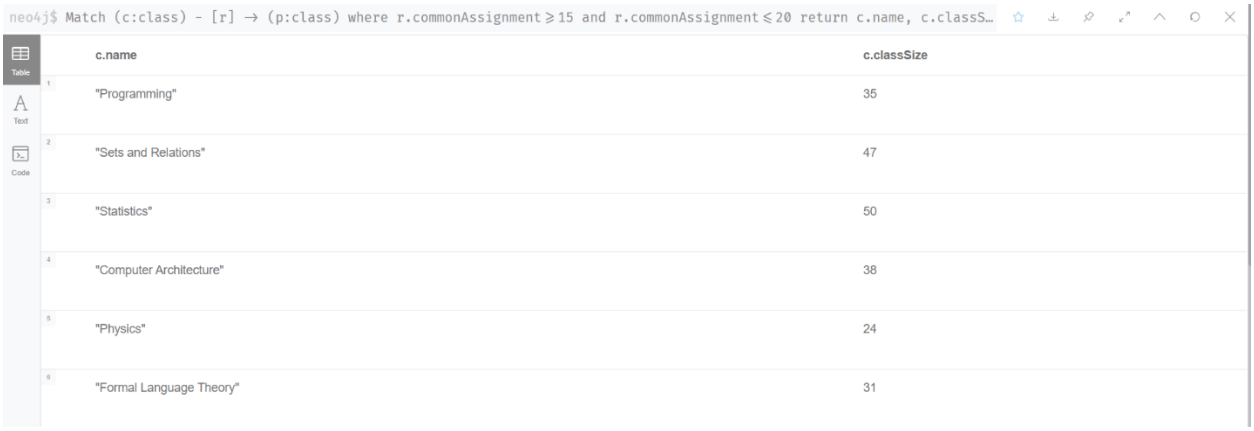
8. Count a certain type of relationship

Query: match (c:class) - [r] ->(d:class) return count (r.connectedBy='Mathematics')



9. Returns up to 3 nodes (and their relationships) where there's a property with a specific value or values or a specific range

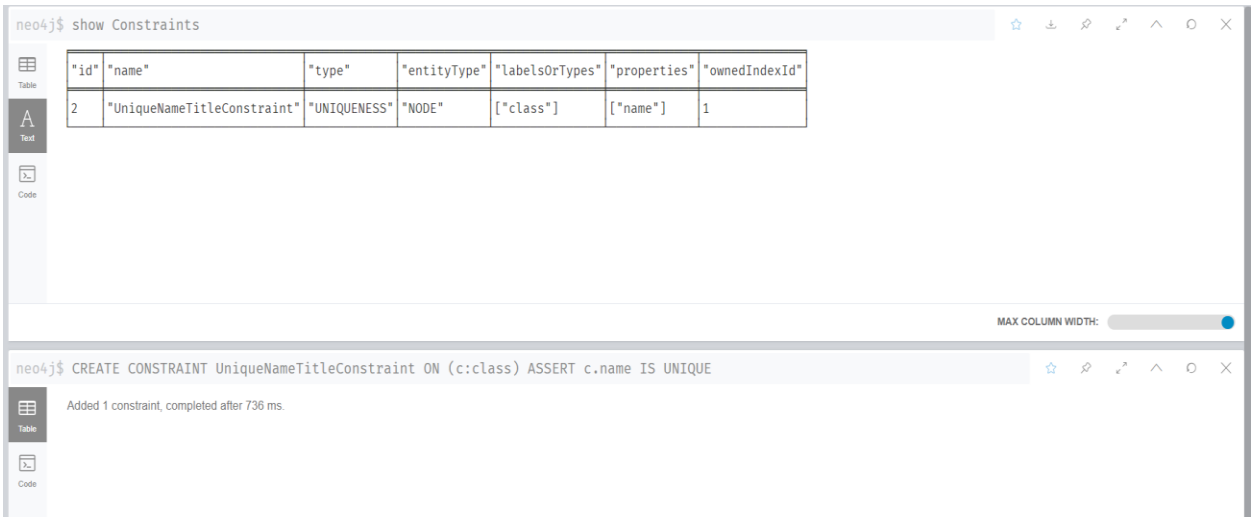
Query: Match (c:class) - [r] -> (p:class) where r.commonAssignment>=15 and r.commonAssignment<=20 return r.connectedBy, r.commonAssignment



	c.name	c.classSize
1	"Programming"	35
2	"Sets and Relations"	47
3	"Statistics"	50
4	"Computer Architecture"	38
5	"Physics"	24
6	"Formal Language Theory"	31

10. Create a unique property constraint

Query: CREATE CONSTRAINT UniqueNameTitleConstraint ON (c:class) ASSERT c.name IS UNIQUE. show Constraints



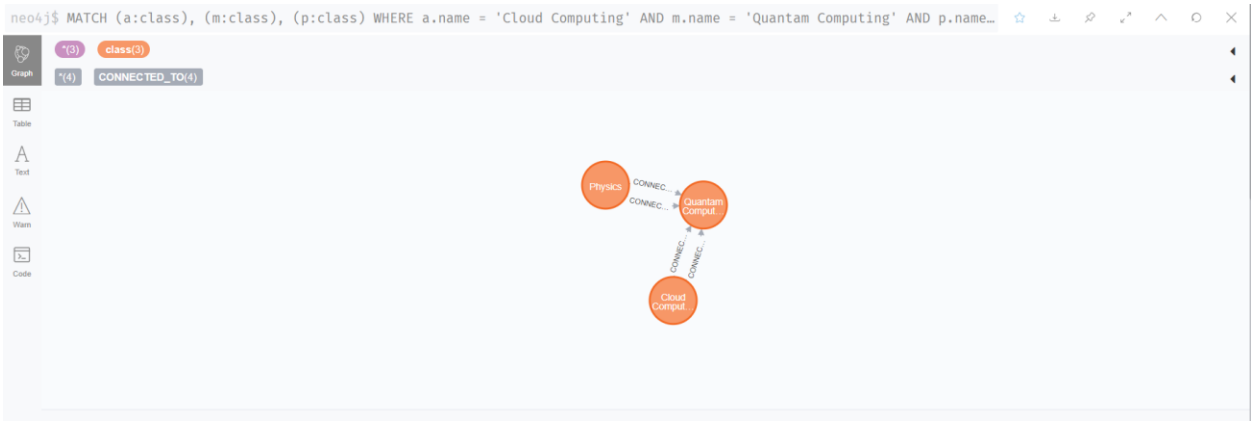
11. A query involving multiple relationships

Query: MATCH (a:class), (m:class), (p:class) WHERE a.name = 'Cloud Computing' AND

m.name = 'Quantam Computing' AND p.name = 'Physics'

CREATE (a)-[:CONNECTED_TO]->(m)<-[:CONNECTED_TO]-(p)

RETURN a, m, p

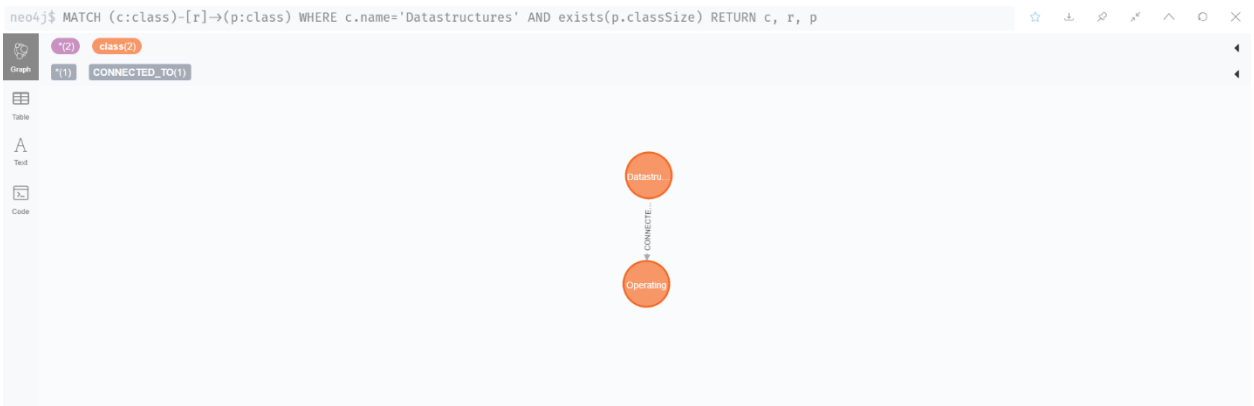


12. A query that uses patterns

Query: MATCH (c:class)-[:CONNECTED_TO]->(:class {name:'Database'}) RETURN c.name, c.classSize

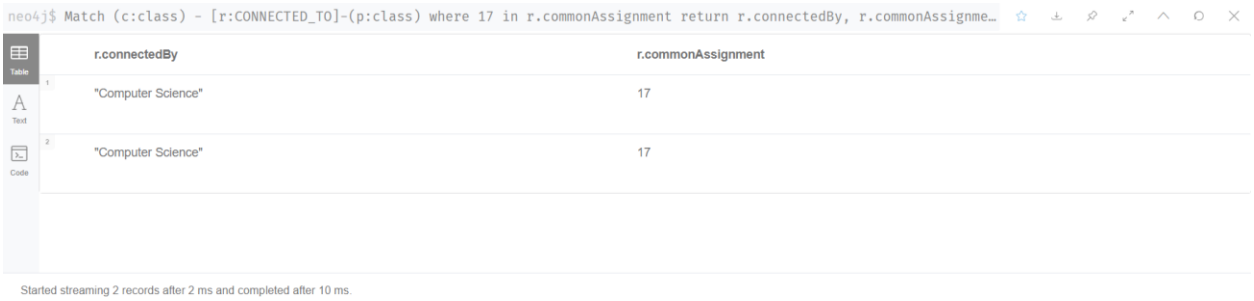
13. A query that tests the existence of a property

Query: MATCH (c:class)-[r]->(p:class) WHERE c.name='Datastructures' AND exists(p.classSize) RETURN c, r, p



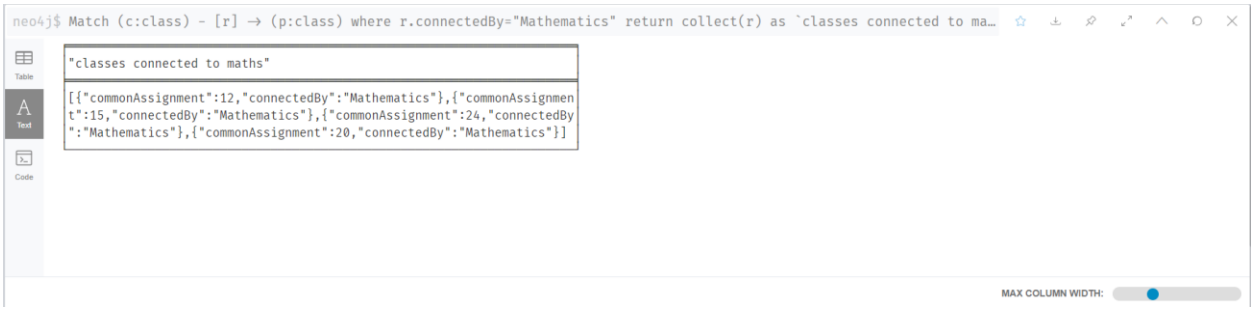
14. A query that tests using List values

Query: Match (c:class) - [r:CONNECTED_TO]-(p:class) where 17 in r.commonAssignment return r.connectedBy, r.commonAssignment



15. A query that collects results

Query: Match (c:class) - [r] -> (p:class) where r.connectedBy="Mathematics" return collect(r) as `classes connected to maths`



16. A query that counts and collects

Query: Match (c:class) - [r:CONNECTED_TO] -> (p:class) return c.name, c.classSize, count(p) AS connections, collect(p.name) AS connectionnames

```
neo4j$ Match (c:class) - [r:CONNECTED_TO] → (p:class) return c.name, c.classSize, count(p) AS connections, collect(p.name) AS con…
```

	c.name	c.classSize	connections	connectionnames
1	"Sets and Relations"	47	2	["Database", "Statistics"]
2	"Database"	40	1	["Programming"]
3	"Programming"	35	1	["Datastructures"]
4	"Discrete Maths"	38	1	["Calculus"]
5	"Statistics"	50	1	["Discrete Maths"]
6	"Datastructures"	45	1	["Operating System"]
7	"Operating System"	36	1	["Computer Architecture"]
8	"Computer Architecture"	38	1	["Compiler Design"]
9	"Calculus"	25	1	["Physics"]

Started streaming 18 records in less than 1 ms and completed after 57 ms.

17. A query that uses WITH

Query: MATCH (c:class)-[:CONNECTED_TO]->(m:class)WITH c, count(m) AS numClasses, collect(m.name) as classes WHERE 1 < numClasses < 4

RETURN c.name, numClasses, classes

```
neo4j$ MATCH (c:class)-[:CONNECTED_TO]→(m:class) WITH c, count(m) AS numClasses, collect(m.name) as classes WHERE 1 < nu…
```

	c.name	numClasses	classes
1	"Statistics"	2	["Database", "Discrete Maths"]
2	"Sets and Relations"	3	["Database", "Database", "Statistics"]
3	"Physics"	2	["Quantam Computing", "Quantam Computing"]
4	"Cloud Computing"	3	["Quantam Computing", "Quantam Computing", "Machine Learning"]

Started streaming 4 records after 7 ms and completed after 150 ms.

18. A query that uses WITH and UNWIND

Query: Match (c:class) - [r] -> (p:class) where r.connectedBy="Computer Science" with collect(r) AS computerscienceclass unwind computerscienceclass AS eachRow Return eachRow.connectedBy, eachRow.commonAssignment

```
neo4j$ Match (c:class) - [r] → (p:class) where r.connectedBy="Computer Science" with collect(r) AS computerscienceclass unwind co…
```

	eachRow.connectedBy	eachRow.commonAssignment
1	"Computer Science"	27
2	"Computer Science"	18
3	"Computer Science"	9
4	"Computer Science"	23
5	"Computer Science"	19
6	"Computer Science"	26
7	"Computer Science"	14
8	"Computer Science"	17

19. Subqueries with WITH

Query: match (c:class) - [r:CONNECTED_TO]->(p:class) with c,r,p match (c) <-[CONNECTED_BY] - (d:class) return distinct c.name, c.classSize

```
neo4j$ match (c:class) - [r:CONNECTED_TO]→(p:class) with c,r,p match (c) ← [CONNECTED_BY] - (d:class) return distinct c.name, c.…
```

	c.name	c.classSize
1	"Database"	40
2	"Programming"	35
3	"Datastructures"	45
4	"Statistics"	50
5	"Calculus"	25
6	"Discrete Maths"	38
7	"Operating System"	36
8	"Computer Architecture"	38
9	"Compiler Design"	36

Started streaming 16 records after 53 ms and completed after 84 ms.

20. Performing subqueries with CALL

Query: call { match (d:class)-[:CONNECTED_TO]->(p:class) return p} match(p) where p.name="Datastructures" return p.name, p.classSize

```
neo4j$ call { match (d:class)-[:CONNECTED_TO]→(p:class) return p} match(p) where p.name="Datastructures" return p.name, p…
```

	p.name	p.classSize
1	"Datastructures"	45

Started streaming 1 records after 308 ms and completed after 312 ms.

21. Ordering results

Query: match(c:class) return c.name, c.classSize order by c.classSize

neo4j$ match(c:class) return c.name, c.classSize order by c.classSize

	c.name	c.classSize
1	"Physics"	24
2	"Calculus"	25
3	"Deep Learning"	27
4	"Cloud Computing"	29
5	"Quantam Computing"	30
6	"Formal Language Theory"	31

Started streaming 20 records after 1 ms and completed after 50 ms.

22. LIMIT keyword with WITH clause to limit intermediate results.

Query: match (c:class) - [r:CONNECTED_TO] - (p:class) where r.commonAssignment=17 with c Limit 33 return distinct c.name

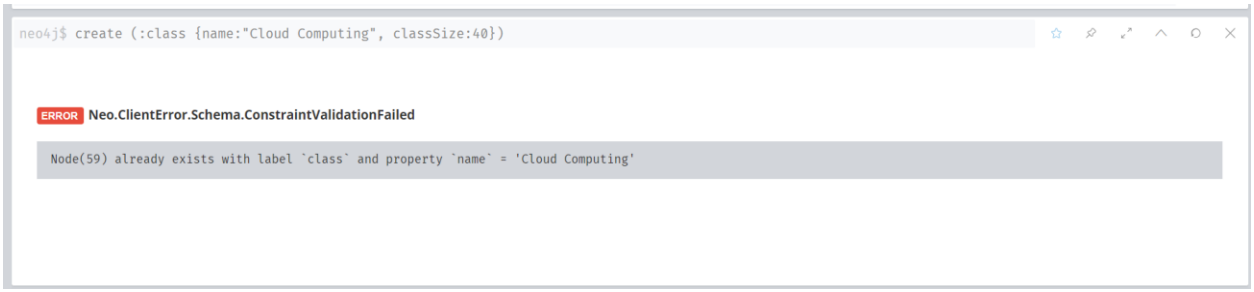
neo4j$ match (c:class) - [r:CONNECTED_TO] - (p:class) where r.commonAssignment=17 with c Limit 33 return distinct c.name

	c.name
1	"CyberSecurity"
2	"Formal Language Theory"

Started streaming 2 records after 1 ms and completed after 15 ms.

23. Ensuring that a property value for a node is unique.

Query: create (:class {name:"Cloud Computing", classSize:40})



24. A query that finds the shortest path (Neo4J has a built-in shortest path function)

Query: match d= shortestPath((c:class)-[*]-(p:class)) where c.name="Compiler Design" and p.name = "Cloud Computing" return d