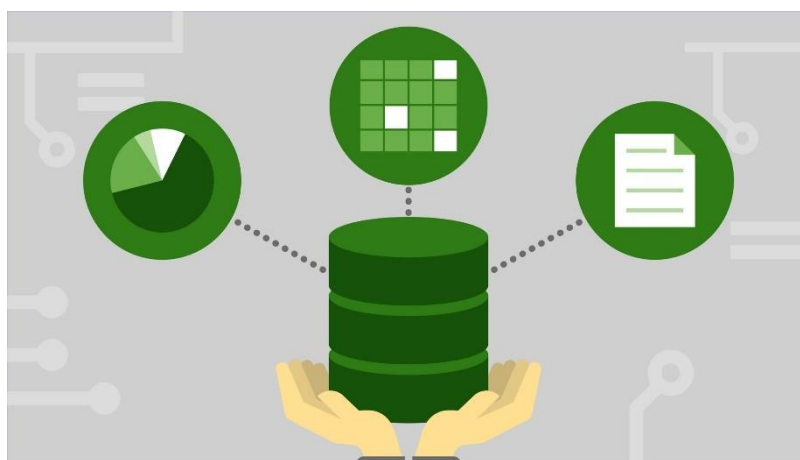


به نام خدا



دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر



آزمایشگاه پایگاه داده

دستور کار شماره ۳

نام و نام خانوادگی

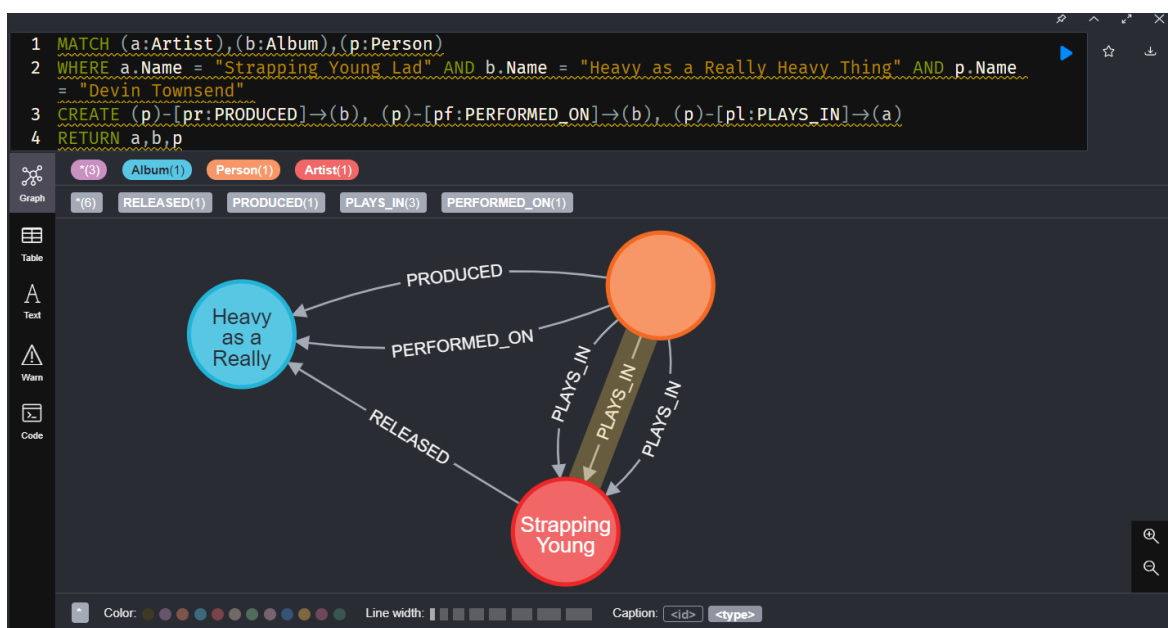
معین شیردل ۸۱۰۱۹۷۵۳۵

آبان ماه ۱۴۰۰

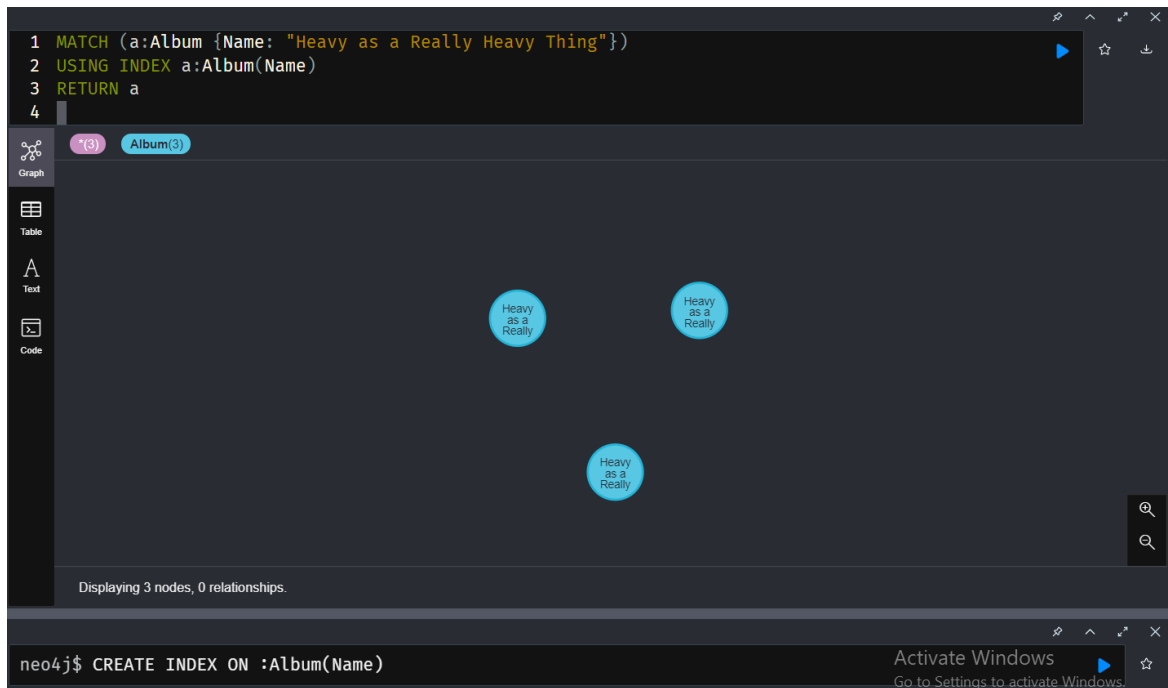
خلاصه آموزش‌های سایت Quackit:



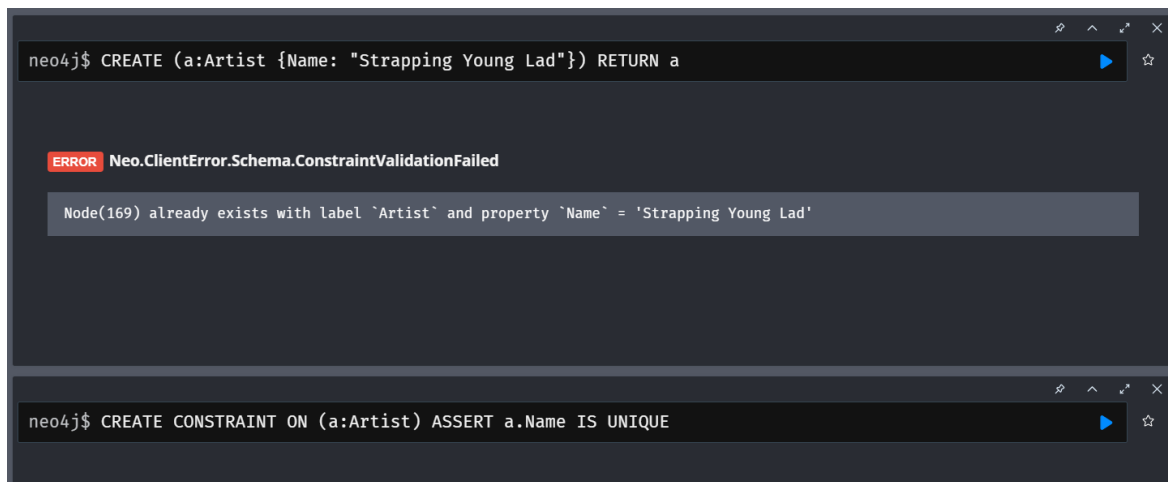
دستور CREATE: با استفاده از این دستور می‌توان تعدادی node با اطلاعات داده شده (Name) ایجاد کرد.



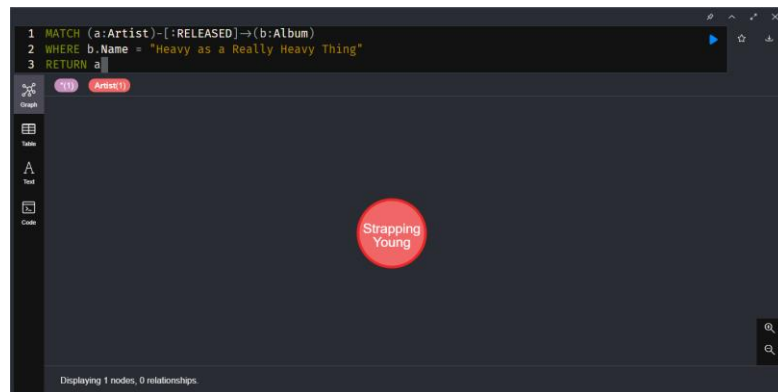
دستورات Match و ساختن روابط: در این قسمت تعدادی رابطه بین موجودیت‌ها تعریف شد. ابتدا این موجودیت‌ها تعریف شدند. سپس به کمک دستور Match یافت می‌شوند و با دستور $(a)-[r:relation]->(b)$ یک رابطه بین a و b می‌سازیم. موجودیت نارنجی رنگ یک Person است که قبل تر ساخته شده‌است.



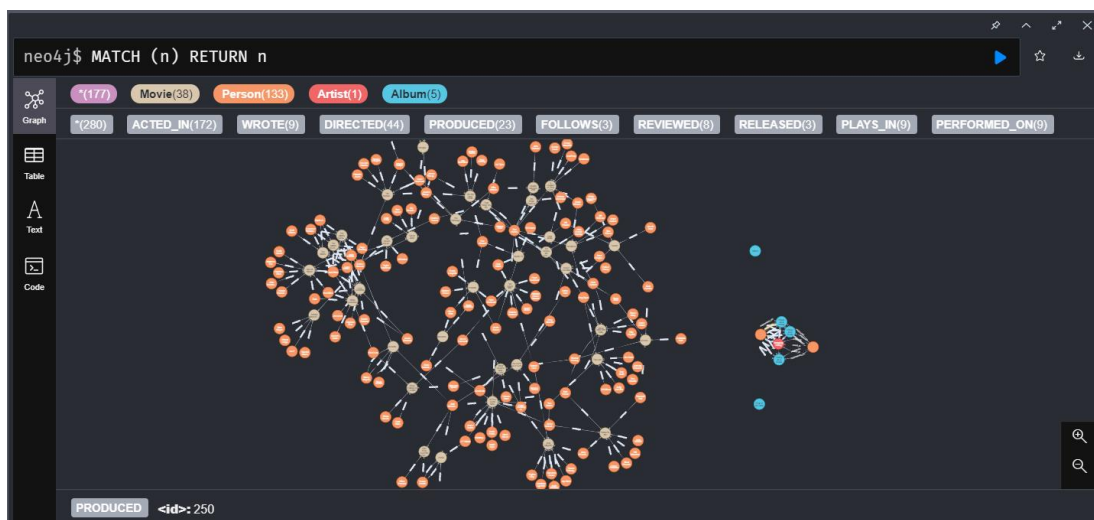
دستور Create Index: برای تعریف کردن شاخص روی نود ها و ویژگی هایشان و افزایش سرعت بازیابی اطلاعات. در مثال بالا یک شاخص روی Name ساخته شد و با استفاده از آن، تعدادی node بازیابی شدند.



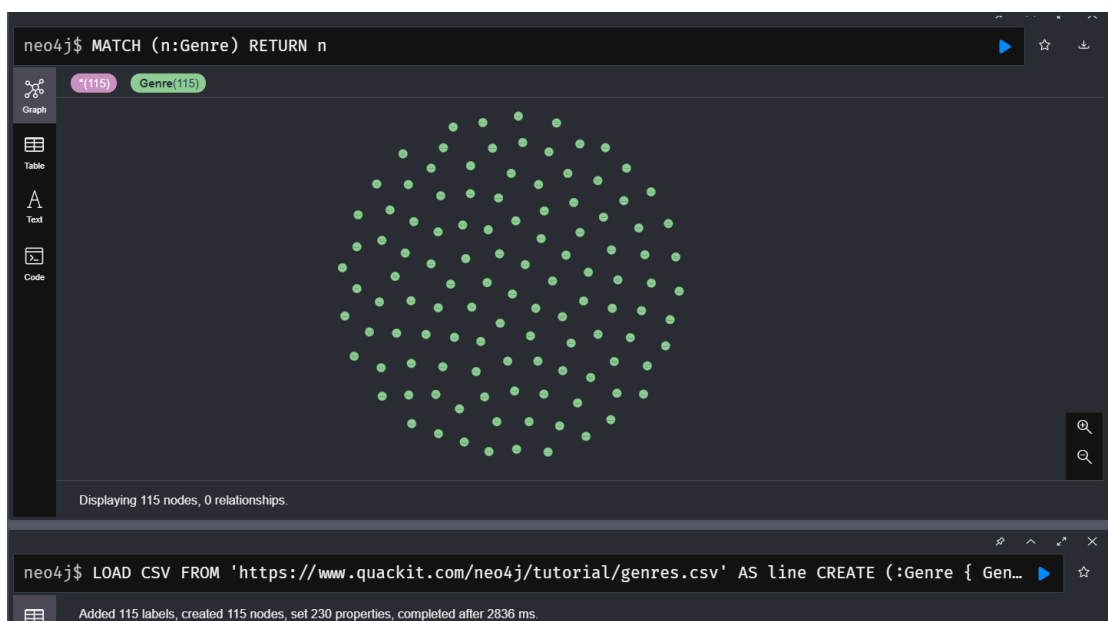
دستور Create Constraint: دو نوع constraint در این بخش معرفی شد که `exists()` محدودیت عدم وجود مقدار خالی برای فیلد مدنظر و `Is Unique` محدودیت عدم وجود نود با مقدار تکراری آن فیلد را نشان می‌دهد. در تصویر مشکل ورودی مقدار تکراری در صورت وجود uniqueness constraint آورده شده است.



استفاده از دستور Match: در این قسمت، از این دستور استفاده شد تا Artist ای (هایی) که آلبومی با نام ... Heavy as a را منتشر کرده اند را بیابد.



با دستور Match(n) Return n نیز می‌توان تمام نود های موجود را دید.



با استفاده از دستور Load csv از لینک داده شده یک فایل csv خوانده شد و genre ها از آن ایمپورت شد. همچنین امکان انتخاب برخی از ستون های فایل csv، یا مشخص کردن header داشتن یا نداشتن ستون های فایل وجود دارد. همچنین برای خواندن فایل های بزرگ، می توان از Periodic Commit استفاده کرد تا پس از هر خواندن تعداد رکورد مشخص از فایل csv، آن ها را کامیت کند و بعد بقیه را بخواند. این عدد مشخص را می توان تعیین کرد ولی به طور پیش فرض ۱۰۰۰ است.

The screenshot shows the Neo4j Cypher Shell interface. The top panel displays the database schema with the following table:

Index Name	Type	Uniqueness	EntityType	LabelsOrTypes	Properties	State
constraint_6a938e7e	BTREE	UNIQUE	NODE	["Artist"]	["Name"]	ONLINE
index_343aff4e	LOOKUP	NONUNIQUE	NODE	[]	[]	ONLINE
index_f7700477	LOOKUP	NONUNIQUE	RELATIONSHIP	[]	[]	ONLINE

Below the table, the 'Constraints' section shows two assertions:

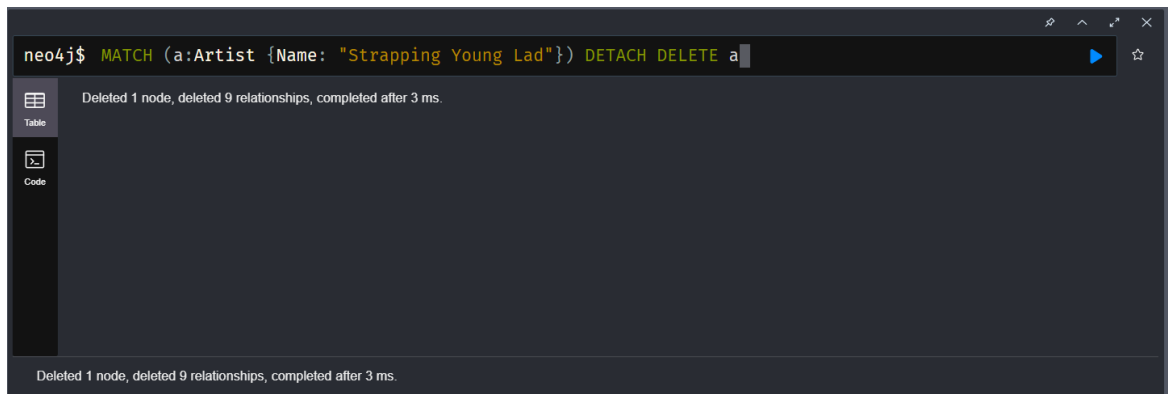
- ON (artist:Artist) ASSERT (artist.Name) IS UNIQUE
- ON (album:Album) ASSERT (album.Name) IS NOT NULL

A command is entered in the shell: `CALL db.schema.visualization`. The bottom panel shows the execution of the command `neo4j$ DROP INDEX ON :Album(Name)`, resulting in the message: 'Removed 1 index, completed after 3 ms.'

به کمک دستور Drop Index می تواند شاخص های تعیین شده را حذف کرد. دستور Drop Constraint نیز عمل مشابه را روی Constraint ها انجام می دهد.

The screenshot shows the Neo4j Cypher Shell interface. The top panel displays the command `neo4j$ MATCH (:Artist)-[r:RELEASED]-(:Album) DELETE r`. The result shows: 'Deleted 3 relationships, completed after 9 ms.' The bottom panel shows the command `neo4j$ MATCH (a:Album {Name: "Killers"}) DELETE a`, resulting in the message: '(no changes, no records)'.

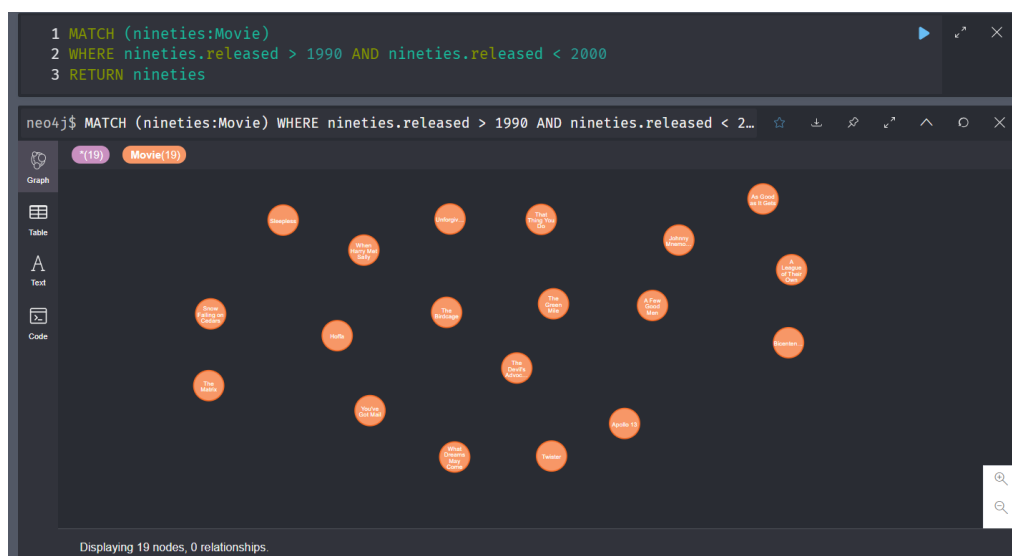
به کمک دستور Delete هم می توان روابط و نود ها را حذف کرد. مشکل در حالتی است که نودی را می خواهیم حذف کنیم و روابطی وابسته به آن نود وجود دارد که در این حالات، یک روش استفاده از Detach Delete است که یک نود و تمام روابطش را پاک می کند. نتیجه در تصویر پایین قابل مشاهده است:



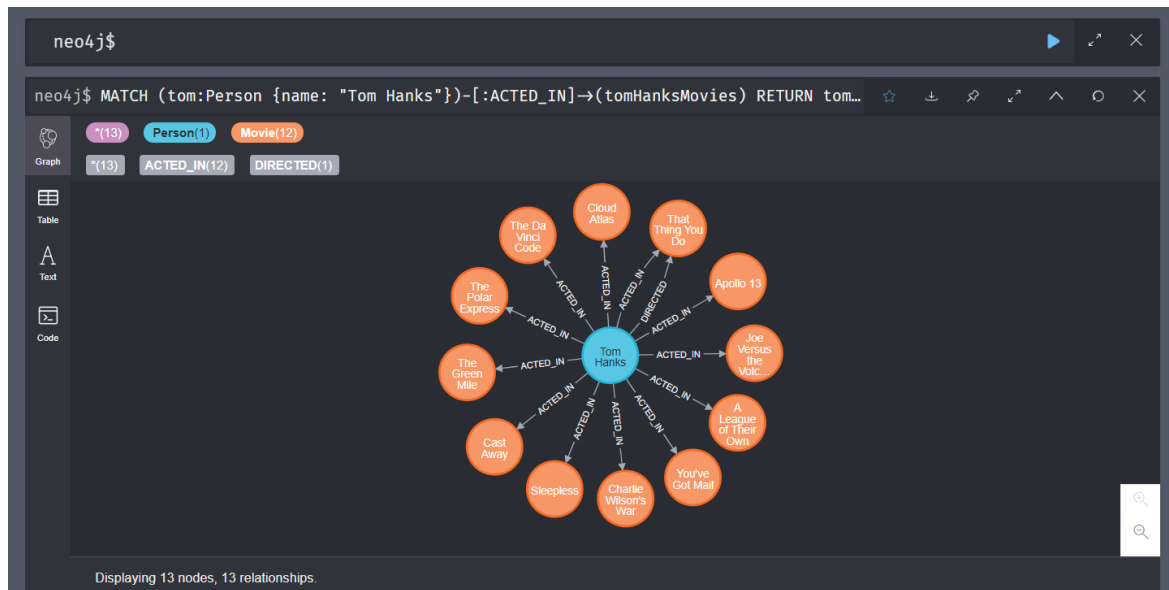
بخش آموزش‌های وبسایت Neo4j:



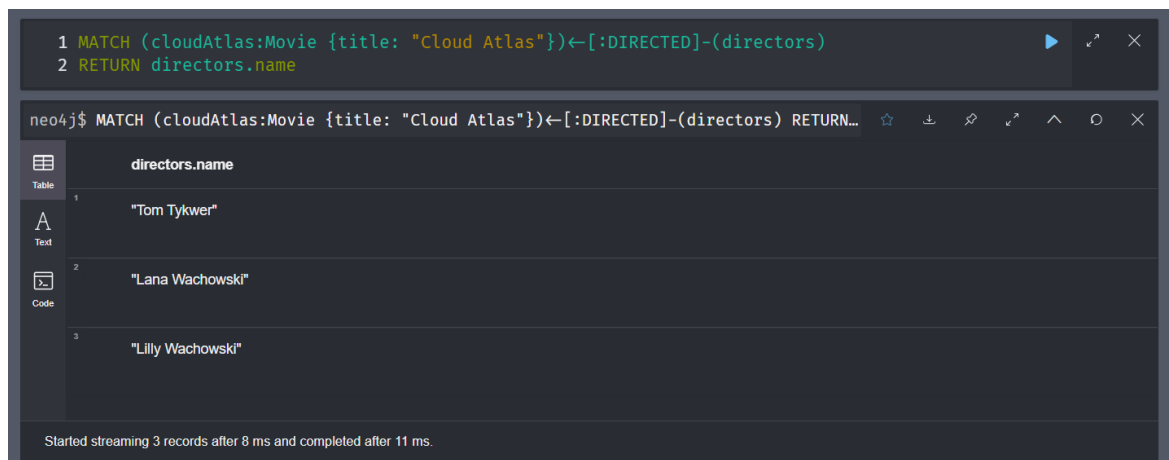
طبق این دستور، نام ۱۰ نفر از افراد موجود در دیتابیس بازگردانده می‌شود. پیش‌تر مشابه این دستور را (البته حالتی که خود نود را برمیگرداند) دیده بودیم.



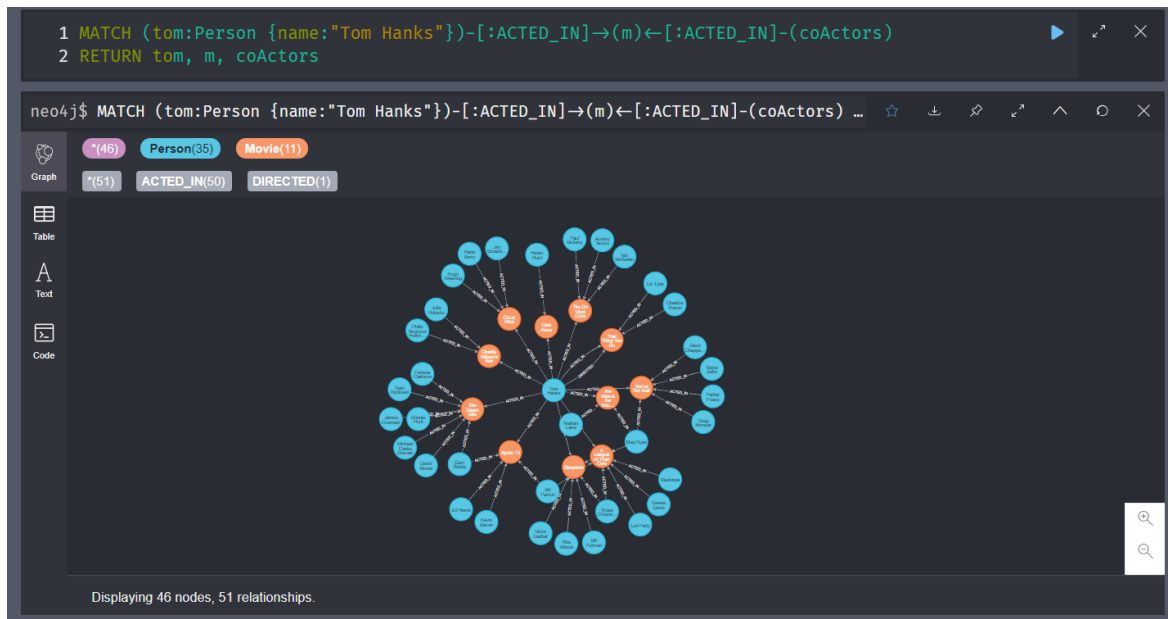
به طریق بالا، نود های فیلم هایی که سال ساختشان بین سال های ۱۹۹۰ و ۲۰۰۰ بود به دست آمد و فیلتر اعمال شد.



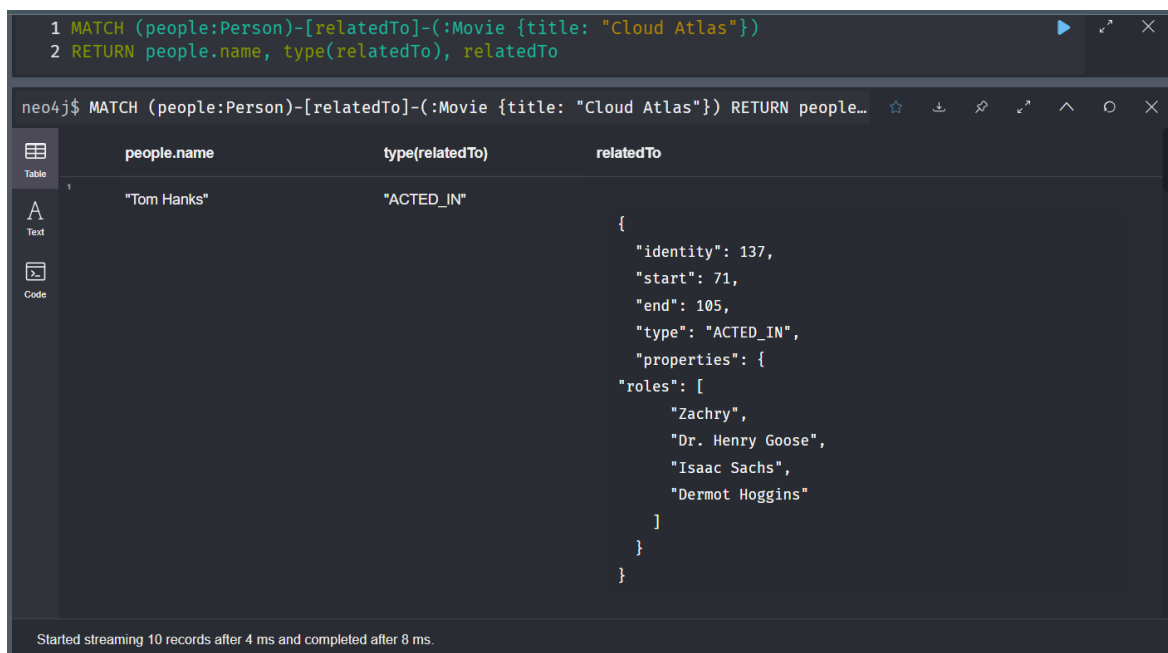
به کمک کد بالا، تمام روابط ACTED_IN که گرهی با نام tom hanks در آن ها شرکت داشته را به دست می آوریم که معادل با تمام فیلم هایی است که Tom Hanks در آنها ایفای نقش کرده است.



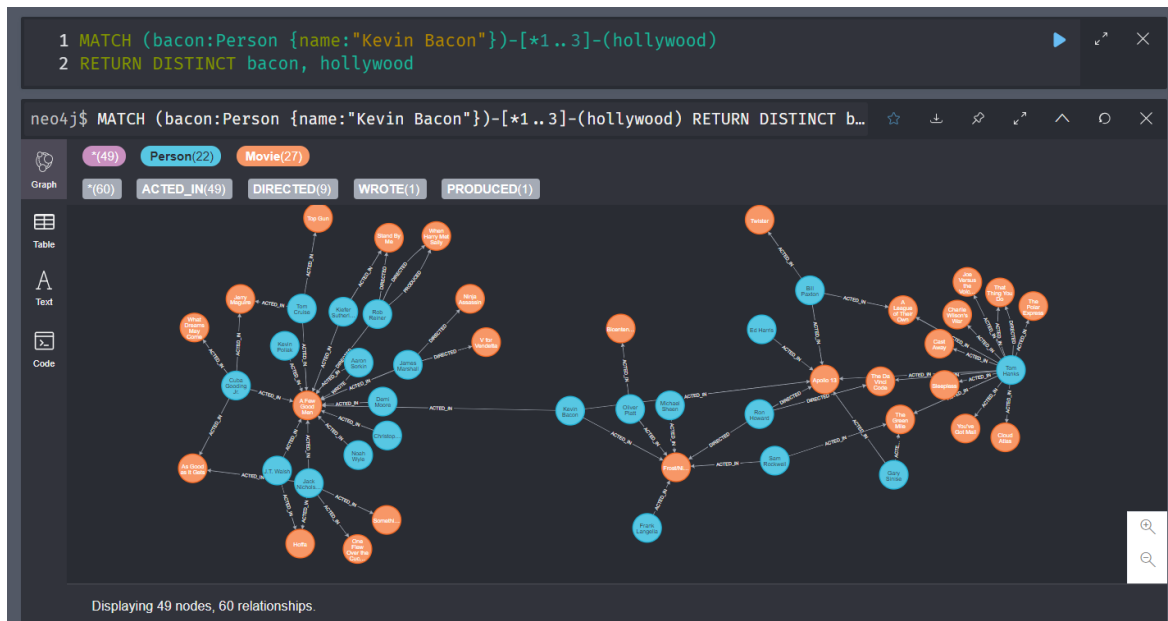
همچنین قابلیت پیدا کردن تمام کارگردان های فیلم Cloud Atlas را نیز با ردیابی روابط DIRECTED وارد بر این گره را نیز داریم.



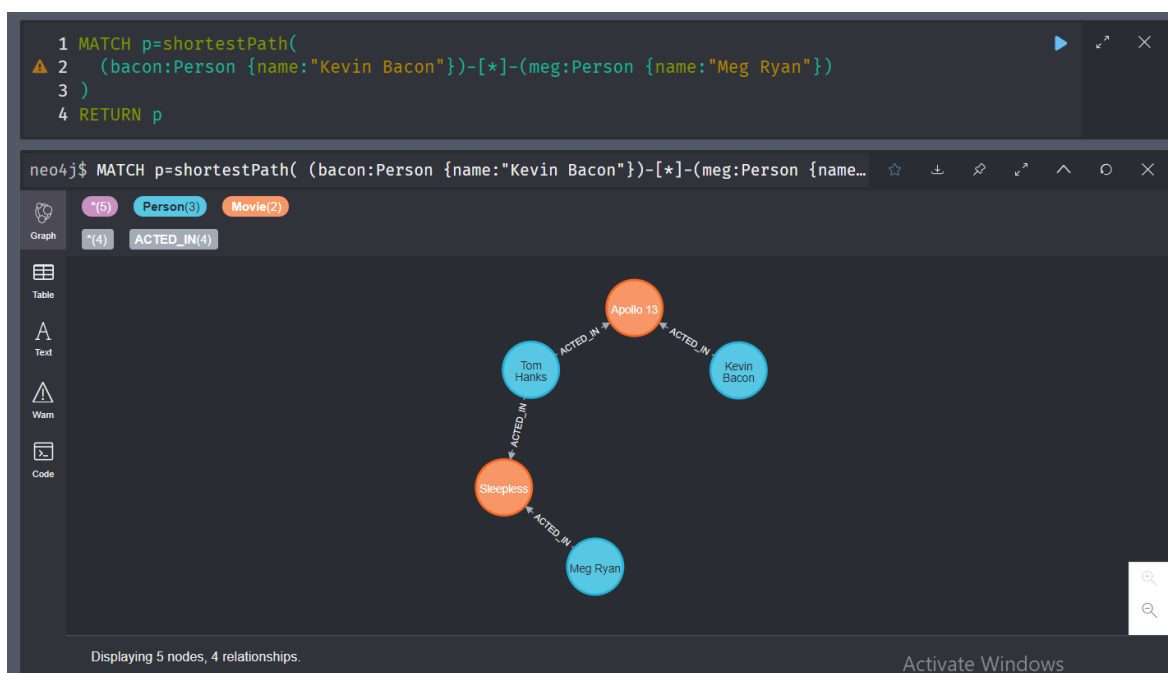
حال می‌توان بازیگرانی که به همراه Tom Hanks در یکی از فیلم‌ها بازی کرده‌اند را نیز به دست بیاوریم. بدین صورت که ابتدا از Tom Hanks به گره فیلم هایش میرسیم و سپس از آن فیلم‌ها، به بازیگرانی میرسیم که آن فیلم با آن بازیگر رابطه ACTED_IN داشته باشد.



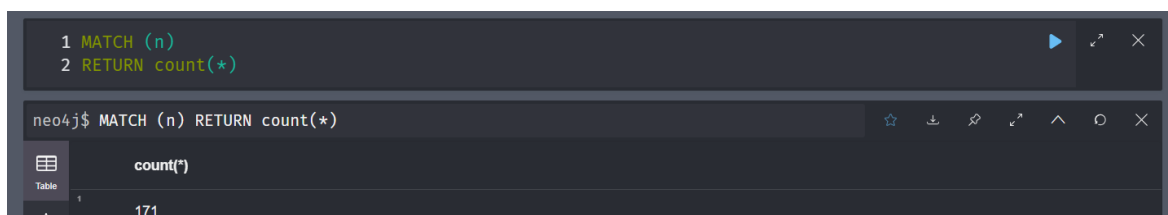
به کمک این کد، تمام روابط فیلم Cloud Atlas شرح داده می‌شوند. به طور مثال رکورد اول این نتیجه، می‌گوید که نود به کمک این کد، تمام روابط فیلم Cloud Atlas شرح داده می‌شوند. به طور مثال رکورد اول این نتیجه، می‌گوید که نود Tom Hanks در یک رابطه ACTED_IN با این فیلم بوده است و ... دستور type(relation) نوع یک رابطه را نشان می‌دهد.



از این قسمت به بعد، مفهوم فاصله نیز مطرح می‌شود. در کد بالا، تمام مسیرهای به طول ۳ از گره با نام Kevin Bacon و دیگر گره‌های درگیر در این مسیرها مشخص شده‌اند. [3 .. 1]* نشانگر طول مسیر رابطه‌ای است.

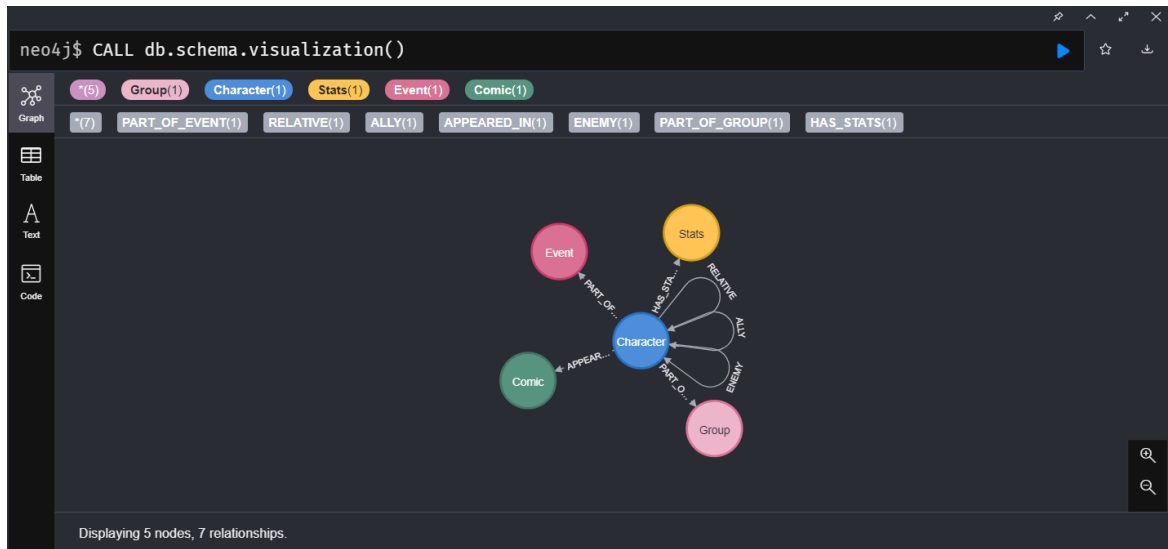


حال در گراف مفهوم کوتاه‌ترین مسیر مطرح می‌شود. در کد بالا، کوتاه‌ترین مسیر بین گره با نام Kevin و Meg Ryan در p ذخیره شده‌است و دیده می‌شود که طول این مسیر ۳ است.



به کمک دستور count(*) نیز تعداد تمامی گره‌ها شمرده و بازگردانده می‌شود.

بخش بررسی روابط کاراکتر های دنیای سینمای مارول:



در این قسمت، ابتدا داده ها با دستور LOAD CSV خوانده شدند و تصویر بالا، نمایی از ساختار کلی دیتابیس خوانده شده است.



در این قسمت، ۵ کاراکتری که در بیشترین تعداد کمیک ها حضور داشته اند نمایش داده شده است. تعداد حضورها، با شمارش طول لیست $(c)-[:APPEARED_IN]->()$ بدست می آید.

```

1 MATCH (e:Event)
2 RETURN e.title as event,
3       size((e)-[:PART_OF_EVENT]-()) as count_of_heroes,
4       e.start as start,
5       e.end as end,
6       e.description as description
7 ORDER BY count_of_heroes DESC
8 LIMIT 5

```

"event"	"count_of_heroes"	"start"	"end"	"description"
"Fear Itself"	132	"2011-04-16 00:00:00"	"2011-10-18 00:00:00"	"The Serpent, God of Fear and brother to the Allfather Odin, rises to challenge Earth's Mightiest in a seven-issue event written by Matt Fraction with art by Stuart Immonen! As the Worthy, heralds of the Serpent, lay waste to the Marvel Universe, how can Thor, Captain America, Iron Man and the Avengers turn back the tide of fear?"
"Dark Reign"	128	"2008-12-01 00:00:00"	"2009-12-31 12:59:00"	"Norman Osborn came out the hero of Secret Invasion, and now the former Green Goblin has been handed control of the Marvel Universe. With his Cabal and the Dark Avengers at his side, can anything stop this long time villain from reshaping the world in his own image? And what has become of the heroes?"
"Acts of Vengeance!"	93	"1989-12-10 00:00:00"	"2008-01-04 00:00:00"	"Loki sets about convincing the super-villains of Earth to attack heroes other than those they normally fight in an attempt to destroy the Avengers to absolve his guilt of ever inadvertently creating the team in the first place."
"Secret Invasion"	89	"2008-06-02 00:00:00"	"2009-01-25 00:00:00"	"The shape-shifting Skrulls have been infiltrating the Earth for years, replacing many of Marvel's heroes with impostors, setting the stage for an all-out invasion."
"Civil War"	86	"2006-07-01 00:00:00"	"2007-01-29 00:00:00"	"After a horrific tragedy raises questions on whether or not super heroes should register with the government, longtime Avengers teammates Captain America and Iron Man end up on opposite sides of the argument! Writer Mark Millar and artist Steve McNiven split the Marvel Universe in two as friend fights friend in one of the most celebrated and successful events of all-time!"

به ازای هر Event، تاریخ شروع و پایان آن (start / end)، توضیحات آن و تعداد کاراکترهای شرکت‌کننده‌ی آن ذخیره و در جدول آورده شده است و در نهایت، ۵ تا از پرجمعیت‌ترین هایشان نمایش داده شده‌است.

```

neo4j$ MATCH (g:Group) RETURN g.name as group, size((g)-[:PART_OF_GROUP]-()) as members ORDER BY...

```

	group	members
1	"X-Men"	41
2	"Avengers"	31
3	"Defenders"	26
4	"Next Avengers"	14
5	"Guardians of the Galaxy"	12

Started streaming 5 records after 9 ms and completed after 9 ms.

در این قسمت برای هر گروه تعداد افراد آن محاسبه و شمارش شده‌اند و ۵ گروه پرجمعیت معرفی شده‌اند.

```

1 MATCH (c1:Character)-[:PART_OF_GROUP]→(g:Group)←[:PART_OF_GROUP]-(c2:Character)
2 WHERE (c1)-[:ENEMY]-(c2) and id(c1) < id(c2)
3 RETURN c1.name as character1, c2.name as character2, g.name as group

```

	character1	character2	group
1	"Logan"	"Sabretooth (House of M)"	"X-Men"
2	"Logan"	"Mystique (House of M)"	"X-Men"
3	"CAIN MARKO JUGGERNAUT"	"Logan"	"X-Men"
4	"CAIN MARKO JUGGERNAUT"	"Storm (Marvel Heroes)"	"X-Men"
5	"Rogue (X-Men: Battle of the Atom)"	"Warren Worthington III"	"X-Men"

Started streaming 5 records after 27 ms and completed after 87 ms.

در این قسمت نیز، ابتدا هر دوتایی از شخصیت هایی که عضو یک گروه هستند Match می شوند (c1 و c2). سپس یک شرط می گذاریم که بین c1 و c2 رابطه دشمنی برقرار باشد و برای اینکه این زوج، دو بار در جدول پدیدار نشوند، شرط $id(c1) < id(c2)$ را نیز اضافه می کنیم. در نهایت نام هر دو شخصیت و گروهشان را خروجی می دهیم.

```

1 MATCH (c:Character)
2 WHERE c.education contains "Ph.D"
3 RETURN c.name as character, c.education as education
4 LIMIT 10

```

	character	education
1	"UNKNOWN ACHEBE"	"Ph.D. in Law (Yale), degrees in Psychology, Political Science and Divinity"
2	"PROFESSOR MENDEL STROMM MENDEL STROMM"	"Ph.D. in robotics"
3	"FRANKLIN HALL GRAVITON"	"Ph.D. in physics"
4	"Morbis"	"Ph.D in Biochemistry"
5	"Tony Stark"	"Ph.Ds in physics and electrical engineering"
6	"Hulk-dok"	"Ph.D in nuclear physics and two other fields"
7		

در این قسمت هم شخصیت هایی را معرفی می کنیم که تحصیلات دکتری دارند و نام و تحصیلاتشان را نمایش می دهیم.

```

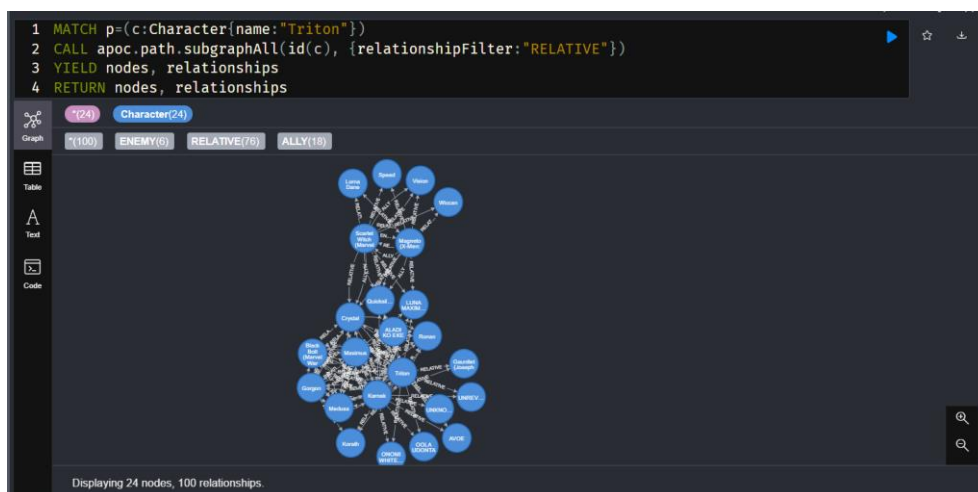
1 MATCH (c:Character)
2 RETURN c.name as name,
3       size((c)-[:ALLY]→()) as allies,
4       size((c)-[:ENEMY]→()) as enemies,
5       size((c)-[:RELATIVE]→()) as relative
6 ORDER BY allies + enemies + relative DESC
7 LIMIT 5

```

	name	allies	enemies	relative
1	"Scarlet Witch (Marvel Heroes)"	16	14	8
2	"Thor (Marvel: Avengers Alliance)"	9	14	10
3	"Invisible Woman (Marvel: Avengers Alliance)"	13	10	7
4	"Logan"	14	10	5
5	"Karnak"	6	2	17

Started streaming 5 records after 25 ms and completed after 42 ms.

در این بخش نیز با شمردن تعداد روابط ENEMY، ALLY و RELATIVE وضعیت هر کاراکتر را به دست می‌آوریم و بر اساس مجموع تعداد این روابط مرتب سازی می‌کنیم.



در این قسمت، تمام روابط فامیلی RELATIVE منشا گرفته از شخصیت Triton را رسم کرده ایم و به مجموعه ۲۴ گره و ۱۰۰ رابطه فامیلییشان رسیده‌ایم.

```

1 CALL gds.wcc.stream({
2   nodeProjection:'Character',
3   relationshipProjection:'ALLY'})
4 YIELD nodeId, componentId
5 WITH componentId, count(*) as members
6 WHERE members > 1
7 RETURN componentId, members
8 ORDER BY members DESC
9 LIMIT 5

```

	componentId	members
1	0	195
2	26	4
3	245	3
4	6	2
5	2	2

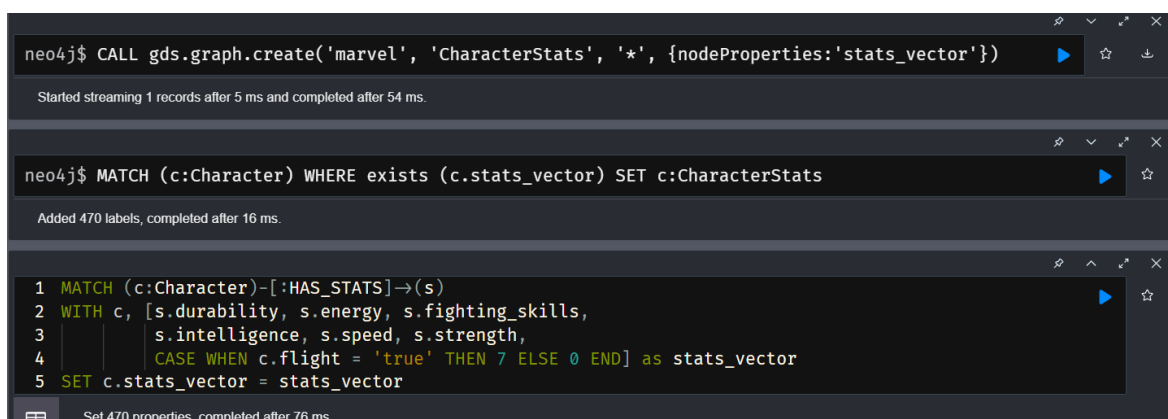
در این بخش گروه های متحد با هم و تعداد اعضایشان را بدست آوردیم. به طور مثال گروهی از متحدان داریم که ۱۹۵ عضو دارد و باقی گروه ها تعداد اعضای کمی دارند.



```
neo4j$ MATCH (c:Character)-[:HAS_STATS]->(stats) RETURN c.name as character, apoc.coll.avg(apoc.m...
```

character	average_stats
"Sasquatch (Walter Langkowski)"	7.0
"Squirrel Girl"	7.0
"Galactus"	7.0
"Deathstrike (Ultimate)"	7.0
"GRAYDON CREED"	7.0
"CHTHON"	7.0
"I INREVEAI ED- GAFA IS HER GREEK NAME GAFA"	7.0

در این بخش، اطلاعات آماری قدرت های شخصیت ها بررسی شده و برای هر یک از آنها میانگین امتیازاتش به دست آمد. قدرتمند ترین های آن ها در جدول بالا مشخص شده اند.



```
neo4j$ CALL gds.graph.create('marvel', 'CharacterStats', '*', {nodeProperties:'stats_vector'})
Started streaming 1 records after 5 ms and completed after 54 ms.

neo4j$ MATCH (c:Character) WHERE exists (c.stats_vector) SET c:CharacterStats
Added 470 labels, completed after 16 ms.

1 MATCH (c:Character)-[:HAS_STATS]->(s)
2 WITH c, [s.durability, s.energy, s.fighting_skills,
3         s.intelligence, s.speed, s.strength,
4         CASE WHEN c.flight = 'true' THEN 7 ELSE 0 END] as stats_vector
5 SET c.stats_vector = stats_vector
Set 470 properties, completed after 76 ms.
```

به کمک دستورات بالا گراف جدیدی آماده می کنیم و برای هر کاراکتر یک وکتور متشکل از میزان قدرت هایش می سازیم و فقط برای قدرت پرواز این تغییر را می دهیم که توانایی پرواز را به قدرت ۷ و عدم توانایی را به صفر مپ می کنیم. این وکتور ویژگی های شخصیت ها می تواند مورد مقایسه قرار گیرد و معیار تصمیم گیری و کلاس بندی بین آن ها شود.

```
neo4j$ CALL gds.beta.knn.mutate('marvel', {nodeWeightProperty:'stats_vector', sampleRate:0.8, top...
```

	createMillis	computeMillis	mutateMillis	postProcessingMillis	nodesCompared	relationshipsWritten	similarityDistribution	co
1	0	166	43	-1	470	7050	{ "p1": 0.2500009536743164, "max": 1.000066757202148, "p5": 0.33333301544189453, "p90": 0.5000028610229492, "p50": 0.5000028610229492, "p95": 1.000066757202148, "p10": 0.33333301544189453, }	

Started streaming 1 records after 1 ms and completed after 207 ms

حال الگوریم KNN را با مقادیر $sampleRate=0.8$ و $topK=15$ اجرا می‌کنیم و نتیجه را در تصویر بالا مشاهده می‌کنیم. پس از آن، الگوریم Louvain Modularity را اجرا می‌کنیم که دسته‌بندی‌ها را با توجه به نتایج KNN انجام می‌دهد.

```
1 MATCH (c:Character)-[:HAS_STATS]->(stats)
2 RETURN c.louvain as community, count(*) as members,
3        avg(stats.fighting_skills) as fighting_skills,
4        avg(stats.durability) as durability,
5        avg(stats.energy) as energy,
6        avg(stats.intelligence) as intelligence,
7        avg(stats.speed) as speed,
8        avg(stats.strength) as strength,
9        avg(CASE WHEN c.flight = 'true' THEN 7.0 ELSE 0.0 END) as flight
```

	community	members	fighting_skills	durability	energy	intelligence	speed	stren
1	134	63	4.301587301587301	4.968253968253969	3.7619047619047623	4.253968253968254	4.095238095238094	4.3650
2	118	47	2.361702127659574	2.0638297872340425	1.6808510638297878	2.6595744680851063	1.7021276595744672	1.8085
3	132	69	4.17391304347826	3.5652173913043486	2.4782608695652177	3.2318840579710137	3.1304347826086967	3.3623
4	393	39	4.358974358974359	5.102564102564103	3.923076923076923	4.128205128205128	3.4871794871794872	4.8461
5	214	97	3.3195876288659796	2.876288659793815	2.0824742268041234	2.9999999999999996	2.5773195876288653	2.5360
6	298	65	4.0769230769230775	3.9692307692307693	3.0615384615384613	3.076923076923077	3.4153846153846144	3.9230

Go to Settings to activate Windows.

در این قسمت پس از مشخص کردن وزن موثر در ایجاد ساختار شبکه و قسمت‌های مختلف آن، مقدار متوسط هر skill در هر community را نمایش می‌دهیم.

```

1 MATCH (c:Character)-[:HAS_STATS]-(stats)
2 RETURN c.labelPropagation as community, count(*) as members,
3        avg(stats.fighting_skills) as fighting_skills,
4        avg(stats.durability) as durability,
5        avg(stats.energy) as energy,
6        avg(stats.intelligence) as intelligence,
7        avg(stats.speed) as speed,
8        avg(stats.strength) as strength,
9        avg(CASE WHEN c.flight = 'true' THEN 7.0 ELSE 0.0 END) as flight

```

	community	members	fighting_skills	durability	energy	intelligence	speed	streng
1	218	138	4.166666666666669	4.797101449275363	3.7463768115942035	3.8115942028985517	3.8840579710144936	4.4275
2	212	171	3.1520467836257318	2.9649122807017556	2.140350877192981	2.982456140350878	2.4093567251462003	2.5321
3	270	15	5.2666666666666675	5.3999999999999995	4.866666666666667	4.799999999999999	4.666666666666667	4.8
4	118	38	4.2631578947368425	3.0789473684210527	2.3157894736842106	3.1315789473684204	3.0526315789473695	3.0263
5	215	16	4.5	3.4999999999999996	2.3749999999999996	3.3125	3.125	4.125
6	73	52	5.846153846153847	6.730769230769231	6.596153846153847	6.115384615384614	6.711538461538463	6.5576

در ادامه برای اینکه ساختار شبکه دارای دسته بندی باشد از Label Propagation استفاده می‌کنیم. در تصویر بالا مقادیر حاصل پس از پیاده سازی این الگوریتم مشاهده می‌شود. تعداد community های بدست آمده از Label Propagation دوبرابر تعداد community های حاصل از Louvain Modularity است.