

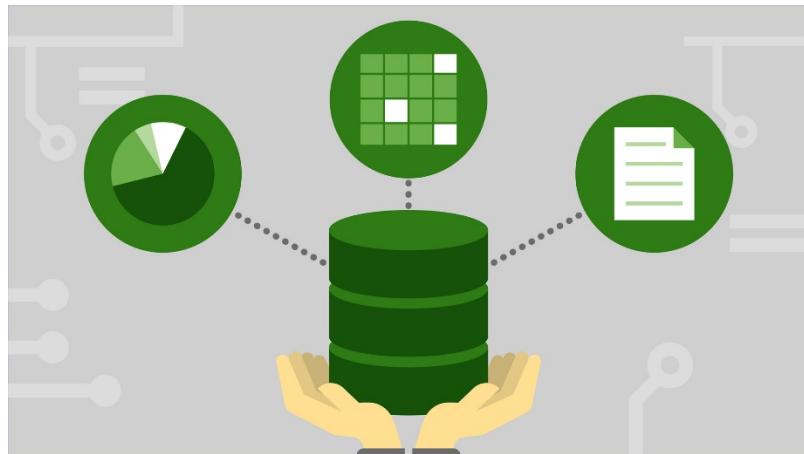
به نام خدا



دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر



آزمایشگاه پایگاهداده

دستورکار شماره ۳

شماره دانشجویی

۸۱۰۱۹۶۴۲۳

3	آشنایی با Neo4j (۱)
3	Neo4j
3	اجزای اصلی Neo4j
3	نصب Neo4j
4	ساخت گره
4	ساخت رابطه
4	ساخت شاخص
5	ساخت محدودیت‌ها
5	انتخاب داده با MATCH
5	وارد کردن داده
6	حذف
7	آشنایی با Cypher (۲)
7	ساخت گراف
7	یافتن بازیگران و فیلم‌ها
7	یافتن الگوهای دو مساله
8	
9	بررسی مقاله‌ی LOTR (۳)
17	مشکلات و توضیحات تکمیلی (۴)
17	آنچه آموختم (۵)

(۱) آشنایی با Neo4j

Neo4j

Neo4j برخلاف پایگاهداده‌های رابطه‌ای، از جداول، ردیف‌ها و ستون‌ها استفاده نمی‌کند. در این پایگاهداده داده‌ها به شکل یک گراف ذخیره و نمایش داده می‌شوند. داده به صورت یک گره و روابط این گره‌ها نمایش داده می‌شود. پایگاهداده‌های دیگر (هم رابطه‌ای و هم غیر رابطه‌ای) معمولاً برای روابط از رفرنس‌ها استفاده می‌کنند و این هزینه زیادی دارد. این پایگاهداده بیشتر بر روی روابط تاکید می‌کند و برای داده‌هایی با روابط داخلی زیاد مناسب می‌باشد. این پایگاهداده حول روابط ساخته می‌شود و نیاز به تعریف foreign key و موارد مشابه نیست.

اجزای اصلی Neo4j

- Nodes

داده‌های اصلی می‌باشند که توسط روابط به یکدیگر متصل می‌شوند. هر گره می‌تواند تعدادی label و properties داشته باشد.

- Relationships

گره‌ها را به یکدیگر وصل می‌کند. بین هر دو گره می‌توان یک یا بیشتر Relationship تعریف کرد.

- Labels

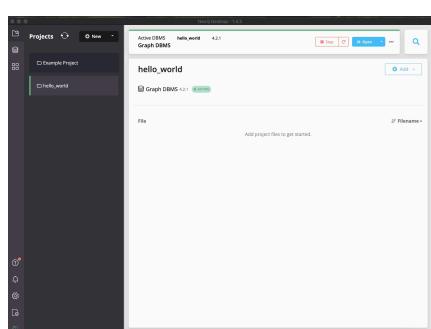
برای گروه‌بندی کردن گره‌ها استفاده می‌شود و هر گره می‌تواند یک یا چند label داشته باشد.

- Properties

ویژگی‌های (attributes) یک گره یا یال می‌باشد و داده‌ها از هر نوعی را می‌تواند ذخیره کند.

نصب Neo4j

در این مرحله، به نصب این پایگاهداده می‌پردازیم.



همانطور که در تصویر مقابل مشاهده می‌شود، یک پروژه‌ی hello_world ساخته می‌شود که یک پایگاهداده به نام Graph DBMS در آن ساخته شده است. حال در ادامه آموزش سایت Quackit انجام خواهد شد. برای مراحل بعد از زبان سایفر استفاده خواهد شد.

ساخت گره

برای ساخت گره، از دستور CREATE استفاده می‌شود. برای مثال در دستور

Properties CREATE (a:Artist { Name : "Lad" }) مربوط به Artist می‌باشد. همچنین قسمت داخل {} مربوط به a نیز صرفا برای استفاده‌های بعدی از این گره در عبارت‌های بعدی استفاده می‌شود. برای نمایش گرهی ساخته شده، باید از دستور {variable name} Return استفاده شود.

در صورت نیاز به ساخت تعداد زیادی گره نیز، می‌توان در دستور CREATE بین پرانتزها، قرارداد و نودها را ساخت.

CREATE (a:Album { Name: "Killers" }), (b:Album { Name: "Fear" })

ساخت رابطه

برای ساخت رابطه میان دو گره، ابتدا از دستور MATCH استفاده می‌کیم تا دو گره مورد نظر را پیدا کیم. می‌توانیم با استفاده از دستور WHERE و قرار دادن یک شرط در آن، گره‌های پیداشده را فیلتر کرده و فقط گره‌هایی که شرط مدنظر را دارند را پیدا کنیم. (اگر متغیرها از قسمت ساخت گره باقی مانده باشند، برای ساخت رابطه بین چند گره، نیاز به انجام این دو قسمت نیست)

حال با دستور (b)->-[r:RELEASED]-[a] CREATE یک رابطه با لیبل RELEASED بین دو گرهی a و b می‌سازیم. متغیر r همان نقش متغیر در قسمت قبل را دارد. مشابه قسمت بالا و دستورات لینک، می‌توان چند رابطه بین چند گره را با استفاده از یک دستور نیز نوشت.

ساخت شاخص

شاخص یک ساختار داده است که سرعت دریافت داده‌ها را افزایش می‌دهد. با استفاده از دستور

CREATE INDEX ON :Album(Name)

می‌توان یک شاخص بر روی ویژگی نام برای تمام گره‌های با لیبل Album قرار داد. همچنین با استفاده از دستور schema: می‌توان شاخص‌های ساخته شده را مشاهده کرد. برای مثال شاخص ساخته شده یک شاخص BTREE بر روی ویژگی نام (و سایر اطلاعات که در تصویر مشاهده می‌شود) است. همچنین می‌توان با استفاده از دستور ... USING INDEX برای استفاده از یک یا چند شاخص موردنظر استفاده کرد. اگرچه استفاده از شاخص باعث افزایش سرعت خواندن داده‌ها می‌شود، اما Neo4j برای اینکار یک نسخه تکراری از داده‌ها را نگهداری می‌کند و بنابراین استفاده از شاخص می‌تواند فضای بیشتری را اشغال کند و نوشتن در دیسک را کندتر کند.

ساخت محدودیت‌ها

در Neo4j دو نوع محدودیت می‌توان ساخت. Uniqueness Constraint که بیان می‌کند مقدار یک ویژگی باید یکتا باشد و دو گره با یک لیبل یا کلا، نباید مثلاً نام یکسانی داشته باشند.

Property Existence Constraint که بیان می‌کند یک ویژگی باید برای تمام گره‌ها وجود داشته باشد.

برای ساخت محدودیت از این دستور می‌توان استفاده کرد:

```
CREATE CONSTRAINT ON (a:Artist) ASSERT a.Name IS UNIQUE
```

یک محدودیت از نوع اول بر روی ویژگی نام تمام گره‌های با لیبل Artist قرار می‌دهیم. مشابه ساخت شاخص می‌توان با :schema محدودیت‌های ساخته شده را مشاهده کرد. برای تست این محدودیت، دستور مشخص شده در بالا دوبار اجرا شده است (برای خوانایی تصویر، بار اول حذف شده است) و پایگاه داده اجازه ثبت این گره را نمی‌دهد.

MATCH داده با

با استفاده از این دستور، می‌توان داده‌هایی که تحت شرایط خاصی می‌باشند را بدست آورد و با return آنها را بازگرداند.

```
MATCH (p:Person {Name: "Devin Townsend"})
```

```
RETURN p
```

برای مثال در این دستور، گره‌هایی با لیبل Person انتخاب می‌شوند و آنها که نامشان Devin Townsend می‌باشد، چاپ می‌شوند. همچنین با زدن قسمت پایین بر روی گره، می‌توان گره‌هایی که به این گره وصل می‌باشند را مشاهده کرد. از دستور MATCH می‌توان برای محدود کردن یالهای نیز استفاده کرد. برای مثال در عبارت

```
MATCH (a:Artist)-[:RELEASED]->(b:Album)
WHERE b.Name = "Heavy as a Really Heavy Thing"
```

```
RETURN a
```

تمام گره‌های با لیبل Artist که یک رابطه از نوع RELEASED با گره‌های با لیبل Album دارند و نامشان Heavy as a Really Album می‌باشد را برمی‌گرداند. همچنین با دستور MATCH (n) RETURN n می‌توان تمام گره‌ها و یالهای بین آنها را نمایش داد. در صورتی که گراف بسیار بزرگ باشد، این عملیات بسیار سنگین خواهد بود و بهتر است با استفاده از LIMIT فقط تعداد مشخص شده‌ای را بازگرداند.

واردکردن داده

می‌توان داده‌ها را از یک فایل csv. وارد پایگاه داده‌ی Neo4j کرد. با استفاده از دستورات

```
LOAD CSV FROM 'https://www.quackit.com/neo4j/tutorial/genres.csv' AS line
CREATE (:Genre {GenreId: line[0], Name: line[1]})
```

ابتدا فایل از سایت موردنظر دریافت می‌شود و سپس گره‌ها ساخته می‌شند. لیبل تمام این گره‌ها Genre می‌باشد. هر گره دو ویژگی Name و GenreId را دراد که به ترتیب ستون اول و ستون دوم می‌باشند.

در صورتی که فایل .csv شامل header باشد، می‌توان بجای [i] از Line.colName استفاده کرد. همچنین برای دریافت فایل‌های بسیار بزرگ، PERIODIC COMMIT کمک‌کننده می‌باشد و بعد از دریافت تعدادی ردیف، آنها را ثبت می‌کند و ردیف‌های بعدی را دریافت می‌کند. این مقدار قابل تنظیم می‌باشد.

حذف

برای حذف قسمت‌های مختلف که در بالا اضافه شد، می‌توان از دستورات زیر استفاده کرد.

دستور DROP INDEX ON :Album(Name) شاخصی برروی گره‌های با لیبل Album می‌باشد را حذف می‌کند.

دستور DROP CONSTRAINT ON (a:Artist) ASSERT a.Name IS UNIQUE مشابه بالا، محدودیت مشخص شده را حذف می‌کند.

دستور MATCH (a:Album {Name: "Killers"}) DELETE a گرهی با لیبل Album و نام Killers را حذف می‌کند. مشابه اضافه کردن می‌توان چند گره را همزمان حذف کرد.

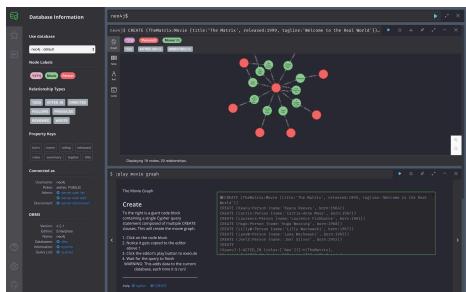
برای حذف رابطه،

```
MATCH ()-[r:RELEASED]-()
DELETE r
```

تمام رابطه‌های با لیبل RELEASED را حذف می‌کند. می‌توان در پرانتزهای دو طرف، ویژگی‌های از گره‌ها مانند لیبل آنها را قرار داد تا يالهای محدودتری حذف شوند. همچنین برای حذف یک گره، ابتدا باید تمام يالهای آن حذف شوند. برای اینکار می‌توان از DETACH DELETE استفاده کرد.

برای حذف تمام پایگاهداده می‌توان از دستور MATCH (n) DETACH DELETE n استفاده کرد.

۲) آشنایی با Cypher



ساخت گراف

مطابق راهنما، ابتدا گراف ساخته می‌شود. در این گراف، گره‌های قرمز لبیل Person و گره‌های سبز لبیل Movie را دارند. همچنین يالهایی از جنس ACTED_IN، WROTE و ... نیز وجود دارد. در پایان کوئری اضافه شده، اطلاعات Tom Hanks، فیلم‌هایی که در آن نقش ایفا کرده است و کارگردان آن فیلم‌ها نمایش داده شده است.



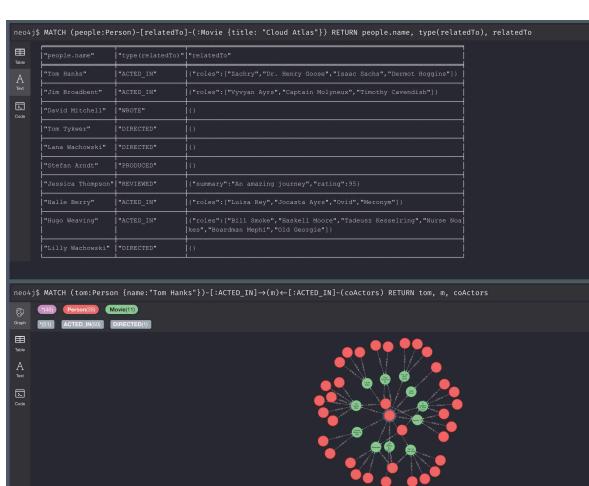
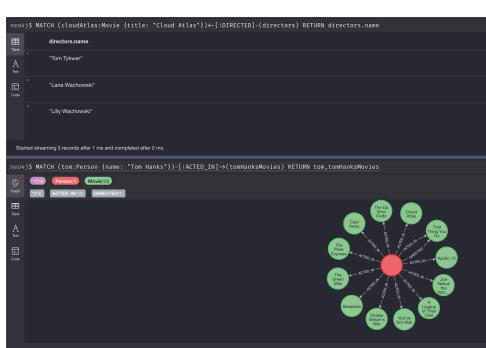
یافتن بازیگران و فیلم‌ها

مطابق تصویر، در کوئری اول، با استفاده از دستور MATCH گره‌های با لبیل PERSON و در WHERE قسمت های بالایی توضیح داده شد) برای بدست آوردن یک فیلم خاص و در کوئری سوم، نام ۱۰ گره با لبیل Person را برمی‌گردانیم. دقت شود چون یک ویژگی را RETURN کرده‌ایم، نمایش گرافی وجود ندارد. برای قسمت پیدا کردن فیلم‌های دهه ۱۹۹۰، مشابه بالا پیش می‌رویم و در قسمت WHERE باید فیلم‌هایی را که سال انتشار آنها بین ۱۹۹۰ تا ۲۰۰۰ می‌باشد بدست بیاوریم.

یافتن الگوهای

در کوئری اول، گره‌هایی که با نام Tom Hanks رابطه‌ی ACTED_IN دارند با آنها بازی کرده است (باز گردانده می‌شود). از دستور MATCH استفاده شده است و نوع رابطه که در [] می‌باشد ACTED_IN قرار داده شده است.

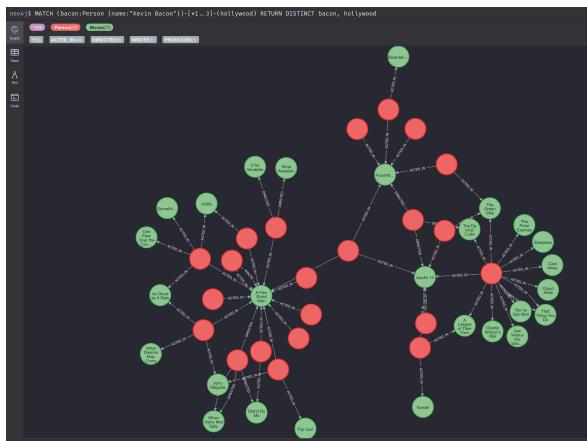
در کوئری بعدی نیز مشابه بالا، نام گره‌هایی که رابطه DIRECTED با گرهی با ویژگی نام Cloud Atlas دارند برگردانده می‌شود که ۳ نام چاپ شده است. جهت فلش در کوئری اول، به سمت راست می‌باشد و در کوئری دوم به سمت چپ (در حین ساخت رابطه این جهت‌ها در نظر گرفته شده است و دلیل واضحی دارد)



در کوئری اول، همکاران Tom Hanks نمایش داده شده‌اند. برای این منظور، گره‌هایی که رابطه ACTED_IN با آنها دارد و گره‌هایی که با آن گره‌ها رابطه ACTED_IN دارند (همکاران Tom

در فیلم‌های مختلف) برگردانده می‌شوند. در کوئری دوم، گره‌های با لیبل Person که هر رابطه‌ای با گرهی با لیبل Movie و نام Cloud دارند بازگردانده می‌شوند. در عبارت RETURN نوع رابطه برگردانده شده است و خروجی مشابه تصویر می‌باشد. مشخص است نام شخص، نوع رابطه و ویژگی‌های آن رابطه برگردانده شده است.

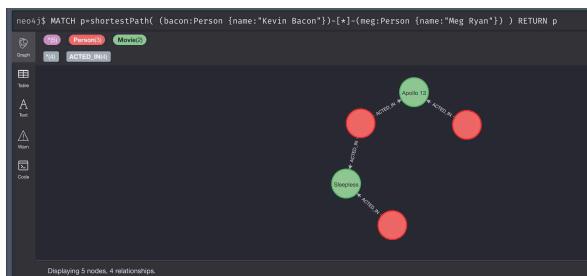
دو مساله



می‌توان با استفاده از سینتکس $(b) \rightarrow [*i...j] - (a)$ ، یک رنج فاصله بین دو نود تعریف کرد. درواقع در عبارت بیان شده، تمام گره‌ها و یالهایی که در مسیری به طول i ای زاز a به b می‌باشند را بدست می‌آورد. در کوئری رو به رو، تمام گره‌هایی (با هر لیبلی مانند فیلم یا Kevin Person) که فاصله‌ی یک تا ۳ با گرهی با لیبل Kevin نام دارند به همراه یالهای بینشان بدست آمده‌اند. گرهی Kevin همان گرهی قرمز مرکز می‌باشد.

برای پیدا کردن کوتاهترین مسیر، j به طور پیشفرض از جستجوی bfs دوطرفه (از مبدا و مقصد جستجو شروع می‌شود و محل برخورد بدست می‌آید) استفاده می‌شود. در صورتی که مساله شرطی داشته باشد که نیاز باشد تمام مسیر قبل از انتخاب بررسی شود (برای مثال یک گره با ویژگی نام $omid$ وجود داشته باشد) روش فوق، لزوماً با موفقیت اجرا نمی‌شود. در این صورت از dfs استفاده می‌شود.

همواره ابتدا روش اول اجرا می‌شود و در صورت شکست، روش دوم اجرا می‌شود. در کوئری مقابل، ابتدا الگوریتم shortestPath اجرا می‌شود. در کوئری آن که یک گراف می‌باشد بازگردانده می‌شود. در تابع shortestPath، از گرهی با لیبل Person با ویژگی نام Kevin، تا گرهای با همان لیبل با ویژگی نام Meg Ryan و عدم اهمیت یالها در مسیر، کوتاهترین مسیر محاسبه می‌شود. همانطور که مشاهده می‌شود خروجی گرافی به طول ۴ از Kevin Bacon به Meg Ryan می‌باشد و این مسیر بین این دو گره می‌باشد.



داسه باشد که نیاز باشد تمام مسیر قبل از انتخاب بررسی شود (برای مثال یک گره با ویژگی نام $omid$ وجود داشته باشد) روش فوق، لزوماً با موفقیت اجرا نمی‌شود. در این صورت از dfs استفاده می‌شود. همواره ابتدا روش اول اجرا می‌شود و در صورت شکست، روش دوم اجرا می‌شود. در کوئری مقابل، ابتدا الگوریتم shortestPath اجرا می‌شود. در کوئری آن که یک گراف می‌باشد بازگردانده می‌شود. در تابع shortestPath، از گرهی با لیبل Person با ویژگی نام Kevin، تا گرهای با همان لیبل با ویژگی نام Meg Ryan و عدم اهمیت یالها در مسیر، کوتاهترین مسیر محاسبه می‌شود. همانطور که مشاهده می‌شود خروجی گرافی به طول ۴ از Kevin Bacon به Meg Ryan می‌باشد و این مسیر بین این دو گره می‌باشد.



در پایان نیز همانطور که در قسمت‌های قبل درباره DETACH توضیح داده شد، گراف با دستور DETACH DELETE حذف می‌شود. برای اطمینان، تعداد کل گره‌های گراف را بررسی می‌کنیم که ۰ را نشان می‌دهد. پس گراف با موفقیت حذف شد.

LOTR مقاله‌ی بررسی (۳)

این مقاله داده‌های مربوط به فیلم ارباب حلقه‌ها را از WikiData دریافت می‌کند و داده‌ها را در Neo4j ذخیره می‌کند. پس از ساخت پایگاه داده، یک تحلیل داده بر روی گراف انجام می‌شود. همچنین چند الگوریتم گراف بر روی داده‌ها انجام می‌شود و در نهایت با استفاده از bloom تصویر سازی‌های مرتبط انجام می‌شود.

مراحل این مقاله به صورت زیر می‌باشد و هر کدام، در قسمت مربوطه به تفصیل توضیح داده خواهد شد:

1. دریافت داده از WikiData و ساخت گراف در Neo4j
 2. جستجو و بررسی گراف ساخته شده
 3. تعیین کردن مقادیر تعیین نشده
 4. بررسی بیشتر گراف حاصل از مرحله ۳
 5. بررسی قسمت های همبند ضعیف در گراف
 6. بررسی شاخص betweenness
 7. تصویرسازی گراف با bloom

در این مقاله، یک Social Network از کاراکترها و روابطی مانند همسر، خواهرزاده، پدر و دشمن وجود دارد. همچنین اطلاعاتی مانند نژاد، کشور و گروههایی، که کاراکترها به آنها تعلق دارند نیز در دسترس می‌باشد.

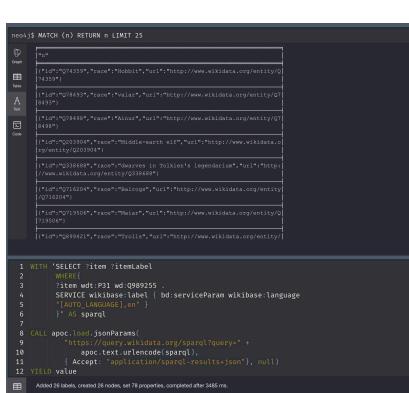
همانطور که گفته شد، برای دریافت داده از API های WikiData استفاده می شود. WikiData یک پایشگاه داده (Knowledge Base) می باشد که داده های ساختاربندی شده را جمع آوری می کند.

ابتدا داده‌های با لیبل نژاد را دریافت می‌کنیم.
برای دریافت داده‌ها procedure ای به نام apoc.load.html را صدای زنیم. APOC یک کتابخانه اضافه شده به Neo4j می‌باشد و صدھا procedure و تابع کاربردی را فراهم می‌کند. برای صدا زدن یک تابع از آن به این صورت عمل می‌کنیم:

```
CALL <package>. <subpackage>. <procedure>(<argument1>, <argument2>, ...);
```

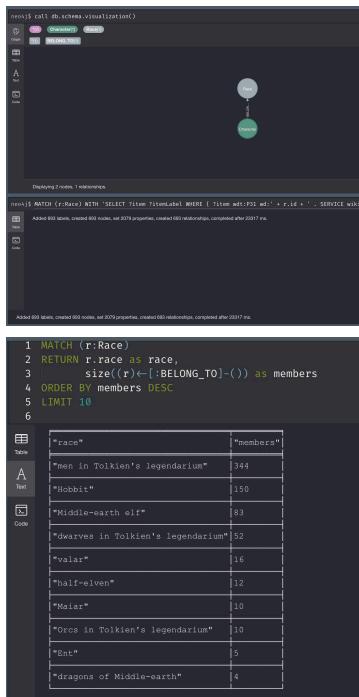
که package همان APOC می باشد. subpackage و در نهایت نام روال موردنظر را مم نویسیم و در پرانتز آرگومان های لازم را وارد می کنیم.

در کد مقابل، ابتدا کوئیری مربوط به SPARQL را برای دریافت یک item خاص از ویکیپدیا با دستور with می‌سازیم. با دستور apoc.load.jsonParams درخواست خود را به WikiData ارسال می‌کنیم و داده‌ها را دریافت می‌کنیم و نام آن را value می‌گذاریم. با دستور UNWIND (تبديل یک لیست به ردیف‌های متوالی) را اجرا می‌کنیم و پس از مدتی ترتیب‌سازی، توسط دستور WITH، گرهای ساخته شده را با نام آن داشت. با این گذشتگی، می‌توانیم از داده‌ها برای این ایجاد کرد.



دستورکار آزمایشگاه یا یگاه داده

ویزگی های race, url و id را قرار می دهیم. در تصویر بالا مشخص است که ۲۶ گره اضافه شده است. در قسمت بالای تصویر، قسمتی از مقادیر هر گره نمایش داده شده است.



در این قسمت، با دستور MATCH، به ازای هر نژاد(گره با لیبل race) تمام کاراکترهای متعلق به خود را مشابه بالا دریافت می‌کنیم. در انتهای، گره‌هایی با لیبل Charachter ایجاد می‌کنیم و ویژگی‌های url و name را برای آن قرار می‌دهیم. همچنین رابطه BELONG_TO را با نژاد ایجاد می‌کنیم. همانطور که مشاهده می‌شود، ۶۹۳ گره جدید و رابطه جدید به گراف اضافه می‌شود. با استفاده از call db.schema.visualization() شما گراف را مشاهده می‌کنیم که تعدادی گره از نوع کاراکتر، رابطه‌ی BELONG_TO با تعدادی گره از نوع Race دارند.

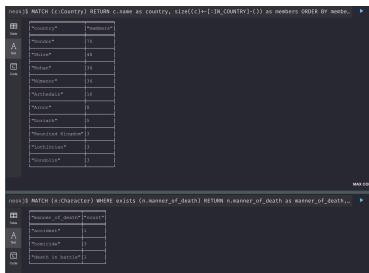
در این قسمت، نام نژاد گرهای Race و تعداد گرهایی را که رابطه‌ی BELONG_TO با آن دارند برمی‌گردانیم. با دستور ORDER BY خروجی را به صورت نزولی(DESC) برمی‌گردانیم. (در واقع ۱۰ نتیجه‌ی کاراکتر متعلق به آنها می‌باشد را برمی‌گردانیم)

در کوئری بعد، تلاش می‌شود تا گراف مسئله بزرگتر شود. با ساخت query مخصوص در ابتدا و مشابه قسمت اول توضیحات، داده‌های خاصی از API دریافت می‌شوند. با استفاده از دستور SET، می‌توان برای یک گره ساخته شده، ویژگی‌های جدیدی ایجاد کرد. در اینجا، جنسیت و نحوه مرگ را به گره‌های Character اضافه می‌کنیم. همچنین با استفاده از دستور MERGE، تعدادی گره Country ایجاد می‌کنیم و با گره‌ی Character متناظر، رابطه‌ی IN_COUNTRY را قرار می‌دهیم. ممکن



است country برای برخی شخصیت‌ها مقداری نداشته باشد. رای جلوگیری از رخدادن مشکل در Neo4j، این قسمت را داخل یک foreach قرار می‌دهیم و به ازای هر ردیف از row، در صددت null نهادن، مقدار ۱ و د، غیر اینصهارت مقدار، ۰ یا مگ دانیم. حال آگ مقدار باشگشت.

۱ باشد، گره‌های اشاره‌شده ساخته می‌شوند. همانطور که در تصویر دیده می‌شود، در نهایت ۱۸ گرهی جدید (مربوط به لیبل Country) و ۲۳۴ اطه (مربوط به IN COUNTRY س.) کارکرده‌اند و کشو (ها) به گرف اضافه شده است.



در این قسمت، با استفاده از دستور MATCH و شرط داخل WHERE (گره حتما باید ویژگی نحوه مرگ را داشته باشد). انواع روش‌های مرگ را در گراف بررسی می‌کنیم. کلا ۳ روش وجود دارد و تعداد کمی از آنها هم وجود دارد (دلیل این مورد این است که در LOTR کشtar زیادی وجود ندارد) در کوئری دوم، تعداد گره‌هایی که با گره Country رابطه IN_COUNTRY دارند، چاپ شده‌اند. Gondor، پیشترین تعداد کاراکتر را دارد.

دستور کار آزمایشگاه پایگاهداده

اطلاعات کشور، صرفا برای ۲۳۶ کاراکتر موجود می‌باشد. می‌توان برای بر کردن کاراکترهای ناقص، فرض کرد برادرها و خواهرها متعلق به یک کشور می‌باشند. برای پیاده‌سازی چنین فرضیاتی، روابط خانوادگی کاراکترها را از WikiData دریافت می‌کنیم و گراف را گسترش می‌دهیم و روابط HAS_FATHER، HAS_MOTHER، SPOUSE و SIBLING را اضافه می‌کنیم. ۳ گره و ۹۷۹ رابطه به گراف اضافه می‌شود.

```

33 FOREACH(ignoreNullColumn($row, 'father')) as father
34   MERGE (c:Character{url:$row['character']})-[:father]->($father)
35   MERGE ($father)-[:HAS_FATHER]->($row)
36 FOREACH(ignoreNullColumn($row, 'mother')) as mother
37   MERGE (c:Character{url:$row['character']})-[:mother]->($mother)
38   MERGE ($mother)-[:HAS_MOTHER]->($row)
39 FOREACH(ignoreNullColumn($row, 'sibling')) as sibling
40   MERGE (c:Character{url:$row['character']})-[:sibling]->($sibling)
41   MERGE ($sibling)-[:HAS_RELATIVE]->($row)
42 FOREACH(ignoreNullColumn($row, 'spouse')) as spouse
43   MERGE (c:Character{url:$row['character']})-[:spouse]->($spouse)
44   MERGE ($spouse)-[:SPOUSE]->($row)
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
756
756
757
758
758
759
759
760
761
762
763
764
765
765
766
767
767
768
769
769
770
771
772
773
774
775
775
776
777
777
778
779
779
780
781
782
783
784
784
785
786
786
787
787
788
788
789
789
790
791
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
15
```

دستورکار آزمایشگاه پاییگاه داده

حال در کوئری زیر، اگر character ای که متعلق به کشوری باشد، sibling ای داشته باشد که متعلق به کشوری نباشد، بین آن sibling و کشور، رابطه‌ی IN_COUNTRY قرار می‌دهیم. با استفاده از این کوئری، ۴۸ رابطه‌ی دیگر به گراف اضافه شد.

```
MATCH (country)-<[:IN_COUNTRY]->(s:Character)-[:SIBLING]->(t:Character) WHERE NOT (t)-[:IN_COUNTRY]->()
MERGE (t)-[:IN_COUNTRY]->(country)
```

حال در این قسمت، مجدداً با دریافت داده‌های مربوط به شغل، زبان، گروه و رخدادهای مرتبط با هر کاراکتر، گراف را تکمیل‌تر می‌کنیم. در قسمت پایین تصویر مشاهده می‌شود ۳۱ گره و ۲۲۹ رابطه جدید در گراف ایجاد شده‌است. در تصویر بالا نیز شمای کلی گراف نمایش داده شده‌است که مشاهده می‌شود گره‌ای از نوع Character وجود دارد که روابطی با گره‌های از نوع دیگر دارد.

حال نام گروه، تعداد اعضاء، نام حداکثر ۳ عضو و ۳ شغل در جدول مقابل چاپ می‌شود. اطلاعات جالبی می‌توان از این جدول بدست آورد. برای مثال، Gandalf در گروه‌های مختلفی عضو است. (عبارت [۳..] یعنی حداکثر ۳ تا)

در این قسمت آخرین مرحله از تکمیل گراف را انجام می‌دهیم و دشمنان هر Character و هایی که او دارد را از WikiData بدست می‌آوریم. ۳۸ گره و ۱۲۶ رابطه جدید ایجاد می‌شود. در این تصویر که گراف نهایی است، در سمت چپ، لیبل تمام گره‌ها و یالها نوشته شده‌است و مدل داده‌ها به صورت تصویر مقابل می‌باشد.

حال به بررسی دو مورد درباره دشمنی در این گراف می‌پردازیم: در کوثری بالا، دشمنی با فامیل فامیل بررسی شده است، با برادر همسرش دشمن می‌باشد. عبارت Varda ۲.* به معنای حداکثر اندازه‌ی مسیری به طول ۲ می‌باشد.

در کوئیری پایین تصویر، a و b در نظر گرفته شده است که اولاً رابطه دشمنی با یکدیگر داشته باشند و
شانیاً یک نسبت فامیلی مادری، پدری، برادر خواهری و یا همسری داشته باشند. Morgoth و
Manwe دو برادر ممکن باشند که با یکدیگر دشمن ممکن باشند.

در ادامه یکسری کار تحلیل داده بر روی گراف با استفاده از کتابخانه‌ی Graph Data Science انجام می‌شود. این کتابخانه پیاده‌سازی بهینه و موازی Neo4j را بر عهده Cypher توسط روال‌های Community Centrality و تشخیص گوگریتم‌های می‌باشد. الگوریتم‌های نظری مسیریابی، پیاده‌سازی شده‌اند. همچنین الگوریتم‌هایی برای استفاده مناسب برای مسایل Data Science مانند تخمین Link و شباهت یالهای وزن دار را پیاده‌سازی کرده‌است. برای استفاده از توابع آن، مشابه APOC استفاده می‌کنیم و نام package gds است. الگوریتم‌های گراف بر روی projection یک گراف، یک نما (view) بر روی گراف ذخیره شده می‌باشد و صرفا اطلاعات و ویژگی‌هایی که ارزش تحلیل دارند نگهداری می‌شوند و در مموری نگهداری می‌شود. catalog گراف نیز یک مفهوم در GDS می‌باشد که اجازه مدیریت گراف‌های مختلف توسط نام آنها را می‌دهد. با توجه به توضیحات بالا، به برسی مقاله‌ی پردازیم.

دستورکار آزمایشگاه پاییگاه داده

```
neo4jjs$ CALL gds.graph.create('family','Character', ['SPOUSE','SIBLING','HAS_FATHER','HAS_MOTHER'])
```

ابتدا با استفاده از دستور بالا، یک project از گراف با دستور `gds.graph.create` ایجاد می‌کنیم. ورودی اول نام گراف جدید، ورودی دوم گره‌ها و ورودی سوم پاله‌ها می‌باشد. در دستور بالا، گرافی از گراف اصلی به نام `family` که صرفا شامل گره‌های `Character` و یالهای مشخص شده در تصویر ایجاد می‌شود. این گراف صرفا اطلاعات خانوادگی را نگهداری می‌کند. در جدول خروجی مشاهده می‌شود ۶۹۹ گره و ۱۰۵۴ یال در گراف `family` وجود دارد.

1.1 Weakly Connected Component

گراف همبند، گرافی می باشد که بین هر دو گره از آن، یک مسیر وجود داشته باشد. گراف جهت دار همبند ضعیف، گراف جهت داری می باشد که در صورت در نظر نگرفتن جهت یال ها، همبند باشد. WCC معمولا در تحلیل اولیه و برای درک ساختار گراف استفاده می شود. با استفاده از

gds.wcc.stat کوئری، نتایج اجرای الگوریتم family بر روی گراف باید بررسی شود. چند مقدار در کوئری بازگردانده می‌شود که با اعداد، آنها را توضیح می‌دهم. تعداد بخش‌های همبند، ۱۴۵ می‌باشد. یعنی ۱۴۵ بخش که هیچ یالی با یکدیگر ندارند و بین خودشان قطعاً مسیری وجود دارد. حدود ۷۵ درصد این بخش‌ها، شامل یک گره می‌شوند و افراد تنها و مستقل می‌باشند. حدود ۹۰ درصد، تا ۳ عضو

دارند و حدود ۱۰ درصد بیشتر از این مقدار عضو دارند. میانگین گرهای هر بخش، ۴.۷۶ می باشد و بزرگترین یux، ۳۲۴ گره (کارکتر) دارد.

با دستور familyComponent.gds.write('wcc', gds) می‌توان خروجی الگوریتم را در یک متغیر مانند Neo4j ذخیره کرد. همانطور که مشاهده می‌شود، اطلاعاتی مانند زمان نوشته شده، تعداد کامپوننت‌ها و اکسترموم‌ها و سایر موارد در خروجی بیان شده است.

حال در کوئری مقابل، ۵ خانواده بزرگ را بررسی می‌کنیم. MATCH همانند MATCH می‌باشد و تفاوت‌شان در این است که برای قسمت‌هایی که تطابق ندارد، null برミ‌گرداند. در خروجی، شناسه کامپوننت، اندازه، ۳ عضو تصادفی خانواده و نژاد اعضای خانواده بررسی می‌شود. [..۳]، ۳ عضور تصادفی را برミ‌گرداند. همانطور که در بالا دیده شد، بزرگترین خانواده ۳۲۴ عضو دارد. تعدادهای اعضا و تعداد آنها در جدول بالا مشاهده می‌شود.

در این قسمت، در بزرگترین خانوداده، افرادی را که همسرانی از نژادهای دیگر دارند، به همراه نژاد خود و همسرشان در خروجی چاپ می‌کنیم. در خط دوم کوئری، شناسه کامپوننت که از بالا بدست آمد را فرادر داده‌ایم و در خط ۴، شرط گذشته شده است که نژاد آنها متفاوت باشد.

دستورکار آزمایشگاه یا بیگانه داده

	character	descendants
1	"DrEuch"	11
2	"Brendi"	10
3	"Ewing"	10
4	"Eros"	9
5	"Elrend"	2

در کوئیری مقابل، افرادی از نژاد half-even (خط دوم کوئیری) که طولانی‌ترین نسل را از خود دارند بررسی می‌کنم. در خط سوم، گرهای end گرهای هستند که حداقل زنگیر روابط پدری یا مادر داشتن آنها با ۲۰ رسید و end دیگر پدری یا مادر گره دیگری نیست. خروجی را مرتب کرده و ۵ گرینه برتر را بر می‌گردانم. شخص Eluchil Dior نسلی به طول ۱۱ دارد و با بررسی این شخصیت، متوجه می‌شویم در زمان‌های قدیم به دنیا آمده است. اطلاعات ۴ شخص دیگر نیز قابل مشاهده می‌باشد.

1.1 Betweenness Centrality

شایعه (centrality) در یک گراف، اهمیت گردها در یک گراف را تعیین می‌کند (ابن احمدیت در روش‌های مختلف تعاریف مختلفی دارد)

```

1 gds.betweennessStream()
2 nodeQuery("MATCH (n:Character) WHERE n.familyComponent = 166
3 OPTIONAL MATCH (n)-[r:HAS_PARENTS]->(p)
4 relationshipQuery("MATCH (s:Character)-[:HAS_FATHER|MISSES_MOTHER|SPOUSE|SUBSISTS]-(t:Character)
5 WHERE r.type = "MISSES_MOTHER" RETURN id(s) as source, id(t) as target",
6 validateRelationships(true))
7 VIELD modelId, score
8 RETURN gds.util.asNode(modelId).name as character,
9 score
10 ORDER BY score DESC
11 LIMIT 10
```

character	score
"Arawn"	60099
"Argonai"	441930.0
"Athomir I"	43984.0
"Aesir"	422524.0
"Argonai"	41940.0
"Athomir I"	41802.0
"Aesssel"	41790.0
"Aesshel I"	40794.0
"Elessor"	40483.102742857174
"Aeswyn"	40360.0

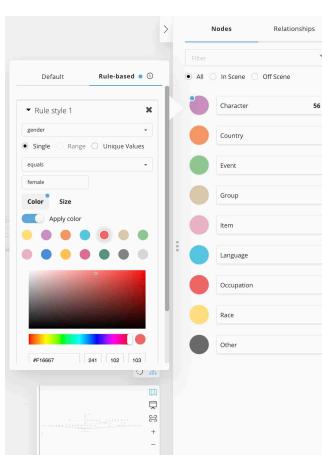
betweenness centrality به هر گره یک عدد نسبت می‌دهد. این عدد برای گره v برابر است با نسبت shortest path های بین هر دو گره که از v می‌گذرند به تمام shortest path ها. برای محاسبه این شاخص، از gds.betweenness.stream استفاده می‌شود. باعث می‌شود نتیجه‌گیریم به صورت جریانی از داده‌ها بازگردانده شود. در مثال مقابل، از کامپوننت خانواده ۳۲۴ نفری که پرجمعیت‌ترین بود استفاده شده است. خروجی این تابع، از دو ستون شناسه گره و امتیاز گره تشکیل شده است. خروجی را بر اساس امتیاز مرتب می‌کنیم و ده گره با بزرگترین مقدار این شاخص مطابق تصویر مقابل می‌باشد. احتمالاً در صورت بررسی LORT، متوجه می‌شویم این افراد بیشترین ارتباط را بین اعضای مختلف خانواده خود ایجاد کرده‌اند.

1.1 Neo4j Bloom

Neo4j Bloom ابزاری ساده برای کاربردهای جستجوی گراف می‌باشد و به کاربران اجازه می‌دهد بدون استفاده از Cypher، اطلاعاتی را از گراف کسب کنند. Neo4j را مطابق تصویر مقابل باز می‌کنیم و همانطور که مشاهده می‌شود، در سمت راست، اطلاعات گره‌ها مشاهده می‌شود (در صورت انتخاب tab روابط، آنها نمایش داده می‌شوند).

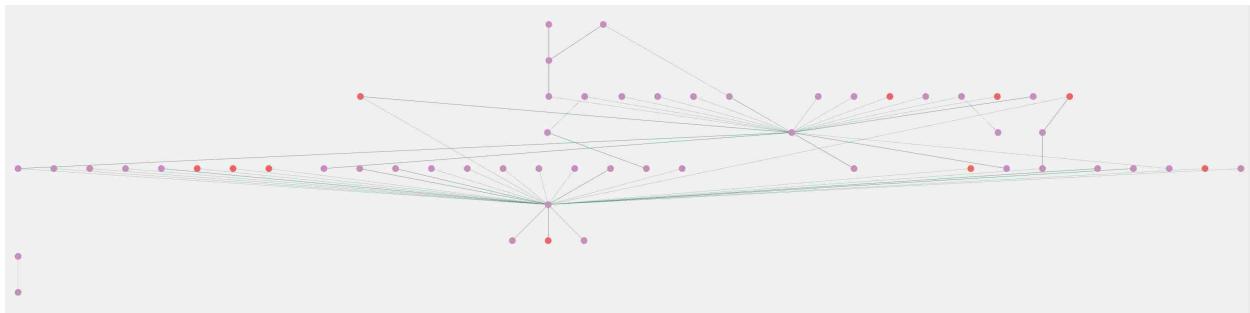
همچنین می‌توان رنگ یالها و گره‌ها را در این ابزار تغییر داد. در صورت نیاز به وجود چند رنگ برای یک لیبل، می‌توان از Tab Rule-based استفاده کرد. همانطور که مشخص است، گره‌های Character با جنسیت زن را قرمز رنگ کرده‌ایم.

می‌توان از این‌بار Near-natural language search را ای جستجو و بررسی، گرف استفاده کرد.

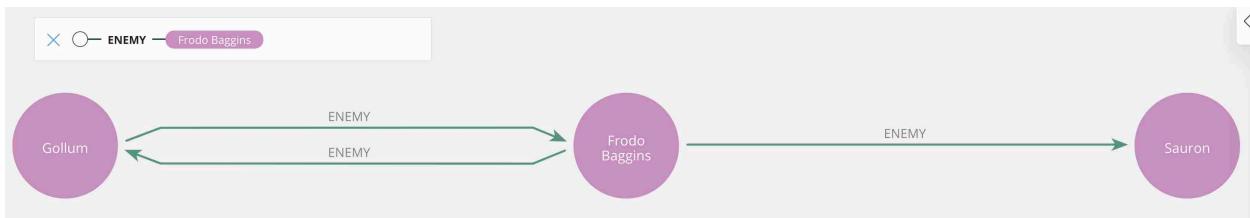


دستورکار آزمایشگاه پایگاهداده

در صورتی که عبارت Enemy را در نوار جستجو وارد کنیم، تصویر زیر مشخص خواهد شد که شامل گرههای میباشد که رابطه دشمنی با یکدیگر دارند. همانطور که مشاهده میشود، دو گره (Sauron و Margoth) محوریت زیادی دارند و بنابراین بیشترین دشمن را در میان کاراکترهای گراف دارند. همچنین دو گره تنها در سمت چپ پایین تصویر دیده میشوند که فقط با یکدیگر دشمن اند و با دیگران رابطه دشمنی ندارند.

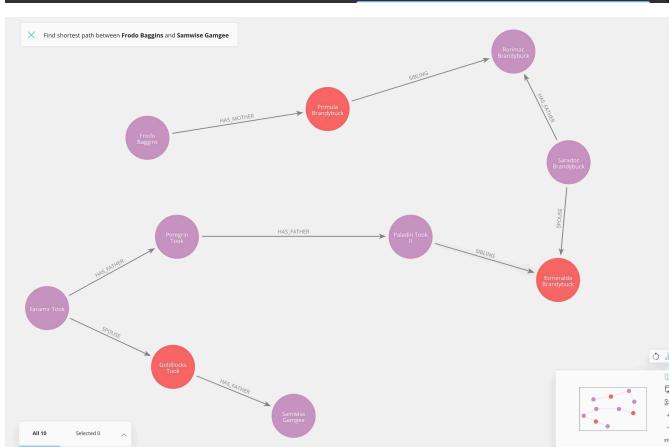


از ابزار Near-natural language search میتوان استفاده های پیشرفته تری نیز کرد. برای مثال با وارد کردن عبارت Enemy of Frodo، دشمنان Frodo را مطابق گراف زیر مشاهده کرد.



قسمت Bloom search phrase در Neo4j، اجازه اضافه کردن جستجوهای دستی و خودنویس را می دهد. با علامت \$ میتوان پارامتر ها را به کوئری اضافه کرد. کوئری نوشته شده، ابتدا گره با نام source و سپس target را بدست می آورد. سپس با استفاده ازتابع shortest path، کوتاه ترین مسیر بین این دو گره را باز می گردد.

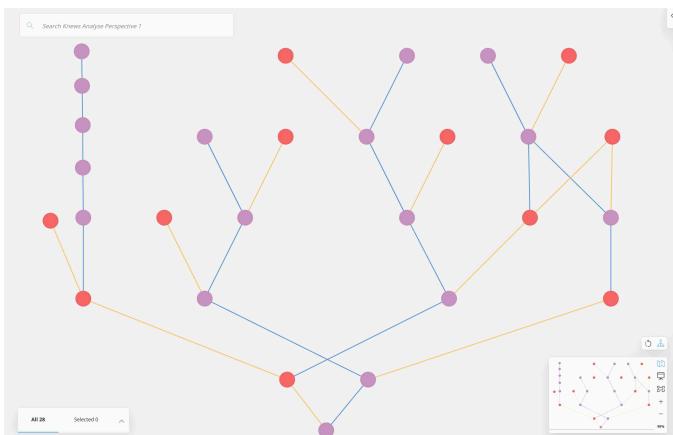
همانطور که مشاهده میشود، search phrase موردنظر نوشته شده است و کافی است در قسمت نوار جستجو، بجای source و target، نام کاراکترها گذاشته شود. توضیح این جستجو و در پایین، کوئری متناظر نوشته شده است.



همانطور که مشاهده میشود، کوتاه ترین مسیر بین Frodo و Samwise Gamgee و Baggins نشان داده شده است. اندازه این مسیر ۱۰ می باشد.

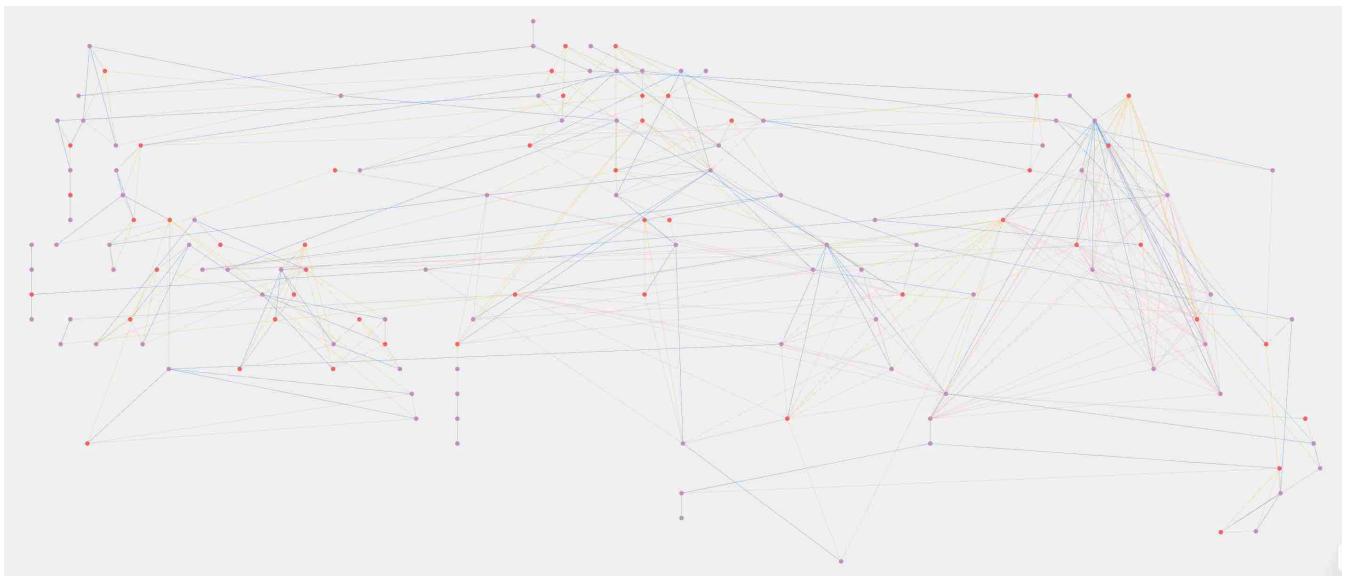
به طور کلی، این ویژگی در جستجوهای محلی بسیار کمک کننده می باشد.

در انتهای، دو کوئری دیگر می‌نویسم.



کوئری اول، صرفا از رابطه HAS_FATHER و HAS_MOTHER استفاده می‌کند و اجداد نام داده شده را نمایش می‌دهد. خروجی آن به صورت مقابل می‌باشد. یالهای HAS_MOTHER با رنگ آبی و HAS_FATHER با رنگ زرد مشخص شده‌اند. پایین‌ترین گره، Frodo Baggins می‌باشد و ما شجره نامه او را (تها پدران و مادران) چاپ کرده‌ایم.

همچنین در تصویر زیر، شجره‌نامه‌ی کامل Frodo Baggins نمایش داده شده‌است که شامل همسران، برادر و خواهران و خویشاوندان دیگر می‌باشد. بدیهتا خروجی بسیار بزرگ است و گردی پایین، شخص مدنظر می‌باشد.



۴) مشکلات و توضیحات تکمیلی

نداشتین این موضوع که دقیقاً چه قسمت‌هایی باید توضیح داده شوند و یا عکس خروجی گذاشته شوند، باعث شد زمان بسیار زیادی صرف نوشتن گزارش بشود.

همچنین ایندا مقاله دیگری را انتخاب کرده بودم اما به دلیل مشکل و کندی در استفاده از API برای Name Entity Recognition مجبور به تغییر مقاله شدم. بجز این موارد، مشکل دیگری وجود نداشت.

گزارش کار کامل بود. مستندات Neo4j و Cypher نیز بسیار بسیار کامل و گویا بودند.

۵) آنچه آموختم

با پایگاهداده‌ی Neo4j و زبان مخصوص Cypher آشنا شدم. مقداری اطلاعات درباره Centrality و الگوریتم‌های گراف کسب کردم و دانسته‌ها را مرور کردم. انواع کوئری‌ها را از سطح بسیار ساده تا مقاله نوشته شده بررسی کردم و همچنین نحوه دریافت داده از Wikidata را آموختم.

همچنین با ابزار Bloom که Visualization و امکانات جستجوی جالبی فراهم می‌کند آشنا شدم. بنظرم، آزمایش بسیار ارزشمند و آموزنده‌ای بود.