Неструктурирани бази на податоци и XML

NoSQL – Graph бази на податоци

Yelp Data-set

Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Изработиле:

Петар Пипоски 171001

Климент Трајановски 171039

Јован Димовски 173221

Борис Велјановски 173008

Содржина

1. Вовед

Како цел на овој труд се поставува имплементација, обработка и анализа на Yelp податочното множество. За тоа успешно да го направиме во оваа точка треба да се запознаеме со термините и податоците кои што ќе ги користиме.

1.1 Граф

Граф престатува колекција од дискретни објекти кои што имаат колекција на релации кои што го поврзуваат тој објект со други. Токму со оваа структура на јазли (nodes), и ребра (edges) ќе се обидеме да ги претставиме сите податоци на начин што е погоден за читање и испрашување.

1.2 Граф во Neo4j

За изработка на овој проект, ние се одлучивме да работиме со Neo4J бидејќи таа е технологијата која што е најраспространета кога стаунва збор за Graph бази на податоци. За работа со оваа технологија се користи посебен прашален јазик наречен Cypher. Структурата на една база на податоци во Neo4j се состои од Nodes кои што треба, но и не мора да имаат Label, што претставува индикација од кој тип е соодветниот Node. Исто така секој Node може да има и атрибути кои што се од тип (key,value) кои што служат за дообјаснување и чување на дополнителни информаии поврзани за тој објект. Во одредени случаи тие информации се исто така објекти, па тие не се чуваат како атрибути, туку се поврзуваат со помош на Relationship, која што исто така треба да има соодветна Label вредност за да се назначи видот на врската, покрај тоа може да се специфицира и насоката при поврзувањето.

1.3 Yelp data-set

Yelp data-set-от претставува податочно множество од објекти од платформата за оценување на бизниси со исто име. Во овој податочен сет постојат информации за бизниси, корисници кои што пишуваат прегледи и

даваат оценки за истите, исто така се чуваат информации и за тие прегледи. На оваа платформа постои и социјален аспект, така што секој корисник може да има и пријатели и да ги следи нивните активности.

2. Методологија

Во оваа точка се објаснува процесот за изработување на базата, нејзино пополнување и испрашување.

2.1 Модел

Првиот чекор при изработувањето е анализа на податоците и креирање на модел кој што ни ги задоволува потребите. Податоците ги добиваме во json формат и тоа;

Business.json


```
"root":{
  "business_id": string"f9NumwFMBDn751xgFiRbNA"
  "name": string"The Range At Lake Norman"
  "address": string"10913 Bailey Rd"
  "city": string"Cornelius"
  "state": string"NC"
  "postal_code": string"28031"
  "latitude": float35.4627242
  "longitude": float-80.8526119
  "stars": float3.5
  "review_count": int36
  "is_open": int1
  "attributes":{
       "BusinessAcceptsCreditCards":string"True"
       "BikeParking": string"True"
       "GoodForKids": string"False"
       "BusinessParking": string"{'garage': False, 'street':False,'validated':
       False, 'lot': True, 'valet': False}"
       "ByAppointmentOnly": string"False"
       "RestaurantsPriceRange2": string"3"
       }
"categories": string"Active Life, Gun/Rifle Ranges, Guns & Ammo, Shopping"
}
```


User.json

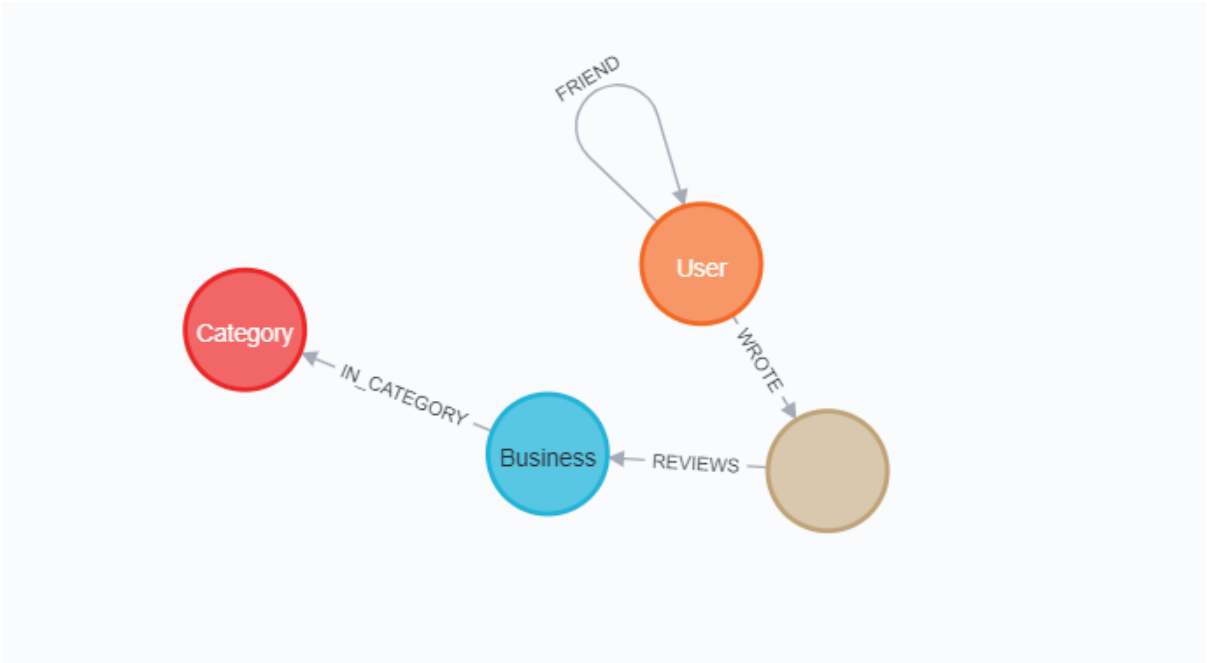

```
"root":{
       "user_id":string"ntlvfPzc8eglqvk92iDIAw"
       "name":string"Rafael"
```

```
        "review_count":int553
        "yelping_since":string"2007-07-06 03:27:11"
        "useful":int628
        "funny":int225
        "cool":int227
        "elite":string""
        "friends":string"oeMvJh94PiGQnx_6GlndPQ, wm1z1PaJKvHgSDRKfwhfDg,
        IkRib6Xs91PPW7pon7VVig, A8Aq8f0-XvLBcyMk2GJdJQ, eEZM1kogR7eL4GOBZyPvBA,
        e1o1LN7ez5ckCpQeAab4iw, _HrJVzFaRFUhPva8cwBjpQ, pZeGZGzX-ROT_D5lam5uNg,
        0S6EI51ej5J7dgYz3-O0lA, woDt8raW-AorxQM_tIE2eA, hWUnSE5gKXNe7bDc8uAG9A,
        c_3LDSO2RHwZ94_Q6j_O7w, -uv1wDiaplY6eXXS0VwQiA, QFjqxXn3acDC7hckFGUKMg,
        ErOqapICmHPTN8YobZIcfQ, mJLRvqLOKhqEdkgt9iEaCQ, VKX7jlScJSA-ja5hYRw12Q,
        ijIC9w5PRcj3dWVlanjZeg, CIZGlEw-Bp0rmkP8M6yQ9Q, OC6fT5WZ8EU7tEVJ3bzPBQ,
        UZSDGTDpycDzrlfUlyw2dQ, deL6e_z9xqZTIODKqnvRXQ, 5mG2ENw2PylIWElqHSMGqg,
        Uh5Kug2fvDd51RYmsNZkGg, 4dI4uoShugD9z84fYupelQ, EQpFHqGT9Tk6YSwORTtwpg,
        o4EGL2-ICGmRJzJ3GxB-vw, s8gK7sdVzJcYKcPv2dkZXw, vOYVZgb_GVe-kdtjQwSUHw,
        wBbjgHsrKr7BsPBrQwJf2w, p59u2EC_qcmCmLeX1jCi5Q, VSAZI1eHDrOPRWMK4Q2DIQ,
        efMfeI_dkhpeGykaRJqxfQ, x6qYcQ8_i0mMDzSLsFCbZg, K_zSmtNGw1fu-vmxyTVfCQ,
        5IM6YPQCK-NABkXmHhlRGQ, U_w8ZMD26vnkeeS1sD7s4Q, AbfS_oXF8H6HJb5jFqhrLw,
        hbcjX4_D4KIfonNnwrH-cg, UKf66_MPz0zHCP70mF6p1g, hK2gYbxZRTqcqlSiQQcrtQ,
        2Q45w_Twx_T9dXqlE16xtQ, BwRn8qcKSeA77HLaOTbfiQ, jouOn4VS_DtFPtMR2w8VDA,
        ESteyJabbfvqas6CEDs3pQ"
        "fans":int14
        "average_stars":float3.57
}
```


Review.json


```
"root":{
        "review_id":string"xQY8N_XvtGbearJ5X4QryQ"
        "user_id":string"OwjRMXRC0KyPrIlcjaXeFQ"
        "business_id":string"-MhfebM0QIsKt87iDN-FNw"
        "stars":int2
        "useful":int5
        "funny":int0
        "cool":int0
        "text":string"As someone who has worked with many museums, I was eager to
        visit this gallery on my most recent trip to Las Vegas. When I saw they
        would be showing infamous eggs of the House of Faberge from the Virginia
        Museum of Fine Arts (VMFA), I knew I had to go!

        Tucked away near the gelateria and the garden, the Gallery is pretty much
        hidden from view. It's what real estate agents would call "cozy" or
        "charming" - basically any euphemism for small.

        That being said, you can still see wonderful art at a gallery of any size,
        so why the two *s you ask? Let me tell you:

        * pricing for this, while relatively inexpensive for a Las Vegas attraction,
        is completely over the top. For the space and the amount of art you can fit
        in there, it is a bit much.
        * it's not kid friendly at all. Seriously, don't bring them.
        * the security is not trained properly for the show. When the curating and
        design teams collaborate for exhibitions, there is a definite flow. That
        means visitors should view the art in a certain sequence, whether it be by
        historical period or cultural significance (this is how audio guides are
        usually developed). When I arrived in the gallery I could not tell where to
        start, and security was certainly not helpful. I was told to "just look
        around" and "do whatever."

        At such a *fine* institution, I find the lack of knowledge and respect for
        the art appalling."
        "date":string"2015-04-15 05:21:16"
}
```

Отако ќе ги разгледаме податоците можеме да започнеме со градење на моделот. Прво треба да се одлучиме за јазлите односно објектите што ќе ги имаме во базата. Очигледни се дека треба да имаме јазел за User, Business и Review. Дополнително за да можеме полесно да сортираме и да бараме врски за категоријата за бизнисите ќе поставиме јазел и за Category. Сега ги имаме потребните јазли во базата, но треба да ги поврзиме и со врски. Бизнисите траба да припаѓаат во одредена категорија па за таа цел го поставуваме реброто IN_CATEGORY, како што приметуваме корисниците може да имаат пријатели, па поставуваме ребро FRIEND, секое Review го пишува одреден корисник, па додаваме ребро WROTE, потоа секое Review го опишува одреден бизнис, па го поставуваме реброто REVIEWS. Крајниот резултат изгледа вака:



Откако го имаме основниот модел, одлучуваме кои атрибути ќе ги поседува секој јазел.

- User:
  yelping_since – кога корисникот се регистрирал на Yelp.

average_stars – просекот на ѕвезди кој што ги доделил корисникот на бизнисите кои што ги оценил.
user_id – уникатен индентификатор на корисникот.
cool – бројот на “cool” рекации што ги добил на своите прегледи.
name – името на коррисникот.
review count – бројот на прегледи што ги испишал корисникот.
useful – бројот на “useful” рекации што ги добил на своите прегледи.
funny – бројот на “funny” рекации што ги добил на своите прегледи.
fans – бројот на корисници кои што ги следат неговите прегледи.

- Business:
  address – адресата на бизнисот.
  city – градот во кој што се наоѓа бизнисот.
  is_open – дали бизнисот работи.
  latitude – географска висина – се користи за приказ на мапа.
  longitude – географска должина – се користи за приказ на мапа.
  business_id – уникатен идентификатор на бизнисот .
  name – име на бизнисот.
  review_count – број на оценки кој што ги добил бизнисот.
  stars – просечен број на ѕвезди кој што ги добил бизнисот.
  state – држава во која што се наоѓа бизнисот.
  postal_code – поштенски број на државата во која што се наоѓа бизнисот.
  BusinessAcceptsCreditCards – дали бизнисот прифаќа кредитни картички.
  BikeParking – дали бизнисот има паркинг за велосипеди.
  GoodForKinds – дали бизнисот е наменет и за деца.
  BusinessParking – Видот на праркинг што го нуди бизнисот.
  ByAppointmentOnly – Дали бизнисот функционира само со најава.

- Category:
  name – Име на категоријата, исто така се користи и како уникатен идентификатор.

- Review:
  date – датум на креирање на прегледот.
  review_id – уникатен идентификатор за прегледот.

cool – бројот на “cool” рекации што ги добил прегледот
text – текстуалната содржина на прегледот.
stars – бројот на ѕвезди што ги доделува прегледот.
useful – бројот на “useful” рекации што ги добил прегледот
funny – бројот на “funny” рекации што ги добил прегледот

2.2 Креирање и пополнување на базата на податоци.

Сега, откако имаме одреден модел можеме да започнеме со градење на нашата база. Прво треба да отвориме нов проект во Neo4j и да ја конфигурираме нашата база, односно да го инсталираме apoc plugin-от, кој што се користи за импортирање на податоци, и да поставиме apoc.import.file.enabled=true, бидејќи податоците ќе ги читаме од фајлови кои што се локално на нашата машина. Следен чекор е да ги симнеме податоците, ние го направивме тоа од https://www.kaggle.com/yelp-dataset/yelp-dataset, и тие податоци да ги поставиме во Import фолдерот од нашиот проект. Сега можеме да започнеме со читање на податоци, тоа го правиме со помош на Cypher, односно со функцијата apoc.load.json за читање на json датотеките вгнесдена во apoc.periodic.iterate за читањето на податоци да можеме да го извршуваме итеративно, односно паралелно на повеќе јадра, бидејќи станува збор за датотеки кои што се неколку Gb.

- Cypher код за читање на User.

```
CALL apoc.periodic.iterate('CALL apoc.load.json("file:/yelp_academic_dataset_user.json") YIELD
value AS user', '
CREATE (u:User {user_id: user.user_id})
SET u.name  = user.name,
    u.review_count = user.review_count,
    u.average_stars=user.average_stars,
    u.fans=user.fans,
    u.yelping_since=user.yelping_since,
    u.useful=user.useful,
    u.funny=user.funny,
    u.cool=user.cool
    ', {batchSize:10000,iterateList:true,parallel:true,params:{},concurrency:1000,retries:3})
```

- Cypher код за читање на Business, Поставување на Category и поставување на релацијата IN_CATEGORY

```
CALL apoc.periodic.iterate('CALL apoc.load.json("file:/yelp_academic_dataset_business.json")
YIELD value AS business', '
UNWIND business.attributes as atr
MERGE (b:Business {business_id: business.business_id})
SET b.name=business.name,
b.address=business.address,
b.city=business.city,
b.state=business.state,
b.postal_code=business.postal_code,
b.latitude=business.latitude,
b.longitude=business.longitude,
b.stars=business.stars,
b.review_count=business.review_count,
b.is_open=business.is_open,
b.BusinessAcceptsCreditCards=atr.BusinessAcceptsCreditCards,
b.BikeParking=business.BikeParking,
b.GoodForKids=business.GoodForKids,
b.BusinessParking=business.BusinessParking,
b.ByAppointmentOnly=business.ByAppointmentOnly,
b.RestaurantsPriceRange2=business.RestaurantsPriceRange2
WITH b, split(business.categories, ", ") as categories
UNWIND categories as cat
MERGE (c:Category {name : cat})
MERGE (b)-[:IN_CATEGORY]→(c)
    ', {batchSize:10000,iterateList:true,parallel:true,params:{},concurrency:1000,retries:0})
```

- Cypher код за поставување на релацијата FRIEND.

```
CALL apoc.periodic.iterate('CALL
apoc.load.json("file:/yelp_academic_dataset_user.json") YIELD value
AS user', '
MATCH (u:User {user_id: user.user_id})
WITH u, split(user.friends, ", ") as friends
UNWIND friends as fnd
MATCH (s:User {user_id : fnd})
MERGE (u)←[:FRIEND]→(s)
    ', {batchSize:10000,iterateList:true,parallel:true,params:
{},concurrency:1000,retries:3})
```

- Cypher код за читање на Review и поставување на релациите WROTE и REVIEWS.

```
CALL apoc.periodic.iterate('CALL apoc.load.json("file:/yelp_academic_dataset_review.json") YIELD value AS review', '
MERGE (r:Review {review_id: review.review_id})
ON CREATE SET r.text=review.text,
r.stars=review.stars,
r.useful=review.useful,
r.funny=review.funny,
r.cool=review.cool,
r.date=review.date
WITH r,review
MATCH (b:Business {business_id: review.business_id})
MATCH (u:User {user_id: review.user_id})
MERGE (u)-[:WROTE]→(r)
MERGE (r)-[:REVIEWS]→(b)
    ', {batchSize:10000,iterateList:true,parallel:true,params:{},concurrency:1000,retries:3})
```

-Додавање на Constraints за индетификаторите на јазлите, а со тоа се креираат и индекси за побрзо пребарување.

```
CREATE CONSTRAINT ON (u:User) ASSERT u.user_id IS UNIQUE;
CREATE CONSTRAINT ON (b:Business) ASSERT b.business_id IS UNIQUE;
CREATE CONSTRAINT ON (c:Category) ASSERT c.name IS UNIQUE;
CREATE CONSTRAINT ON (r:Review) ASSERT r.review_id IS UNIQUE;
```

Сега, откако базата е направена и пополната, можеме да започниме со пишување на прашалници и применување на логика врз податоците.

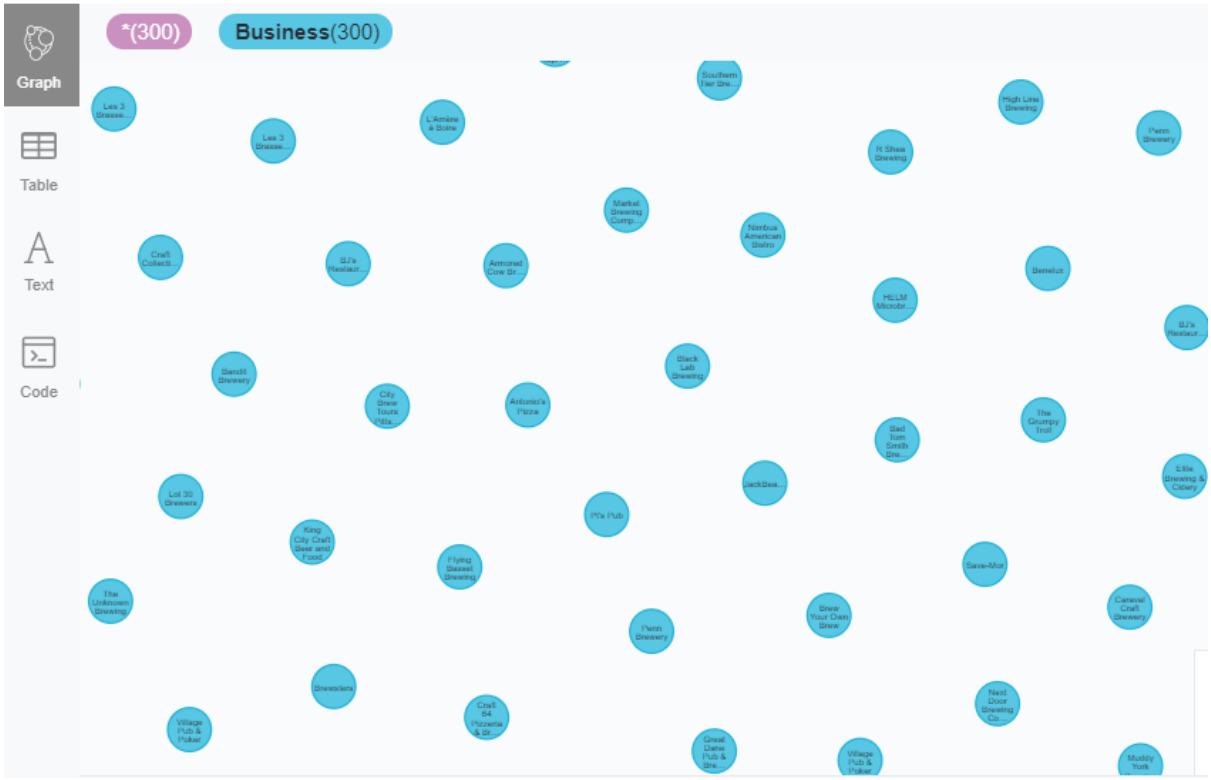
2.3 Прашалници

Во оваа точка ќе почнеме од едноставно филтрирање, за добивање на податоци со одредени атрибути, па понатаму ќе го надоградуваме тој прашалник за да добиеме конкретни податоци што не се очигледни, при што ќе ги опфатиме сите релации и видови на податоци во базата.

- Филтриање по категорија, ги бараме сите бизниси што припаѓаат во категоријата “Breweries”.

```
MATCH (c:Category)←[:IN_CATEGORY]-(b:Business)
WHERE c.name CONTAINS "Breweries"
return b
```

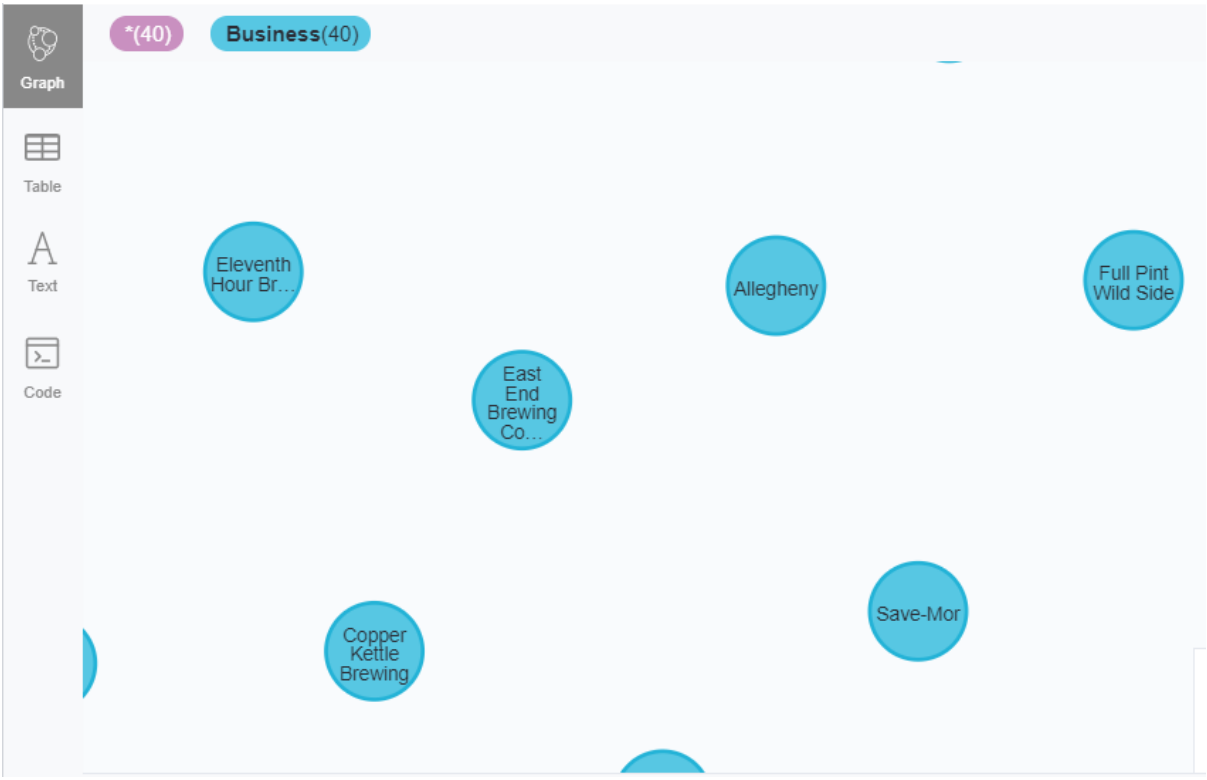
- Како одговор добиваме 300 бизниси. completed after 187 ms.



- Додаваме и во кој град да се наоѓаат пивниците.

```
MATCH (c:Category)←[:IN_CATEGORY]-(b:Business)
WHERE c.name CONTAINS "Breweries" and b.city =
"Pittsburgh"
return b
```

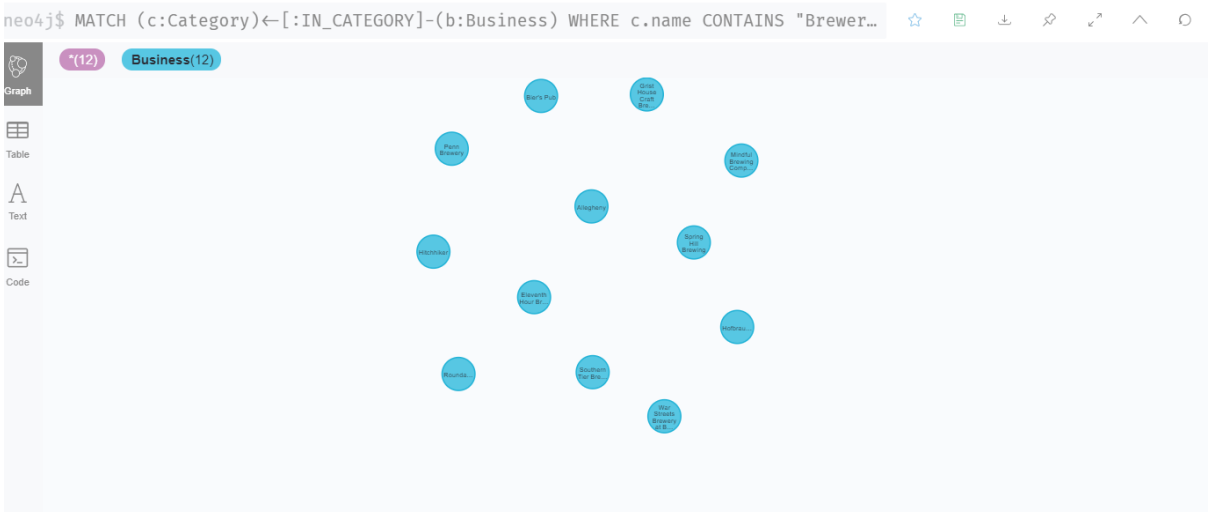
- Со ова бројот на бизниси кои што одговараат на овој прашалник се намалува на 40. completed after 8 ms.



- Дополнително, ќе претпоставиме дека ние сме корисник со user_id=”4m9NXICYBC5i9t4aTt-I6w”, оваа информација ќе ја искористиме за да ги добиеме пивниците кои што ние сме ги оцениле.

```
MATCH (c:Category)←[:IN_CATEGORY]-(b:Business)
WHERE c.name CONTAINS "Breweries" and b.city="Pittsburgh"
MATCH (b)←[:REVIEWS]-(r:Review)←[:WROTE]-(me:User {user_id :"4m9NXICYBC5i9t4aTt-I6w"})
return b
```

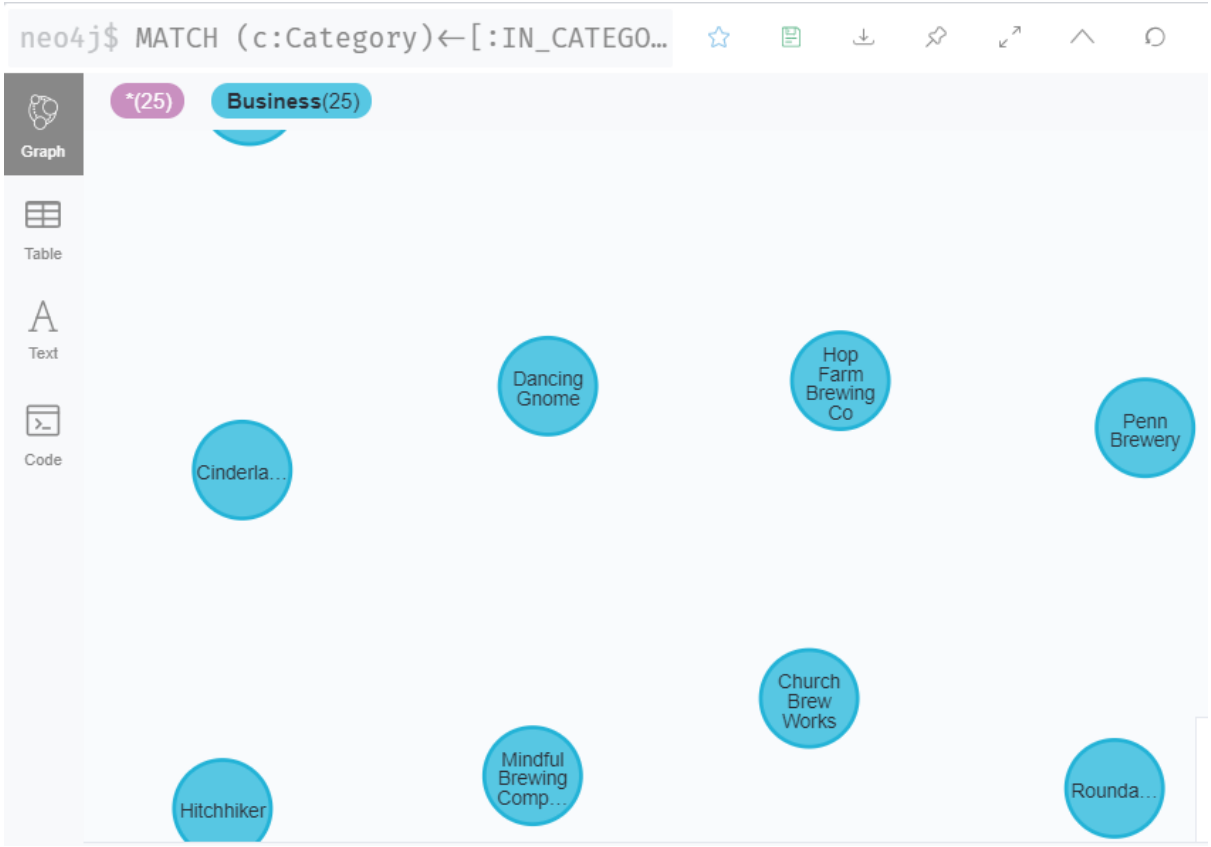
- Како одговор добиваме дванаесет пивници. completed after 9 ms.



- Следно, ќе се обидеме да ги добиеме пивниците, кои што им се допаднале на нашите пријатели, односно пивниците што нашите пријатели ги оцениле со 4 / 5 или повисоко.

```
1 MATCH (c:Category)←[:IN_CATEGORY]-(b:Business)
2 WHERE c.name CONTAINS "Breweries" and
  b.city="Pittsburgh"
3 MATCH (b)←[:REVIEWS]-(r:Review)←[:WROTE]-
  (friend:User)-[:FRIEND]-(me:User {user_id
  :"4m9NXICYBC5i9t4aTt-I6w"})
4 where r.stars ⩾ 4
5 return b
```

- Како резултат добиваме 25 пивници. completed after 124 ms.



- Следно ќе се обидеме да ги прикажеме името, адресата и просекот на ѕвездите кои што се доделени на тие бизниси, бројот на прегледи напишани за тие бизниси, подредени по просечниот број на ѕвезди.

```
MATCH (c:Category)←[:IN_CATEGORY]-(b:Business)
MATCH (b)←[:REVIEWS]-(r:Review)←[:WROTE]-
(friend:User)-[:FRIEND]-(me:User {user_id
:"4m9NXICYBC5i9t4aTt-I6w"})
MATCH (b)←[:REVIEWS]-(r2:Review)
WHERE c.name CONTAINS "Breweries" and
b.city="Pittsburgh" and r.stars ⩾4
RETURN b.name, b.address, avg(r2.stars) as stars,
count(r2) as noReviews ORDER BY stars DESC
```

completed after 232 ms.

- Со овој прашалник ја достигнуваме најголемата комплексност на прашањата што може да се достиге кога сакаме да ја испрашуваме базата на податоци, опфатени се сите релации, и сите објекти при барање на одговорот.

2.4 Graph Algorithms

Бидејќи структурата на нашата база на податоци е граф, тоа ни овозможува да извршуваме разни алгоритми и да применуваме одредена логика што наликува на логичко расудување, барање на најкраток пат итн.

2.4.1 Одредување на сличност на јазли

Еден од алгоритмите кои што ги нуди gds библиотеката е gds.nodeSimilarity , кој што ја одредува сличноста да два јазли во даден подграф врз основа на релациите кои што ги поседуваат. Тоа ќе го направиме за бизнисите. Прво треба да се креира графот.

```
CALL gds.graph.create(
    'bicGraph',
    ['Business', 'Category'],
    'IN_CATEGORY'
);
```

Со оваа команда креираме граф составен Business, Category јазлите и IN_CATEGORY релацијата што е спремен за да се обработува од gds алгоритмите.

```
CALL gds.nodeSimilarity.stream('bicGraph')
YIELD node1, node2, similarity
RETURN gds.util.asNode(node1).name AS Business1,
gds.util.asNode(node2).name AS Business2, similarity
ORDER BY similarity DESCENDING, Business1, Business2
```

Како резултат го добиваме коефициентот на сличност 0.0 – 1.0, и името на двата бизниси кои што се споредуваат.

2.4.2 Одредување на кластери

Друг алгоритам кој што го нуди библиотеката е gds.labelPropagation кој што детектира кластери во одреден подграф, а како резултат го враќа името на јазелот и ID на кластерот. Прво треба да го направиме графот.

```
CALL gds.graph.create(
    'friendGraph',
    'User',
    'FRIEND',
)
```

Со оваа команда креираме граф составен само со User јазелот и Friend релацијата што е спремен за да се обработува од gds алгоритмите.

```
CALL gds.labelPropagation.stream('friendGraph')
YIELD nodeId, communityId AS Community
RETURN gds.util.asNode(nodeId).name AS Name, Community
ORDER BY Community, Name
```

Вака изгледа синтаксата за повикување на лгоритмот, а како резултат добиваме табела со колони; ид на кластер и име на корисникот.

Иако има и многу други алгоритми сметаме дека се доволни овие два, за да се нагласи нивното постоење и нивната употреба.

3. Добиени резултати

Во оваа точка ќе дадеме примери за употреба на прашалниците и добиените резултати во контекст на апликацијата на која што оваа база може да биде основа.

- Првите прашалници каде што се филтрира според типот на бизнисот и локацијата на бизнисот може да се искористат при имплементација на некакво пребарување во апликацијата, пример корисникот сака да ги пронајде бизнисите кои што припаѓаат во одредена категорија, тоа би се остварило со текстуален инпут, исто така можеме да популираме drop-down мени со сите градови на бизнисите, па корисникот да го одбере и тоа.
- Доколку корисникот е најавен, со следниот прашалник можеме да чуваме евиденција на неговите прегледи, во еден таб да може да ги гледа сите бизниси кои што ги оценил.
- Дополнително слецниот прашалник можеме да го искористиме за давање препораки на корисникот за бизниси кои што би го интересирале.
- Последниот прашалник можеме исто така да го искористиме за давање на препораки на корисникот кога тој сака да се води по бизниси кои што имаат високи оценки со голем број на луѓе кои што го потврдиле тоа.

4. Заклулчок

За крај ќе направиме мала паралела со SQL бази на податоци и ќе ги нагласиме подобрувањата кои што ги прават Graph базите, како и нивната најчеста примена.

- Дополнителни информации,односно Graph базите, постои механизам таканаречен Label што му дава контекст на јазелот, покрај тоа релациите имаат насока.
- Graph базите имаат многу подобра скалабилност, бидејќи индекси за пребарување се користат само за јазлите од каде што почнуваме да го изминуваме графот. Односно за SQL база на податоци, да поврзе еден ентитет со други со операцијата JOIN потребно е повеќе пати да ја пребарува табелата на индекси, а кога записите се во

10тици милиони, како во нашиот случај доа се одразува на времето за извршување на одреден прашалник. Во Graph-базите на податоци доколку почнеме да изминуваме од одредена точка, тоа време на изминување ќе останува исто без разлика колку други записи додаваме во базата.

- Во Graph базите полесно се наоѓаат индиректни врски помеѓу два јазли, односно во SQL базите на податоци доколку два ентитети се оддалечени со N релации, тогаш за да се добие таа врска потребно е да сенаправат N – JOIN операции, додека кај Graph базите само треба да се изминат ребрата.
- Полесен начин на одговарање на прашања кои што не се поврзани, односно листање на сите релации на одреден јазел е скоро инстанто.
- Подобро искористување на меморија, односно кај SQL базите на податоци може да постојат празни вредности за одредени атрибути, односно во редицата да има null елемент, додека кај Graph – базите на податоци тоа се избегнува со тоа што доколку вредноста е null, таа не се запишува ни како релација ни како атрибут.
- Агилност, во Graph базите на податоци може да се додаваат итеративно елементи и атрибути на јазлите без да се наштети врз постоечките податоци.
- Graph базите на податоци бидејќи моделот е граф дозболуваат примена на логика за расудување.

4.1 Top Use-Cases за Graph бази наподатоци

-Real time recommendations - правење на препораки врз база на податоците во реално време.

-Master data Management - чување на податоци за хиерархијата во некоја институција.

-Fraud Detection - откривање на прекршоци, преклопување на информации што треба да се единствени.

-Network and IT-operations – мапирање на поврзувања во мрежа.

Доколку имате потреба од градење база на податоци, а потреба на таа база е некоја од наведените карактеристики или таа база служи за некоја од овие функционалности тогаш Graph база на податоци е вистинскиот избор.