VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS

Techninis aprašymas
„GREMLIN“ UŽKLAUSŲ SISTEMA

Vasarė Pratuzaitė
Informatikos *2 kursas, 1 grupė, 2 pogrupis*

Vilnius, 2024

Turinys

Įvadas

Duomenų bazės yra struktūrizuotas informacijos rinkinys, kurį galima valdyti. Anksčiau duomenys buvo saugomi įvairiose bibliotekose ar specialiuose žurnaluose, užimdami vietą ir padarydami juos sunkiai prieinamus. Kompiuterių atsiradimas ankstyvaisiais 1960-aisiais pažymėjo kompiuterizuotų duomenų bazių pradžią. Duomenų bazių atsiradimas pagreitinęs žmonių gyvenimus padėjo žmonėms saugoti vis didėjančios informacijos kiekius. Įdomu tai, kad duomenų bazės atsirado prieš pirmųjų kompiuterių atsiradimą. Taip pat, jos būna įvairių tipų, pavyzdžiui, grafinio pavidalo, kai duomenys tokiose duomenų bazėse dažniausiai suprantame kaip viršūnių ir briaunų ryšį, kitaip tariant, tokį reiškinį vadiname grafu. Grafinio pavidalo duomenų bazėse viršūnės nusako kažkokius duomenis, o briaunos, jungiančios viršūnes, pasako mums sąsajas tarp viršūnių. Vienas garsus matematikas Leonhardas Euleris XIXa. pradėjęs tyrinėti Königsbergo tiltų problemą, atrado vieną iš pirmųjų grafo teorijos pavyzdžių, kuris parodė, kaip grafinė struktūra, kuri gali būti naudojama modeliuoti realius ryšius. Grafo struktūra pradėta naudoti duomenų bazių kontekse dar 1960-aisiais sukūrė pradžią šiai naujai besivystančiai sričiai. Jos dėka buvo kuriamos navigacinės sistemos, tokios kaip GPS, ir buvo puikiai priderinamos kelių maršrutų paieškai atlikti. Kartais mes galime grafą interpretuoti kitaip - viršūnes galima matyti kaip lenteles, o viršūnių savybes kaip lentelės stulpelius arba, kitaip tariant, atribudus, pavyzdžiui, sukurdami viršūnę, kurią pavadinsime konkrečiu žmogumi ir atitinkamai sukursime miesto ir operacinės sistemos viršūnes tam, kad galėtume surasti ryšį, kai duomenų padaugėja. Toks grafinis principas mums neleis pasimesti, ir taps, lyg mūsų žemėlapis, kuris bus daug aiškesnis nei, kad surašyti duomenys. Šiame sparčiai besivykstančiame technologiniame pasaulyje, nuolatos susiduriame su didėjančios informacijos kiekiu, kurį tenka apdoroti, tačiau šis grafinis duomenų bazės metodas leidžia nesunkiai modifikuoti ar pridėti naujus duomenis. Taip pat, teikia pridėtinę vertę, dėl savo darbo našumo, efektyvius būdus kaip keisti ryšius, turi veiksmingus algoritmus, kurie padeda ieškoti informacijos tam tikrose duomenų bazėse, ir labai dažnai yra naudoja įvairaus tipo aplikacijose, kurių vienas iš tikslų - greitai atnaujinti tam tikrą informaciją konkrečiu metu. Todėl šis būdas be galo efektyvus ir tinkamas naudoti įvairiose srityje. Įdomu tai, kad tokios duomenų bazės turi labai daug pranašumo, nes jas galima pritaikyti įvairiose užklausų kalbose, kurios linkusios atlikinėti darbus susijusiais su grafais. Tokių kalbų yra gana nemažai, pavyzdžiui, Cypher, GQL, SPARQL ar Gremlin. Visos šios kalbos ypatingos ir unikalios, nes gali daryti skirtingus dalykus, tad turi savų privalumų. Viena iš šių kalbų skirta dirbti su

standartizuota grafų užklausų kalba, kita ypatingesnė tuo, kad gali apdoroti duomenis iš „Neo4j“ duomenų bazės, kuri yra populiauriausia ir galingiausia grafo duomenų bazių sistemų. „Gremlin“ yra nepriklausoma nuo konkrečių grafinių duomenų bazių, bet yra integruota į „Apache TinkerPop“ platformą. Tai reiškia, kad „Gremlin“ užklausos gali būti vykdomos ne tik grafio duomenų bazėse, bet ir kitose duomenų bazėse, kurios remia „TinkerPop“ – suteikiant galimybę taikyti tas pačias užklausas skirtinguose grafinių duomenų kontekstuose. Algoritmiskai judančios mašinos per „Gremlin“ pasaulį, patvirtino jo logiką. „Gremlin“ stengėsi padaryti jas efektyvesnes, išraiškingesnes, kad būtų patogios ir reikalingos naudoti. „Greičiau, greičiau, dabar link pasaulio galo, kur visam laikui būtų dabar, spindinčiai apgaubiantis tai, kas yra - „The TinkerPop“.“

1pav.

Grafo kūrimo taisyklės, sintaksė

„Gremlin“ kalboje kartu su „Apache TinkerPop“, grafo kūrimo taisyklės yra glaudžiai susijusios su viršūnių ir briaunų sukūrimu bei jo tarpusavio ryšiu. Jo struktūra yra topologiškai kuriama ir reikalauja laikytis griežtų nurodymų tam, kad būtų galima atlikti įvairias užduotis susijusias su grafais. Verta prisiminti, kad rašydamas Gremlin užklausų kalba, duomenis į grafą galima įkelti tik vienu būdu. Norint sukurti viršūnę, reikia pasinaudoti „g.addV“ funkcija su nurodytu grafo šaltiniu. „g“ yra grafo keliautojo šaltinis (GraphTraversalSource), o „addV(label) nurodome, kad pridedame naują viršūnę su konkretaus tipo etikete (label). Taip pat, jeigu yra reikalingas naujos viršūnės savybės priskyrimas arba, kitaip tariant, naujo mazgo savybės papildymui, „property“ metodas tai ir leis padaryti, tik reikėtų prirašyti kelis duomenis – pateikti raktą ir reikšmę. Raktas žymėtų abstrakčią informaciją, o reikšmė konkretų duomenį. Pavyzdžiui, pagal taisyklingą sintaksišką užrašymą g.addV(label).property(raktas, reikšmė) galime sukurti grafo viršūnę, kuri atitiktų naują žmogų mūsų grafe. Tarkime g.addV(label).property(‘vardas’, ‘Benas’) – sukuriame viršūne „Benas”. Tokiu būtų mes galime sukurti ne vieną viršūnę, tačiau tam, kad grafas būtų logiškas, reikalingas viršūnių apjungimas, kuris su kitos funkcijos pagalbą, tai ir padaro. Norėdami sukurti briauną tarp dviejų viršūnių galime pasinaudoti dar viena dažnai naudojama funkcija „addE” jis užrašo taip: addE(label).from(šaltinis).to(tikslas). Pavyzdžiui, Benas domisi futbolu galėtume užrašyti: addE(‘domisi’).from(‘Benas’).to(‘Futbolas’). Sukurdami viršūnes ir briaunas reikia dar kelių funkcijų, kurios galėtų grąžinti viršūnę ar briauną, todėl viršūnių radimui naudosime funkciją V(), o briaunoms rasti – E(). Nors pačios funkcijos geba atlikti darbą be patikslinimų, tam, kad kodas būtų suprantamas, galime nurodyti etiketę, kuri užsirašytu su šia funkcija būtent taip: g.V().hasLabel(x, y). Pavyzdžiui tam, kad galėtume rasti visus žmones, kurie domisi futbolu, mums užtektų parašyti šią eilutę su atitinkamais duomenimis: g.V().hasLabel(“žmogus”, “futbolas”).Tokiu būtų bus randamos visos reikiamos viršūnės. Analogiškai mes galime rasti reikiamas briaunas nurodydami funkciją E().

Klausimų sudarymo sintaksė

„Apache TinkerPop“ vienas iš pagalbininkų, kuris leidžia naudoti Gremlin užklausų kalbą, todėl jos dėka, mes galime rašyti klausimus susijusius su sąvokų grafu ar žinių baze. Tam, kad sukurti užklausą, reikia vadovautis tam tikrais žingsniais. Užklausose duomenis galima filtruoti, rikiuoti, transformuoti, pavyzdžiui sujungiant duomenis. Tam, kad būtų galima vykdyti įvairias užklausas, yra svarbu žinoti klausimų sudarymo sintaksę. Tačiau pirmiausia vertėtų įsiminti, kad užklausa turėtų prasidėti nuo konkrečios viršūnės ar viršūnių aibės. Toliau sekant svarbiausius žingsnius, svarbu atkreipti dėmesį į grafo šaltinį bei kaip jis yra pavadintas. Tuomet panaudojus funkciją grafo viršūniams pasiekti galime atlikti kitas, įvairias operacijas, kaip judėjimą iš esamos viršūnes į išeinačias – ‚out()‘, ar judėti nuo esamos viršūnės į įeinančias – ‚in()‘, ar nukeliauti iš esamos viršūnės į įeinačias ar išeinančias viršūnes – ‚both()‘. Šios trys operacijos yra vienos iš esminių „Gremlin“ keliautojo judėjimo strategijų, kurios leidžia analizuoti grafus. Norėdami filtruoti briaunas ar viršūnes pagal jų savybes ar kažkokius kriterijus, naudojami filtravimo metodai, tokie kaip: has(), hasLabel(), where(), filter(), is(), not(), and(), or(), hasId() ir kt.

has()	Pagal nurodytas savybes, filtruoja duomenis
hasLabel()	Pagal nurodytą etiketę, filtruoja mazgus
where()	Papildomoms sąlygoms rašyti ar filtrams keliautojui
filter()	Papildomoms sąlygoms rašyti, veikia panašiai kaip where()
is()	Naudojamas tikrinimui
not()	Neigiamui fitravimui atlikti
and()	Leidžia atlikti jungimą „ir“ sąlygų grandinėje
or()	Leidžia atlikti jungimą „arba“ sąlygų grandinėje
hasld()	Filtruoja viršūnes gal jų identifikatorių

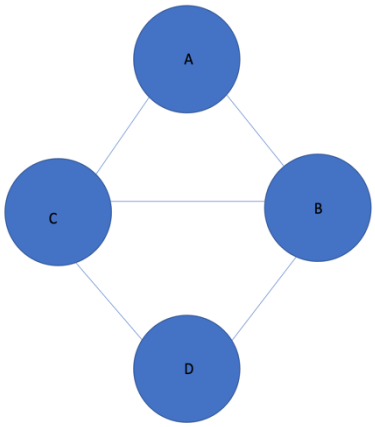
1lentelė. Reikšmių paaiškinimas

Ir galiausiai agregavimas ir transformavimo metodas yra pagrindiniai elementai, kurie sudaro šios unikalios užklausų sistemos sintaksę. Čia įeina skaičiavimai su duomenimis, pavyzdžiui count(), group(), valueMap(), sum(), max(), fold(), unfold() funkcijos, kurios yra skirtos

duomenų apibendrinimui ir paskutiniam žingsniui pasiekti atsakymus į sąvokų grafe iškeliamus klausimus.

Algoritmas, kurio pradiniai duomenys yra sąvokų grafas (arba žinių bazė) ir klausimas, o išvesties duomenys – atsakymas į klausimą.

"Gremlin" grafo perėjimų algoritmas iš esmės veikia taip: užduotis pradedama nuo viršūnių aibės ar konkretaus mazgo, nustatančio pradinį tašką perėjimams. Tada "Gremlin" leidžia vykdyti įvairius perėjimo žingsnius per grafą, kurio kiekvienas žingsnis siaurindamas arba transformuodamas esamą viršūnių aibę ar viršūnę. Svarbu tinkama perėjimo tvarka, kurią nusako operacijos „out“, „in“, ir kt. "Gremlin" užklausa baigiasi terminuojamuoju žingsniu, kuris grąžina norimą rezultatą, tokį kaip viršūnių sąrašą, konkrečią viršūnę arba tam tikrų reikšmių sumą. Išsiaiškindami, kad yra algoritmų kurie gali būti įgyvendinami naujant “Gremlin” užklausas, vertėtų turėti omenyje, kad tik tam tikra veiksmų grupė gali būti vykdoma, nes ši kalba gali spręsti grafo duomenų keliavimo problemas. Vienas iš tokių algoritmų galėtų būti trumpiausio kelio radimo, kurio pagrindinė esmė yra rasti trumpiausią kelią tarp dviejų viršūnių grafe. Pagal šį pavyzdį bandome rasti trumpiausią kelią tarp A ir D viršūnių ir randame, kad perėję tris viršūnes rasime trumpiausią grafą.




```
g.addV('A').as('A').
  addV('B').as('B').
  addV('C').as('C').
  addV('D').as('D').
  addE('link').from('A').to('B').
  addE('link').from('A').to('C').
  addE('link').from('B').to('D').
  addE('link').from('C').to('D').iterate()

g.V().hasLabel('A').as('start').
  V().hasLabel('D').as('end').
  repeat(both().simplePath()).until(hasId('end')).path().limit(1)
```

2pav. Grafo vizualizacija su programos kodu

Sąvokų grafo (arba žinių bazės) pavyzdys

Sąvokų Grafas:

viršūnės:

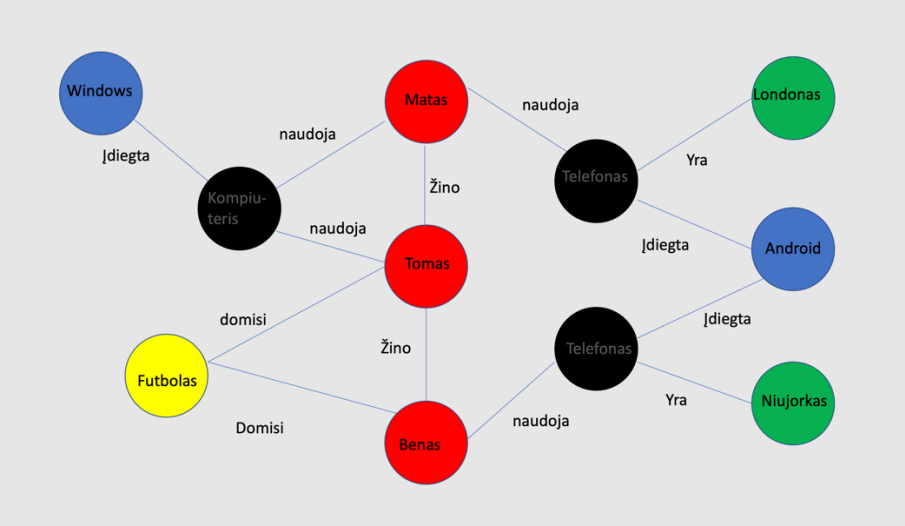
- Žmonės
- Miestas
- Pomėgis
- Įrenginys
- Operacinė sistema

Briaunos:

- Yra
- Naudoja
- Domisi
- įdiegta

Reikšmė:

- Žmogus - raudona spalva
- Miestas – žalia spalva
- Pomėgis – geltona spalva
- Įrenginys – juoda spalva
- Operacinė sistema – mėlyna spalva



3pav. Grafas

Trys pavyzdiniai klausimai, į kuriuos bus ieškoma atsakymo sąvokų grafe (arba žinių bazėje)

Trys pavyzdiniai klausimai:

1. Kas gyvena mieste „Niujorkas“?
2. Kuris asmuo naudojasi įrenginiu, kuris turėtų „Windows“ operacinę sistemą?
3. Kur yra daugiausia žmonių, kurie naudojasi ta pačia operacine sistema?

Trys algoritmo vykdymo scenarijai, kai pradiniai duomenys – pavyzdinis sąvokų grafas (arba žinių bazė) ir trys skirtingi klausimai, o išvesties duomenys – atsakymai į klausimus.

Siekiant atsakyti į pavyzdinius klausimus, reikia išanalizuoti sąvokų grafą. Po to reikia išnagrinėti pateiktus klausimus, pagal kuriuos svarstysime apie algoritmo vykdymo scenarijus.

Grafo paieška:

Šiame scenarijuje naudojama grafo paieškos algoritmas, siekiant rasti atsakymus į visus tris klausimus.

1 klausimas:

- Pradedame nuo viršūnės, atitinkančios miestą „Niujorkas“.
- Naudojame grafo paieškos algoritmą, kad rastume visus su ja susijusius asmenis.
- Asmenys, turintys ryšį su "yra" etikete ir "Niujorkas" reikšme, yra atsakymas į šį klausimą.

2 klausimas:

- Pradedame nuo viršūnių, atitinkančių asmenis.
- Naudojame grafo paieškos algoritmą, kad rastume visus su jais susijusius įrenginius.
- Įrenginiai, turintys ryšį su "naudoja" etikete ir "Windows" reikšme, yra atsakymas į šį klausimą.

3 klausimas:

- Suskaičiuojame kiekvienos operacinės sistemos naudotojų skaičių.
- Operacinė sistema, turinti daugiausia naudotojų, yra atsakymas į šį klausimą.

Gremlin užklausos:

Pirmoji:

užklausa pradeda nuo viršūnių su etikete 'Žmogus'. Paskui naudoja where() filtrą, kad rastų viršūnes, susietas su išeinančiu ryšiu 'yra'. Toliau filtruoja pagal miesto reikšmę 'Niujorkas'. Ir galiausiai naudoja valueMap() funkciją, kad grąžintų kiekvieno asmens savybes, kaip žodyną.

```
g.V().hasLabel('Žmogus').where(outE('yra').has('miestas', 'Niujorkas')).valueMap()
```

Antroji:

pradedama nuo viršūnių su etikete 'Žmogus'. Po to seka išeinančiu ryšiu 'naudoja'. Vėliau naudoja where() filtrą, kad įsitikintų, jog pasiektas mazgas turi 'Įrenginys' etiketę. Toliau filtruoja pagal operacinės sistemos reikšmę 'Windows'. O paskutiniame žingsnyje, naudoja valueMap() funkciją, kad grąžintų kiekvieno įrenginio savybes kaip žodyną.

```
g.V().hasLabel('Žmogus').outE('naudoja').where(inV().hasLabel('Įrenginys')).filter(has('opera
cinė sistema', 'Windows')).valueMap()
```

trečioji:

prasideda nuo viršūnių su etikete 'Žmogus' radimo. Po to seka išeinančiu ryšiu 'naudoja'. Paskui naudoja groupCount() funkciją, kad suskaičiuotų kiekvienos operacinės sistemos naudojimo atvejus. Vėliau demostruoja grupavimą pagal 'operacinė sistema' savybę. Ir rikiuoja rezultatus mažėjimo tvarka naudojant orderBy(desc).

```
g.V().hasLabel('Žmogus').outE('naudoja').groupCount().by('operacinė
sistema').orderBy(desc)
```

Išvados

1. naudoja grafo paieškos algoritmą, kuris yra bendra technika, skirta rasti kelius grafo viduje. Šiame kontekste algoritmas naudojamas norint rasti asmenis, gyvenančius mieste „Niujorkas“, ir įrenginius, kuriuos naudoja asmenys, naudojantys „Windows“ operacinę sistemą.
2. programuojamos užklausos, norint rasti asmenis, gyvenančius mieste „Niujorkas“, ir įrenginius, kuriuos naudoja asmenys, naudojantys „Windows“ operacinę sistemą.
3. naudoja matematinę logiką ir apdorojimą, siekiant išgauti informaciją iš grafo. Šiame kontekste matematinė logika naudojama norint suskaičiuoti kiekvienos operacinės sistemos naudotojų skaičių ir nustatyti operacinę sistemą, turinčią daugiausia naudotojų.

Apibendrinimas

- Visi trys scenarijai teisingai atsako į užduotus klausimus.
- Kiekvienas scenarijus naudoja skirtingą metodą grafui išnagrinėti.
- Geriausias metodas priklausys kokios bus konkrečios užduoties ir duomenų struktūros.

Taip pat, Gremlin yra ir sistema, ir algoritmas, nes;

- Kaip sistema: Gremlin yra grafo duomenų bazių kalba, skirta duomenų paieškai atlikti, apdorojimui grafo struktūrose. Ji apima platų funkcijų spektrą, skirtą darbui su grafais, įskaitant grafo paiešką, ciklų aptikimą ir duomenų vizualizavimą.
- Kaip algoritmas: Gremlin taip pat gali būti apibūdinamas kaip algoritmų rinkinys, skirtas grafo duomenų apdorojimui. Šie algoritmai naudojami įgyvendinti Gremlin kalbos funkcijas.

Taigi, Gremlin yra algoritmas ir aukšto lygio sistema , kurioje naudojami skirtingi algoritmai grafo duomenų apdorojimui ir analizei atlikti.

Šaltiniai

[1] „“TinkerPop“ Documentation“ Prieiga per internetą: https://tinkerpop.apache.org/docs/current/reference/#_tinkerpop_documentation. [žiūrėta: 2024 m. balandžio 28].

[2] „Practical Gremlin An Apache TinkerPop Tutorial“ Prieiga per internetą: https://kelvinlawrence.net/book/Gremlin-Graph-Guide.html#whygraph. [žiūrėta: 2024 m. balandžio 28].

[3] “Graph database — Using Cosmos DB (Gremlin API)” Prieiga per internetą:https://caiomsouza.medium.com/graph-database-using-cosmos-db-gremlin-api-ef0b6e0e517c. [žiūrėta: 2024 m. balandžio 28].

[4] ChatGPT. Prieiga per internetą: https://chat.openai.com. [žiūrėta: 2024 m. balandžio 28d.]