VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Baigiamasis bakalauro darbas

Grafų duomenų bazių valdymo sistemos

(Graph Database Management Systems)

Atliko: 4 kurso, 1 grupės studentas

Ignas Jacikas (parašas)

Darbo vadovas:

Irmantas Radavičius (parašas)

Recenzentas:

Dr. Rimantas Kybartas (parašas)

Vilnius
2016

Turinys

Anotacija ... 4
Summary ... 5
Įvadas ... 6
1. Grafai ir duomenų bazės ... 7
1.1. Savybių grafas ... 7
1.2. Pavyzdys ... 8
2. Reliacinės ir nereliacinės duomenų bazių valdymo sistemos ... 9
2.1. Reliacinės duomenų bazių valdymo sistemos ... 9
2.2. Grafų duomenų bazių valdymo sistemos ... 10
3. Rinkoje esantys grafų duomenų bazių sprendimai ... 12
3.1. ArangoDB ... 12
3.1.1. ArangoDB savybės ... 12
3.1.2. ArangoDB licenzijos ... 13
3.1.3. ArangoDB užklausų kalba ... 13
3.2. Neo4j ... 13
3.2.1. Neo4j savybės ... 13
3.2.2. Neo4j licencijos ... 14
3.2.3. Neo4j užklausų kalba ... 14
3.3. OrientDB ... 14
3.3.1. OrientDB Savybės ... 15
3.3.2. OrientDB licenzijos ... 15
3.3.3. OrientDB užklausų kalba ... 15
3.4. Titan ... 16
3.4.1. Titan Savybės ... 16
3.4.2. Titan licenzijos ... 17
3.4.3. Titan užklausų kalba ... 17
4. Sistemų palyginimas ... 18
4.1. Sistemų išskirtinės savybės ... 19
4.2. Duomenų tipai ... 19
4.3. Sistemų licenzijavimas ... 21
4.4. Sistemų palaikomos operacinės sistemos ... 21
4.5. Sistemų palaikomos aplikacijų programavimo sąsajos ... 22
4.6. Sistemų užklausų kalba ... 22

5. Sistemų testavimas 23
5.1. Duomenų bazių valdymo sistemų diegimas 23
5.2. Testinių duomenų pakrovimas ir užklausų greitaveika 24
6. Vertinimo kriterijai 26
7. Išvados 27
Literatūros sąrašas 28

Anotacija

Vienas iš naujesnių sprendimų duomenims saugoti yra nereliacinės grafų duomenų bazių valdymo sistemos. Šių sistemų duomenų saugojimo modelis yra paremtas grafų teorija, tai reiškia, kad juose duomenys saugomi kaip grafai. Todėl tokios sistemos gali būti naudingos saugant tarpusavyje labai susijusius duomenis.

Šiame darbe pristatomos grafų duomenų bazių valdymo sistemos. Šios sistemos palyginamos su reliacinėmis duomenų bazėmis. Apžvelgiami rinkoje esantys sprendimai. Jie tarpusavyje palyginami. Tai pat sukuriamas automatizuotas sprendimas šioms sistemoms testuoti, jis testuoja duomenų įterpimo ir užklausų vykdymo greitį. Įrankis pritaikytas ArangoDb, Neo4j, OrientDB ir Titan nereliacinėms duomenų bazių valdymo sistemoms. Darbo pabaigoje išskiriami vertinimo kriterijai, kurie gali padėti išsirinkti grafų duomenų bazių valdymo sistemą.

Summary

One of the newer solutions for data storage is graph database management systems. These systems data storage model is based on graph theory. This means that the data is stored as graphs. Such systems can be useful in storing data with a lot of connections.

This paper presents graphs database management systems. These systems are compared with relational databases. This paper reviews the currently popular graph database solutions and they are compared in this work. This document describes the automated testing system. Which testing data insertion and query processing performance. A tool is adapted for ArangoDb, Neo4j, OrientDB and Titan graph database management systems. At the end of the document there are described key criteria for choosing graph database management system.

Įvadas

Mes gyvename ryšių pasaulyje, čia nėra izoliuotos informacijos dalių, o tik susijusios tarpusavio ryšiais informacijos dalys. Pastaraisiais dešimtmečiais duomenų kaupimo srityje dominavo reliacinės duomenų bazių valdymo sistemos. Nors šiuo metu jos vis dar užima tvirtas pozicijas, bet jas mėgina išstumti nauji sprendimai, nereliacinės duomenų bazių valdymo sistemos, dar vadinamos kaip NoSQL duomenų bazės. Po šiuo pavadinimu išties slepiasi keletas duomenų kaupimo būtų skirtingiems uždaviniams spręsti. Taigi mes gyvename tarpusavio ryšių laikas, kuriuos labai gerai iliustruoja socialiniai tinklai, klausimas kaip sekasi šias problemas spręsti duomenų bazių valdymo sistemoms ir kaip iš didelės gausos naujų produktų tokių kaip grafų duomenų bazės išsirinkti tinkamiausią sprendimą iškylantiems uždaviniams spręsti.

Šiame darbe bus apžvelgtos duomenų bazių valdymo sistemos, trumpai aptarta kokiems uždaviniams spręsti jos skirtos. Daugiau dėmesio bus skiriama grafų duomenų bazėms, jų pranašumams prieš reliacinius sprendimus.

Darbo tikslas:

Pateikti rekomendacijas grafų duomenų bazės pasirinkimui, kuriomis vadovaujantis būtų galima pasirinkti produktą turimam uždaviniui spręsti.

Darbo uždaviniai:

- Apsibrėžti kas yra grafų duomenų bazių valdymo sistema ir kokiems uždaviniams jos skirtos spręsti.
- Surasti ir surinkti informaciją šiuo metu rinkoje pirmaujančių grafų duomenų bazių valdymo sistemas.
- Suskirstyti kūrėjų pateikiamą informaciją į grupes, pagal kurias galima būtų skirstyti sistemas.
- Atlikti nereliacinių grafų duomenų bazių valdymo sistemų testavimą įrašymo ir užklausų vykdymo efektyvumo testavimą.
- Išskirti vertinimo kriterijus kuriais remiantis būtų galima vertinti grafų duomenų bazes.

1. Grafai ir duomenų bazės

Visų pirma kas yra grafas? Paprastai tariant tai yra viršūnių ir briaunų rinkinys[RWE13]. Paprastai tariant grafas atspindi esybes ir ryšius kuriomis juos siejamos. Šiame skyriuje aptarinėsime populiariausia grafų duomenų modelį, kuris vadinasi savybių grafas. Pagrindinės šiuo modelio savybės yra:

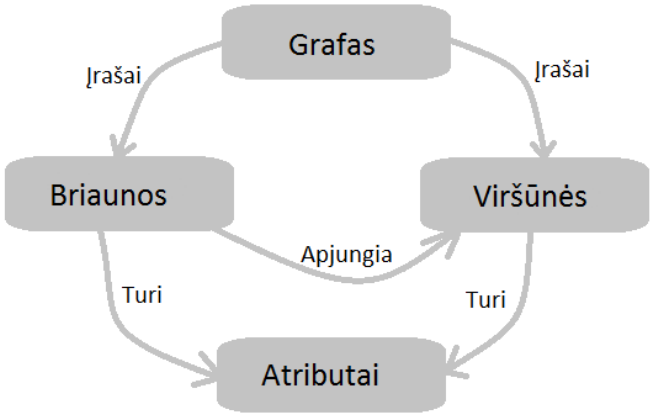
- Turi viršūnes ir briaunas.
- Viršūnės turi savybes.
- Briaunos turi pavadinimus arba antraštes yra kryptinės ir visada turi pradžios ir pabaigos viršūnę.
- Briaunos taip pat gali turėti savybes. [RWE13]

Dauguma žmonių savybių grafo duomenų modelį laiko intuityviu ir lengvai suprantamu. Šio modelio pagalba gana lengva atskleisti pagrindines šių sistemų savybes.

1.1. Savybių grafas

Objektiniai modeliai arba esybių ryšių diagramos yra labai panašios į savybių grafo domenų modelį. Savybių grafas saugo susijusias esybes (viršūnes), kurios gali saugoti daug atributų. Viršūnės gali turėti etiketes kurios sukuria skirtingus vaidmenis joms.

Ryšiai teikia pavadintus semantiškai susijusius kryptinius ryšius tarp dvejų viršūnių. Ryšys visada turi kryptį, tipą pradinę ir galutinę viršūnę. Kaip ir viršūnės briaunos gali turėti atributus.

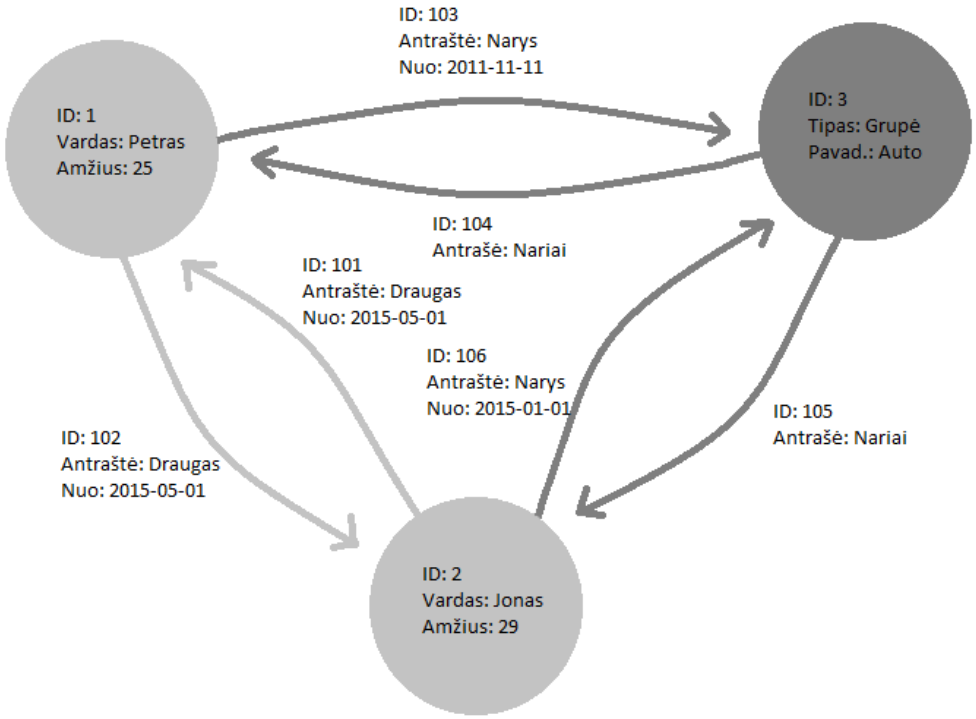


1 pav. Grafų duomenų bazės blokų kūrimo schema.

Grafų duomenų duomenų bazės turi vieną pagrindinę taisyklę: nėra neveikiančių nuorodų. Dėl to kad ryšys visada turi pradžios ir pabaigos viršūnę, negalima ištrinti viršūnės kuri turi ryšį su kita viršūne. Ir visada galima teikti, kad egzistuojanti briauna niekada nerodo į neegzistuojančią viršūnę. [RWE13]

1.2. Pavyzdys

Taigi grafų duomenų modelis turi viršūnes, briaunas ir savybes. Viršūnės gali atspindėti tokias esybes kaip asmuo, grupė ar bet kokį kitą objektą kurį norime išsaugoti. Briaunos jungia viršūnę su viršūne taip sukurdamos ryšius su jomis. Savybės papildoma informacija saugoma prie viršūnių ir briaunų. 2 pav. pavaizduojama paprasta schema atvaizduojanti bazėje galimą saugoti struktūrą.



2 pav. Paprasta grafų duomenų bazėje saugomų duomenų schema.

Pateiktoje schemoje matome viršūnes kuriose yra Jonas ir Petras jie sujungti briaunomis kurios nurodo, kad Jonas yra Petro draugas ir atvirkščiai, taip pat kiekviena briauna turi savybę nurodančią datą nuo kada jie yra draugai. Pateiktame pavyzdyje yra viršūnė kuri turi tipą ir jis yra grupė, o jos pavadinimas yra „Auto.“. Ši viršūnė turi briaunas rodančias į Joną bei Petrą jos antraštė nusako, kad jie yra tos grupės nariai. Taip pat matome briaunas einančias iš vaikinų ir nurodančias, kad kiekvienas iš jų yra narys toje grupėje. Taigi viršūnė kaip ir minėta anksčiau gali būti, bet koks objektas ir jis gali būti sujungtas briauna kuri nurodo viršūnių tarpusavio ryšius.

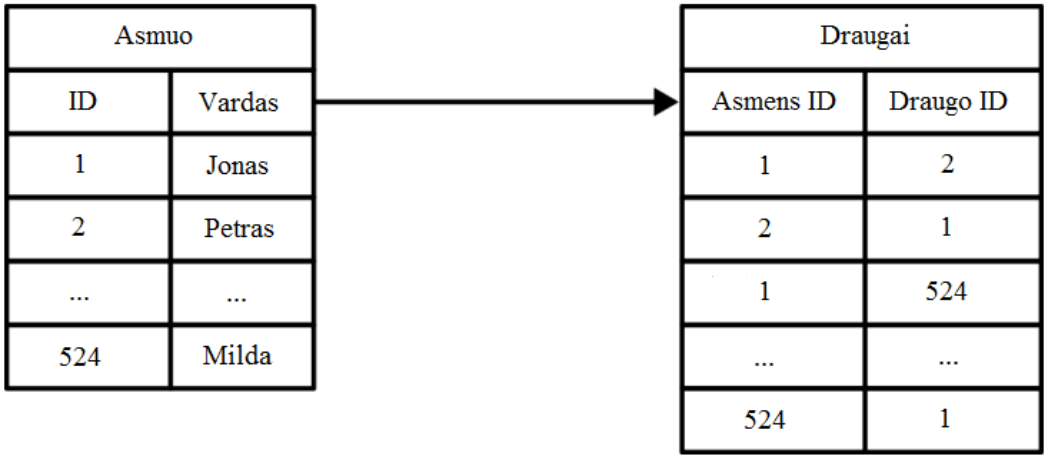
2. Reliacinės ir nereliacinės duomenų bazių valdymo sistemos

Šiame skyriuje apžvelgsime reliacines duomenų bazių valdymo sistemas ir nereliacines grafų duomenų bazių valdymo sistemas. Pateiksime duomenų modelio pavyzdžių išskirsime kuo grafų duomenų bazės skiriasi nuo reliacinių duomenų bazių, kaip skiriasi paieška vienose ir kitose duomenų bazių valdymo sistemose.

2.1. Reliacinės duomenų bazių valdymo sistemos

Nuo senų laikų reliacinės duomenų valdymo sistemos yra dominuojančios duomenų valdymo sistemos. Reliacinė duomenų bazė turi rinkinį lentelių, kurios turi rinkinį eilučių ir stulpelių. Eilutės apibrėžia objektus o stulpeliai jų savybes ar atributus. Eilutėje esantis stulpelis saugo savybė tos eilutės objekto. Problema yra sukurti tokį duomenų modelį sudarytą iš skirtingų lentelių, kuris išvengtų duomenų dubliavimosi. Šis procesas vadinamas duomenų normalizavimu. Kad skirtingos lentelės galėtų naudotis viena kitos duomenimis jos jungiamos raktų pagalba, kur vienos lentelės stulpelis nurodo į kitos lentelės stulpelį. Šių nuorodų eilės išlaikymas žinomas kaip nuorodinis vientisumas ir yra klasikinis reliacinės duomenų bazės modelis, užtikrinantis jos lankstumą.[ME92]

Taigi reliacinėse duomenų bazėse ryšiai egzistuoja. Jie naudojami skirtingų lentelių susiejimui tarpusavyje. Patys ryšiai neturi jokių savybių, kuo duomenų modelis darosi didesnis tuo daugiau ryšių atsiranda duomenų bazėje ir tuo sudėtingesnis darosi duomenų tarpusavio susiejimas. Toliau pateikiamas paprasčiausias pavyzdys 3 pav., kuriame pateikiamas reliacinės duomenų bazės, kurioje saugomi asmenys ir kurie iš jų yra draugai.



3 pav. Paprastas reliacinės duomenų bazės duomenų modelio pavyzdys.

Taigi tokiame paprastame pavyzdyje išsiaiškinti kas yra asmens draugai nėra sudėtinga tiesiog reikia patikrinti draugų lentoje užtenka tokios užklausos:

```
1   select
2     draugas.vardas
3   from
4     asmuo asmuo,
5     draugai draugai,
6     asmuo draugas
7   where
8     asmuo.id = draugai.asmens_id and
9     draugai.daugo_id = draugas.asmens_id and
10    asmuo.vardas = 'Jonas';
```

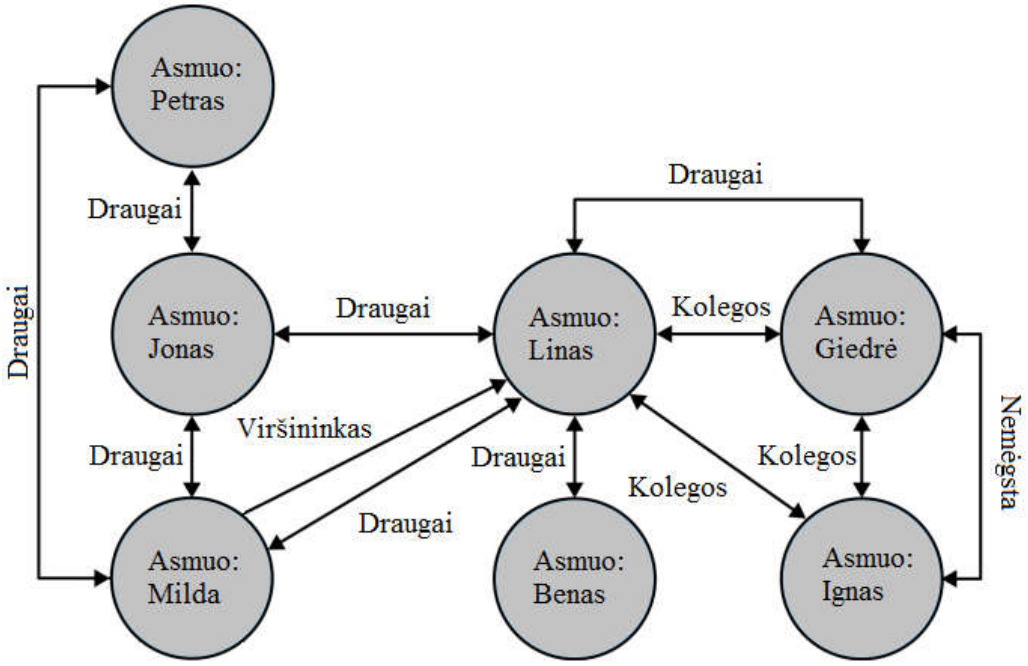
4 pav. Paprasta SQL užklausa.

Pavyzdyje 4 pav. matoma užklausa gražins iš duomenų bazės, kad Jonas turi du draugus tai Petrą ir Mildą. Tai nėra sudėtinga užklausa ir ji reliacinėse duomenų bazėse vykdoma greit ir jos kaštai nedideli, bet jei norėtume išsiaiškinti kokie yra Jono draugų draugai tokiu atveju sudėtingėja užklausa auga jos kaštai ir vykdymo laikas, ko gilesnis lygis draugų draugų paieškoje tuo labiau užklausos kaštai ir vykdymo laikas auga.

2.2. Grafų duomenų bazių valdymo sistemos

Visiška priešingybė yra grafų duomenų bazės. Kurios duomenų nesaugo skirtingose nuo modelio priklausančiose lentelėse. Vietoj to tokioje duomenų bazėje yra viena struktūra – grafas. Čia nėra raktų koncepcijos, dėl to kad kiekviena viršūnė turi briauną, kuri yra tiesioginis ryšys su gretima viršūne. Taigi duomenys jau yra sujungti briaunomis. Šitoks modelis turi savų pliusų ir minusų. Vienas iš didžiausių minusų, labai sudėtinga skaidyti grafą per skirtingas mašinas norint balansuoti sistemą horizontaliai. Grafų duomenų bazėje kur viršūnių ir briaunų santykiai nenuspėjami paprastai neegzistuoja nepriklausomi duomenų blokai. Nepaisant to, šios duomenų bazės turi didelį privalumą. Nepaisant grafo dydžio, gretimos viršūnės ar briaunos radimo trukmė yra konstanta. Todėl šios sistemos tokios patrauklios, nes keliai per viršūnės visada trunką apibrėžtą laiko trukmę. [RN10]

Pratęsiant asmenų tarpusavio draugystės ryšių saugojimo problemas aptartas skyriuje „ 2.1. Reliacinės duomenų bazių valdymo sistemos“ kaip jau minėta grafų duomenų bazėje duomenys nesaugomi nuo duomenų modelio priklausančiose lentelėse toliau schemoje 5 pav. pateikiamas pavyzdys duomenų saugomų grafe. Šiame pavyzdyje matosi keletas realiame gyvenime pasitaikančių ryšių. Taigi iš šio grafo iškart galime matyti daug aiškesnį vaizdą kas su kuo yra draugai, kas yra kolegos, ar kas ko nemėgsta ar yra vadovas. Socialiniai tinklai yra dažnas pavyzdys daug tarpusavio ryšiu turinčių duomenų, kurie nėra pilnai apibrėžtos struktūros ir gali lengvai kisti.



5 pav. Paprastas draugų, kolegų ir kitų ryšių grafas

Taigi grafų duomenų bazių valdymo sistema yra aktyvi duomenų bazių valdymo sistema su kūrimo, skaitymo, atnaujinimo ir šalinimo metodais, kuri išpildo grafų duomenų modelį. Galima būtų išskirti dvi architektūrines sistemų savybes kalbant apie grafų duomenų bazes:

- Duomenų saugojimas:

  Kai kurios grafų duomenų bazės naudoja įgimtą grafų saugojimo mechanizmą (angl. native graph storage) specialiai sukurtą ir optimizuotą grafu saugojimui ir darbui su jais. Bet ne visos grafų duomenų bazės naudoja tokį mechanizmą. Kitos sistemos grafus transformuoja į reliacinės, objektinės ar dar kokios kitos duomenų bazių valdymo sistemos duomenų saugojimo modelį.

- Duomenų apdorojimas:

  Grafų duomenų bazės nenaudoja indeksų, o viršūnė su viršūnę kurios yra sujungtos briaunomis rodo fiziškai viena į kitą.

Taigi galime išskirti esminį skirtumą nuo reliacinių duomenų bazių valdymo sistemų, grafų duomenų bazėse ryšiai yra svarbiausias komponentas, ne taip kaip kitose duomenų saugojimo sistemose nereikalingi tokie dalykai kaip išoriniai raktai. [RWE13]

3. Rinkoje esantys grafų duomenų bazių sprendimai

Šiame skyriuje bus apžvelgtos pagrindinės šiuo metu rinkoje pirmaujančių pagal „www.db-engines.com“ grafų duomenų bazių reitingą savybės. Apžvelgsime keturias duomenų bazes skirtas tik grafų saugojimui tai:

- Neo4j
- Titan

Ir dvi keleto modelių duomenų bazes, kurios gali saugoti ne vien grafus:

- ArangoDB
- OrientDB

Taip bus palygintos ne vien skirtingos grafų duomenų bazės, bet ir grafų duomenų modelio bazės su keleto modelių sprendimais.

3.1. ArangoDB

ArnagoDB yra atviro kodo keleto modelių nereliacinė duomenų bazių valdymo sistema, kuri gali saugoti ne tik grafų modelio, bet dokumento ir raktas-reikšmė duomenų modelio pavidalu duomenis. Kodą, sistemos versijas tai pat pranešti apie klaidas ar prašyti naujų sistemos patobulinimų galima GitHub sistemoje. Šiame skyriuje trumpai apžvelgsime kūrėjų dokumentacijose išskiriamus jų kūrinių privalumus. [ARA]

3.1.1. ArangoDB savybės

ArangoDb duomenų bazė įgyvendina lankstų rinkinį dokumento, raktas-reikšmė ir grafų duomenų modelį. Pagrindinės savybės:

- Keleto modelių duomenų bazė, kuri leidžia duomenis apdoroti ir pasiekti kaip raktas-reikšmė arba grafus, taip palikdama laisve naudotojui dirbti su duomenimis kaip jam patogiau ar reikia atitinkamu atveju.
- Galimybė esant dideliam duomenų kiekiui juos išskaidyti apibrėžtomis grupėmis arba automatiškai po skirtingus serverius, kuriuos galima pasiekti per koordinatorių, kurių gali būti nevienas norint balansuoti apkrovas.
- ArnagoDB gali būti išplėsta naudojant JavaScript. JavaScript kodas yra įvykdomas tiesiai duomenų bazėja naudojant greitą V8 JavaScript varikliuką.
- Užklausų rašymui naudojama ArangoDb užklausų kalba (AOL), kuri yra šiek tiek panaši į struktūrizuotą užklausų kalbą (angl. trumpinys: SQL). [ARA]

3.1.2. ArangoDB licenzijos

ArangoDB yra atviro kodo ir yra nemokama. Bendruomenė gali lengvai pranešti apie klaidas jas nurodant GitHub sistemoje. Tai pat ten galima prašyti patobulinimų.

3.1.3. ArangoDB užklausų kalba

Užklausų rašymui naudojama ArangoDb užklausų kalba (AQL), kuri yra šiek tiek panaši į struktūrizuotą užklausų kalbą (angl. trumpinys: SQL). AQL palaiko įrašymo, atnaujinimo ir skaitymo galimybes, bet nepalaiko duomenų apibūdinimo savybių, tokių kaip sukūrimo arba šalinimo duomenų bazės, tipų ar indeksų. Kūrėjai teigia, kad sintaksė skiriasi nuo struktūrizuotos užklausų kalbos nors kai kurie raktiniai žodžiai ir sutampa, bet AQL turėtų būti suprantama kiekvienam kas turi pagrindines žinias apie SQL užklausų rašymą. Toliau pateikiamas AQL užklausos 6 pav. pavyzdys:

```
1
2  FOR d IN 1 OUTBOUND 'asmuo/Jonas' GRAPH 'GrafoPavadinimas' RETURN d.vardas
3
```

6 pav. AQL užklausų kalbos pavyzdys.

Pateikta užklausa atrenka visus Jono draugus ir išveda jų vardus. Kaip matosi iš užklausos iš tiesų AQL užklausų kalba nelabai turi kažką bendro su struktūrizuota užklausų kalba.

3.2. Neo4j

Neo4j yra atviro kodo nereliacinė grafų duomenų bazių valdymo sistema įgyvendinta JAVA ir SCALA kalbomis. Kodą ir sistemos versijas galima rasti GitHub sistemoje. Ši duomenų bazė yra viena populiariausių grafų duomenų bazių. Šiame skyriuje trumpai apžvelgsime kūrėjų dokumentacijose išskiriamus jų kūrinių privalumus.

3.2.1. Neo4j savybės

Neo4j duomenų bazė įgyvendina savybių grafo duomenų modelį. Keletas duomenų bazių valdymo sistemos savybių:

- Ryšiai materializuojami kūrimo metu, todėl nereikia vykdyti sudėtingų apdorojimo procesų užklausų vykdymo metu.
- Visi ryšiai Neo4J duomenų bazėje yra vienodai svarbūs ir greiti, tai sudaro galimybę, realizuoti ir naudoti naujus ryšius greičiau kai atsiranda toks poreikis.

- Kompaktiškas saugojimas ir atminties rezervavimas grafams nulemia tai, kad galima proporcingai didinti ir saugoti milijardus viršūnių vienoje duomenų bazėje naudojant vidutinę techninę įrangą. [NEO]

3.2.2. Neo4j licencijos

Neo4J bendruomenės versija yra nemokama ir atviro kodo duomenų bazė. Ši versija įgyvendina visą skyriuje „Neo4j savybės“ išvardintą funkcionalumą. Įmonėms skirta versija įgyvendiną visą bendruomenės versijos funkcionalumą ir papildomai turi šias savybes:

- Keičiamo dydžio grupavimas.
- Automatinį sistemos stabdymą.
- Didelės laiko dalies pasiekiamumą.
- Atsarginių kopijų darymo galimybę veikiant sistemai.
- Išsamų monitoringą. [NEO]

3.2.3. Neo4j užklausų kalba

Neo4J yra deklaratyvi, SQL įkvėpta užklausų kalba grafų modelių apibūdinimui. Ji leidžia apibūdinti ką norime atrinkti, įterpti, atnaujinti arba ištrinti iš grafų duomenų bazės nereikalaudama apibūdinti kaip konkrečiai tai atlikti. Pateikiamas pavyzdys 7 pav.:

```
1
2  MATCH
3    (asmuo:asmuo{vardas:'Jonas'})-[:draugas]->(vardas)
4  RETURN
5    asmuo.vardas, draugas.vardas
6
```

7 pav. Neo4j užklausų kalbos pavyzdys.

Pavyzdyje 7 pav. pateikiamas paprastas pavyzdys kaip Neo4j užklausų kalba atrinkti visus asmens draugus, kaip matome užklausoje ji gražins visus Jono draugus. Užklausa panašumų su struktūrizuota užklausų kalba neturi.

3.3. OrientDB

OrientDB yra atviro kodo keleto modelių dokumentų ir grafų duomenų bazių valdymo sistema, kurioje įgyvendintas įgimtasis grafų apdorojimas ir saugojimas sujungtas su sprendimais, kurie paprastai aptinkami tik dokumentų duomenų bazių valdymo sistemose. [ORI]

3.3.1. OrientDB Savybės

Taigi OrientDB yra grafų ir dokumentų nereliacinė duomenų bazių valdymo sistema, šiame skyriuje išskiriamos kūrėjų labiausiai pabrėžiamos savybės:

- Realiu laiku tranzakcijas apdorojanti duomenų bazė, kurios duomenis galima atstatyti po netikėto sistemos veikimo sutrikimo ar lūžimo, visos laukiančios tranzakcijos automatiškai gražinamos į ankstesnę būseną.
- Dydžio keičiamumas, sistema palaiko keleto pagrindinių serverių ir horizontalaus duomenų skaidymo architektūrą.
- Platus duomenų tipų palaikymas.
- Schemos apibrėžiamumas turi skirtingus lygius:
    - Pilnai apribojama, kai kuriamos klasės turi nurodytą savybių struktūra ir ji yra privaloma
    - Visai neribojama, kai kuriamos klasės neturi jokių iš anksto numatytų savybių ir gali būti kuriamos nepriklausomai nuo modelio.
    - Hibridinis ribojimas, kai keletas savybių yra apibrėžtos iš anksto, bet įrašai gali apsibrėžti savo savybes kurios nenumatytos iš anksto. [ORI]

3.3.2. OrientDB licenzijos

OrientDB kaip ir daugelis nereliacinių sprendimų turi savo bendruomenės versiją, kuri yra atviro kodo. Šią versiją galima naudoti ir komerciniuose projektuose nemokamai. OrientDB taip pat turi išleidę ir mokamą versiją, kuri iš esmės yra tapati OrientDB grafų nereliacinė duomenų bazių valdymo sistema su keletu papildomų funkcijų tokių kaip:

- Metrinė analizė, serverių apkrovų analizavimas ir jų gerinimas.
- Užklausų optimizatorius, galimybė rasti užklausas, kurios stabdo sistemos veikimą.
- Konfigūruojami įspėjimai.
- Besikartojanti foninė užduotis atsarginių kopijų darymui.
- Duomenų auditas.
- Pilnas produkto palaikymas.

3.3.3. OrientDB užklausų kalba

OrientDB užklausų kalba yra sukurta struktūrizuotos užklausų kalbos (angl. trumpinys: SQL) pagrindu turinti keletą papildymų darbui su medžiais ir grafais. Kaip patys kūrėjai teikia daugelis

programuotojų yra pažįstami su SQL tipo užklausomis todėl pradėti naudoti OrientDB užklausų kalba yra gana paprasta. Toliau pateikiamas paprastas užklausos 8 pav. pavyzdys:

```
1
2   SELECT
3     vardas, out('draugas').vardas
4   FROM
5     asmuo
6   WHERE
7     vardas = 'Jonas'
8
```

8 pav. OrientDB užklausų kalbos paprastas pavyzdys

8 pav. pavyzdyje pateikiama asmens vardu Jonas draugų atrinkimo užklausa. Užklausa gražina visus to asmens draugus saugomus duomenų bazėje. Kaip matome iš pavyzdžio OrienDB užklausų kalba turi panašumų su struktūrizuota užklausų kalba, bendrai išlaikoma šios kalbos struktūra papildant ją galimybe dirbti su grafais.

3.4. Titan

Titan yra keičiamo dydžio (angl. scalable) grafų duomenų bazė skirta grafų turinčių milijardus viršūnių ir briaunų saugojimui ir peržiūrėjimui paskirstytų per tarpusavyje susietų atskirų mašinų masyvus. [TIT] Šiame skyriuje trumpai apžvelgsime kūrėjų dokumentacijose išskiriamus jų kūrinių privalumus.

3.4.1. Titan Savybės

Titan įgyvendina grafų duomenų modelio valdymo sistemą, kuri orientuojasi į didelius grafus, kurių saugojimui neužtenka vienos mašinos. Kaip patys kūrėjai deklaruoja jie koncentruojasi į tokio tipo grafų kompaktišką serializavimą, visavertį duomenų modeliavimą ir efektyvų užklausų vykdymą.

- Labai didelių grafų palaikymas.
- Labai didelių kiekių lygegrečių užklausų ir operacijų su grafais palaikymas.
- Skirtingų duomenų saugojimo sprendimų palaikymas, tokių kaip Cassandra, HBase, OracleBerkelyDB
- Titan integruojasi su daug įvairių sprendimų darbui su grafais analizei, paieškai operavimui ar atvaizdavimui. [TIT]

3.4.2. Titan licenzijos

Kaip ir dauguma nereliacinių sprendimų Titan yra atviro kodo nereliacinė duomenų bazių valdymo sistema. Kuri iš esmės išplečia kitų sistemų Cassandra, HBase, ir Hadoop galimybes jas papildant grafų atvaizdavimo ir apdorojimo galimybėmis.

3.4.3. Titan užklausų kalba

Gremlin yra Titan nereliacinės duomenų bazės užklausų kalba skirta gauti ir modifikuoti duomenis grafuose. Gremlin yra kelio orientacijos kalba, kuria glaustai galima išreikšti sudėtingas grafų kelio ir modifikavimo operacijas. Tai yra funkcinė kalba, kurioje kelionės operatoriai yra sujungti į grandinę, kad suformuotų į kelią panašias išraiškas. Toliau pateikimas Titan naudojamos Gremlin užklausų kalbos 9 pav. pavyzdys:

```
1
2   g.V('Petras').outE('draugas')
3
```

9 pav. Gremlin užklausų kalba.

Pavyzdyje 9 pav. pateikiamas paprastas Gremlin užklausos pavyzdys, ši užklausa atrenka ir gražina visus Petro draugus. Kaip matome iš pateikto pavyzdžio tai nieko bendro su struktūrizuota užklausų kalba neturinti kalba, kaip kūrėjai ir teigia ji yra funkcinė.

4. Sistemų palyginimas

Ankstesniuose skyriuose trumpai apžvelgėme keturias populiarias nereliacines grafų duomenų bazių valdymo sistemas. Šiame skyriuje jas lyginant išskirsime vertinimo kriterijus pagal kūrėjų pateikiamą informaciją. Toliau pateikiama palyginamoji lentelė, kurioje lyginamos pagrindinės grafų duomenų bazių savybės, kurios gali turėti įtakos pasirenkant sistemą. Žiūrėti lentelę: 1 lentelė.

1 lentelė. Grafų duomenų bazių valdymo sistemos.

Duomenų bazė	ArangoDB	Neo4j	OrientDB	Titan
Aprašymas	Keleto modelių DBVS	Grafų duomenų bazė	Keleto modelių DBVS	Grafų DBVS optimizuota klasteriams
Duomenų bazės modelis	Grafų DBVS, Dokumento, Raktas-reikšmė	Grafų DBVS	Grafų DBVS, Dokumento, Raktas-reikšmė	Grafų DBVS
Pradinė versija	2012	2007	2010	2012
Dabartinė versija	2.8.7, Gegužė 2016	3.0.0, Balandis 2016	2.1.15, Balandis 2016	1.0.0, Rugsėjis 2015
Licenzija	Apache 2	GPLv3, išplėstinė komercinė	Apache 2	Apache 2
Įgyvendinimo kalba	C, C++, JavaScript	Java	Java	Java
Palaikomos programavimo kalbos	C# Clojure Java JavaScript (Node.js) PHP Python Ruby	.Net Clojure Go Groovy Java JavaScript Perl PHP Python Ruby Scala	.Net/ C# C C++ Clojure Java JavaScript JavaScript (Node.js) PHP Python Ruby Scala	Clojure Java Python

4.1. Sistemų išskirtinės savybės

Visos nagrinėjamos nereliacinės grafų duomenų bazių valdymo sistemos turi savybių kurias save pristatydami ar lygindami su kitais sprendimais išskiria labiausiai. Šiame skyriuje jas ir aptarsime.

ArangoDB ir OrientDB yra keleto modelių nereliacinės duomenų bazių valdymo sistemos leidžiančios saugoti ne tik grafus bet ir dokumentus, bei raktas-reikšmė. Tuo tarpu Neo4j ir Titan yra tiesiog grafų duomenų bazių valdymo sistemos. Taip pat reiktų pastebėti, kad ArangoDB labiau orientuota į dokumentų ir rakto-reikšmės saugojimą nei į grafus. Jos duomenų modelis labiau primena reliacinės duomenų bazės modelį kur viename rinkinyje saugomos viršūnės, o kitame briaunos.

OrientDB save pristato kaip vienintelę iš nagrinėjamų šiame darbe grafų duomenų bazių valdymo sistemą, kuri yra realiu laiku tranzakcijas apdorojanti duomenų bazė, kurios duomenis galima atstatyti po netikėto sistemos veikimo sutrikimo ar lūžimo, visos laukiančios tranzakcijos automatiškai gražinamos į ankstesnę būseną. Bet iš tiesų ji nevienintelė patenkanti į taip apibūdintų sistemų sąrašą[FAH+15], kartu su ja sąraše yra ir Neo4j. Aišku šios sistemos nepatenka tarp lyderių, kur puikuojasi tokie vardai kaip Oracle ar Microsoft duomenų bazės.

Duomenų skaidymas horizontaliai į skirtingus serverius, grafų duomenų bazių sistemose tai gana sudėtingas uždavinys, bet tokį funkcionalumą palaiko ArangoDB, OrientDb ir Titan, vieninteliai Neo4j palaiko serverių replikavimą, kur vienas iš serverių yra pagrindinis. Taip esant didelėmis apkrovomis gali kentėti įrašymo greitis, taip pat nustojus veikti pagrindiniam serveriui reikalingas laiko tarpas kol kuris nors iš kitų serverių tampa pagrindiniu. Kitose sistemose tokios problemos nėra.

4.2. Duomenų tipai

Saugant duomenis nereliacinėje grafų duomenų bazių valdymo sistemoje gali būti svarbu kokiu formatu grafo viršūnių ar briaunų savybių duomenys bus saugomi. Toliau 2 lentelėje pateikiamas mūsų analizuojamų duomenų bazių duomenų tipų palyginimo lentelė.

2 lentelė. ArangoDB, Neo4j, OrientDB, Titan duomenų tipų palyginimas.

ArangoDB	Neo4j	OrientDB	Titan
Neturi	Boolean – loginis kintamas	Boolean – loginis kintamas	Boolean – loginis kintamas
Neturi	Int – sveikas skaičius 32 bitų	Integer – sveikas skaičius 32 bitų	Int – sveikas skaičius 32 bitų
Neturi	Short – sveikas skaičius 16 bitų	Short – sveikas skaičius 16 bitų	Short – sveikas skaičius 16 bitų
Neturi	Long – sveikas skaičius 64 bitų	Long – sveikas skaičius 64 bitų	Long – sveikas skaičius 64 bitų
Neturi	Float – slankiojo kablelio skaičius tik trys skaičiai po kablelio	Float – slankiojo kablelio skaičius	Float – slankiojo kablelio skaičius
Neturi	Double – slankiojo kablelio skaičius tik trys skaičiai po kablelio	Double – slankiojo kablelio skaičius	Double – slankiojo kablelio skaičius
Neturi	Neturi	Datetime – data miliskundžių tikslumu	Neturi
String – tekstinė eilutė	String – tekstinė eilutė	String – tekstinė eilutė	String – tekstinė eilutė
Neturi	Neturi	Neturi	Geoshape – geografinė figūra

Lentelėje išvardinti tik pagrindiniai ir nėra išvardinti absoliučiai visi duomenų tipai kuriuos palaiko šios keturios duomenų bazės. Bet pagal lentelę galime matyti, kad iš esmės ArangoDB palaiko tik tekstinės eilutės duomenų tipą ir į jį viską saugo, todėl norint operuoti bet kokiais kitais tipais dirbant su šia duomenų bazių valdymo sistema bus neįmanoma išskyrus juos saugoti ir peržiūrėti. Plačiausią duomenų tipų palaikymą turi OrientDB duomenų bazė, nuo jos nedaug atsilieka ir Titan, bei Neo4j, jos neturi datos formato, kas duomenų bazėse dažnai gali būti labai svarbus tipas. Neo4j taip pat gali saugoti skaičius po kablelio tik 3 skaitmenų tikslumu, tad reikiant didesnio tikslumo gali kilti problemų. Titan vienintelė iš visų duomenų bazių gali saugoti geografines figūras.

4.3. Sistemų licenzijavimas

Visos apžvelgiamos sistemos save pristato kaip atviro kodo nemokamus sprendimus, tai pat su galimybe pirkti komercinę versiją. Komercines versijas galima būtų skirstyti į dvi grupes tai kai mokama už palaikymą arba mokamoji versija turi daug papildomų būtinų tokiai sistemai funkcijų. Mūsų nagrinėtais atvejais iš esmės trys sistemos yra licencijuojamos pagal „Apache 2.0“ licenziją ir taip pat galima pirkti papildomą sistemų palaikymą šioms nereliacinėms grafų duomenų bazių valdymo sistemoms:

- ArangoDB
- OrientDB
- Titan

Vienintelė duomenų bazė licencijuojama pagal „GPL v3“ licenziją:

- Neo4J

Taigi Neo4j yra vienintelė iš nagrinėjamų nereliacinė grafų duomenų bazių valdymo sistema, kurios komercinė versija nėra vien palaikymas, ją įsigyjant taip pat galima naudoti daug papildomų funkcijų, visos jos aprašytos skyriuje „3.2.2. Neo4j licenzijos“. Iš esmės kaip patys kūrėjai teigia jei jūs projektas atviro kodo tai jūs galite naudoti bendruomenės nemokamą versiją, bet jei jūsų projektas komercinis, tai jūs turite pirkti komercinę versiją.

Taigi pasirenkant sistemą būtina atkreipti dėmesį į licenzijavimą, žinoti ko jums reikia iš sistemos ir įsivertinti ar tai nekainuos papildomai.

4.4. Sistemų palaikomos operacinės sistemos

Pasirenkant sistemą taip pat svarbu kokioje aplinkoje grafų duomenų bazių valdymo sistema gali veikti. Toliau pateikiamoje 3 lentelėje matome, kad visos duomenų bazės iš esmės palaiko

3 lentelė. Operacinių sistemų palaikymas.

	ArangoDB	Neo4j	OrientDB	Titan
Linux	Taip	Taip	Taip	Taip
Mac OS X	Taip	Taip	Taip	Taip
Raspbian	Taip	Ne	Taip	Ne
Solaris	Taip	Ne	Taip	Ne
HP-UX	Ne	Ne	Taip	Ne
IBM AIX	Ne	Ne	Taip	Ne
Unix	Ne	Ne	Ne	Taip
Windows	Taip	Taip	Taip	Taip

pagrindines operacines sistemas. Taigi iš esmės tai gali būti svarbus vertinimo kriterijus tuo atveju jei nereliacinė grafų duomenų bazių sistema bus talpinama aplinkoje, kurios operacinė sistema yra viena iš rečiau naudojamų.

4.5. Sistemų palaikomos aplikacijų programavimo sąsajos

Viena iš svarbių duomenų bazių valdymo sistemų savybių yra palaikomos aplikacijų programavimo sąsajos. Pagal 1 lentelę matome, kad iš esmės visos keturios nagrinėjamos nereliacinės grafų duomenų bazių valdymo sistemos palaiko tris aplikacijų programavimo sąsajas:

- Clojure
- Java
- Python

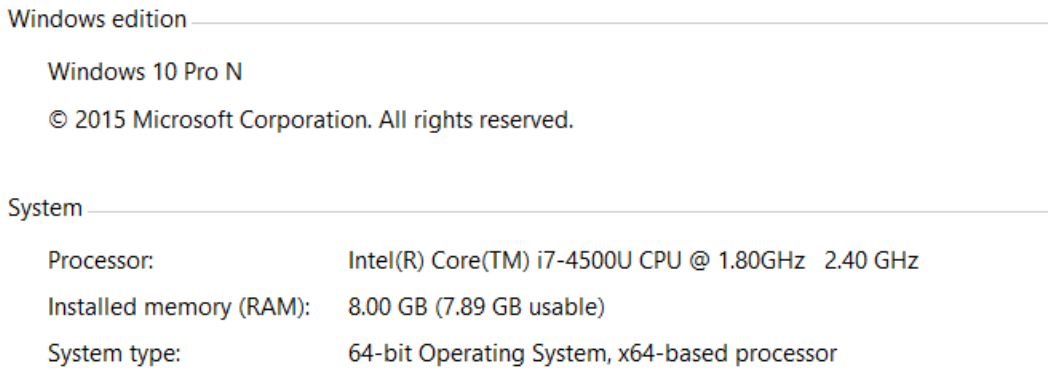
Trys sistemos ArangoDB, OrientDB ir Neo4j palaiko gerokai daugiau aplikacijų programavimo sąsajų už Titan. Daugiausia palaikomų sąsajų turi OrientDb sistema ir pirmauja tarp visų sprendimų nuo jos ne daug atsilieka Neo4J. Reiktų atkreipti dėmesį kad šis sąrašas nėra pilnas, kadangi tai yra atviro kodo sistemos dalis sąsajų yra kurtos trečiųjų šalių ir oficialiai naudojamos.

4.6. Sistemų užklausų kalba

Kaip buvo aptarta ankstesniuose skyriuose paprastai nereliacinės duomenų bazių valdymo sistemos turi savo užklausų kalbas, ne taip kaip reliaciniuose sprendimuose, kur duomenų bazių valdymo sistemos palaiko vieną standartą, čia nėra vienos standartizuotos kalbos. Savo kalbas pristatydami dauguma rašo, kad jos paremtos ar įkvėptos standartizuotos užklausų kalbos. Skyriuose: 3.1.3., 3.2.3., 3.3.3., 3.4.3. buvo aprašomos užklausų kalbos skirtos būtent toms duomenų bazių valdymo sistemoms. Palyginus visas kalbas tarpusavyje matosi, kad panašiausia į struktūrizuotą užklausų kalbą yra OrientDB kalba, kas gali būti privalumas pradedant naudoti naują sistemą ar keičiant reliacinį sprendimą į nereliacinę grafų duomenų bazių valdymo sistemą. Programuotojams lengviau pereiti prie kalbos, kuri jau yra pažįstama. Bet Gremlin funkcinė kalba atrodo paprasčiausia ir nėra tiesiog Titan grafų duomenų bazės sprendimas tai atskiras Titan kūrėjų atviro kodo produktas, kurį galima laisvai naudoti ir pritaikyti kitoms sistemoms, ji jau yra pritaikyta Neo4j sistemai.

5. Sistemų testavimas

Šiame skyriuje apžvelgsime analizuojamų nereliacinių grafų duomenų bazių valdymo sistemų bandymų rezultatus, kaip gerai paruošti diegimo paketai darbui nuo parsisiuntimo iki paleidimo, kiek laiko sugaištama norint paleisti duomenų bazių valdymo sistemą. Dokumentacijos aiškumą ir išbaigtumą darbo pradžiai, aplikacijų programavimo sąsajų panaudojimui. Taip pat patikrinsime duomenų įterpimo ir užklausų atsakymo efektyvumą. Testavimui buvo naudojama sistema su Windows 10 Pro operacinė sistema su Intel Core i7 procesoriumi 8 GB operatyviosios atminties ir 256 GB SSD kietuoju disku. Detalesnis sistemos aprašymas matomas pateiktame 10 pav. vaizde.



10 pav. Naudojamos sistemos parametrai

5.1. Duomenų bazių valdymo sistemų diegimas

Pradedant darbą su visomis sistemomis tikėsi, kad jų paketai bus pilnai paruošti visoms viešai nurodomoms naudoti operacinėms sistemoms ir nuo diegimo pradžios iki pabaigos nekils jokių problemų. Tai pat dokumentacijoje bus aprašyti visi žingsniai ką ir kaip reikia padaryti norint pradėti darbą.

ArangoDB sistemos 2.8.9. versijos diegimo parsisiuntimas iškart prasideda nuo pasirinkimo į kokią sistemą bus diegiama duomenų bazė. Parsisiuntus duomenų bazė buvo greit per keletą sekundžių ir be jokių problemų sudiegta, startuotas serveris ir atidaryta pradinė internetinės naršyklės sąsaja. Naudojimo metu vykdant nesudėtingas užklausas su nedideliais grafais duomenų bazė nustojo veikti bent kelis kartus. Su Neo4j ir OrientDB taip pat nekilo jokių problemų diegiant sistemas, taip pat jokių problemų nekilo ir bandant užklausas. Parsisiuntus Titan duomenų bazių valdymo sistemą gana ilgai nepavyko jos paleisti. Atlikus paieška internete ir radus keletą pasiūlytų problemos sprendimų būdų iš trečiųjų šalių, o ne iš pačių sistemos kūrėjų, teko modifikuoti paleidimo failą ir po kurio laiko sistemą pavyko paleisti ir išbandyti Gremlin užklausų kalbą.

Pradedant naudoti aplikacijų programavimo sąsają, su visų nereliacinių grafų duomenų bazių sistemų komunikavimo buvo naudota JAVA kalba ir jai skirti sprendimai. Dokumentacijose buvo

pateikta pavyzdžių, bet nevisi pavyzdžiai veikė teko atlikti kodo modifikacijas norint išbandyti paprastą funkcionalumą. Kaip pradėti naudotis šiomis sąsajomis geriausiai buvo aprašyta ArangoDB dokumentacijoje viskas buvo išvardinta pažingsniui ir paaiškinant taip, kad net mažai patyręs naudotojas suprastų.

5.2. Testinių duomenų pakrovimas ir užklausų greitaveika

Sistemų testavimui buvo pasirinkta parašyti automatinę sistemą JAVA kalba, kuri visoms keturioms nereliacinėms duomenų bazių valdymo sistemoms pakrauna duomenis ir atlieka užklausų testavimą. Sistema išveda laiką kiek truku atlikti kiekvieną užduotį, kad duomenys būtų tikslesni tas pats veiksmas buvo atliekamas po 5 kartus.

Testavimui buvo pasirinktas nedidelis grafas sudarytas iš 5242 viršūnių ir 28980 briaunų. Testiniai duomenis gauti iš Standfordo didelių tinklų rinkinių saugyklos.[STA] Atlikus duomenų pakrovimo į duomenų bazes bandymus gauti rezultatai atvaizduojami 4 lentelėje.

4 lentelė. Duomenų pakrovimo trukmė.

ArangoDB	Neo4j	OrientDB	Titan
~180s	~2.5s	~1.7s	~7s

OrientDB ir Neo4j sistemos pasirodė neblogai ir duomenys į jas abi buvo įterpti panašiai greitai. Labai prastai šiame bandyme pasirodė ArangoDB, nors duomenų terpimo logika visais atvejais vienoda, skiriasi tik naudojamos aplikacijų programavimo sąsajos.

Toliau buvo bandomas šių nereliacinių grafų duomenų bazių sistemų užklauso vykdymo greitis pakrautuose duomenyse ieškant skirtingu lygiu nutolusių viršūnių. Toliau pateikiama 5 lentelė su testavimo rezultatais.

5 lentelė. Duomenų atrinkimo trukmė.

Lygis	ArangoDB	Neo4j	OrientDB	Titan
1 lygio	0.0010s	0.0101s	0.0050s	0.0030s
2 lygio	0.0030s	0.0150s	0.0170s	0.0175s
3 lygio	0.0202s	0.0202s	0.0249s	0.0345s
4 lygio	0.1961s	0.0242s	0.0317s	0.0534s
5 lygio	1.4892s	0.0300s	0.0421s	0.0748s

Kaip ir pirmajame teste prasčiausiai pasirodė ArangoDB nereliacinė keleto modelių valdymo sistema. Neo4j sistema parodė stabiliausią užklausų vykdymo rezultatą, jos buvo vykdomos panašiu greičiu visais bandymais. OrientDB pagal vidutinį vykdymo greitį nedaug atsiliko nuo Neo4j.

Atlikus duomenų bazių sistemų testavimus galima išskirti, nors ir visos duomenų bazės savo pristato kaip galinčias dirbti su grafais, bet pastebimai ArangoDB duomenų bazės ir užklausų

vykdymo laikai ir duomenų pakrovimo trukmė buvo gerokai didesnė. Iš programavimo pusės išskirčiau dokumentacijos stoką, dažnai dokumentacijoje pateikiama informacija yra pasenusi ir tam tikras funkcionalumas aprašomas joje veikė ant senesnių sistemų versijų, o naujas atitikmuo neaprašytas, dažnai teko ieškoti internete galimų sprendimų.

6. Vertinimo kriterijai

Šiame skyriuje apžvelgsime į ką reiktų atkreipti dėmesį pasirenkant nereliacinę duomenų bazių valdymo sistemą.

- Duomenų saugumas.

Vienas iš esminių duomenų bazių uždavinių duomenų saugojimas, taigi labai svarbu esant sistemos sutrikimams kaip bus elgiamasi su tranzakcijomis ar duomenys, kurie jau yra sistemoje nedings ir kaip bus elgiamasi su laukiančiomis tranzakcijomis iš nagrinėtų sistemų OrientDB ir Neo4j yra įtrauktos į realiu laiku tranzakcijas apdorojančių sistemų sąrašą. Šį kriterijų tarp nagrinėtų sistemų labiausiai atitinka OrientDB, tai yra vienas iš jos kūrėjų akcentuojamų privalumų taip pat ir realiu laiku veikiant sistemai galimybė daryti atsargines sistemos kopijas.

- Saugomi duomenų tipai.

Svarbus kriterijus, kokius duomenis planuojama saugoti duomenų bazėje. Darbo metu buvo pastebėta, kad ne visos sistemos kaip grafo savybes gali saugoti įprastus duomenų bazėm tipus. Dažnai tokios savybės kaip data saugomos kaip paprastos teksto eilutės todėl norint su jomis operuoti gali kilti problemų. Daugiausia palaikomų tipų turi OrientDB duomenų bazė, todėl ją reiktų rinktis norint saugoti įvairius tipus jiems skirtuose laukuose. Jei reikia saugoti geografines figūras tokių tipų palaikymą turi Titan duomenų bazių valdymo sistema.

- Kokiam tikslui naudojama sistema

Renkantis sistemą reiktų atkreipti dėmesį, kokiu tikslu ji bus naudojama, tai bus atviro kodo projektas ar komercines. Įgyvendinat atviro kodo projektus reiktų atkreipti dėmesį ar visos reikalingos funkcijos yra atviro kodo duomenų bazės versijoje. Neo4j atveju nemažai papildomų funkcijų yra tik komercinėje versijoje. Kitų sistemų komercinės versijos iš esmės yra tik palaikymas

- Kūrimo aplinka

Svarbus kriterijus ar sistema turi gerai dokumentuotas ir oficialiai palaikomas aplikacijų programavimo sąsajas. Darbo metu buvo pastebėta, kad ne visos sistemos turi geras dokumentacijas, taip pat sąsajos yra pakitusios tarp versijų. Geriausiai dokumentuotos sistemos buvo Neo4j ir OrientDB ir lengviausia išbandyti buvo Neo4J sistemą. Pagal šį kriterijų reiktų rinktis Neo4J sistemą.

- Efektyvumas

Darbo metu nustatyta, kad iš visų sistemų savo duomenų įterpimo ir užklausų vykdymo efektyvumu gerokai atsilieką nuo kitų sprendimų. Todėl renkantis sistemą reiktų palyginti skirtingų sistemų užklausų vykdymo ir duomenų įterpimo efektyvumus. Atlikti bandymai parodė, kad geriausia sistema yra Neo4J.

7. Išvados

Šiame darbe buvo pristatytos nereliacinės grafų duomenų bazių valdymo sistemos, parodyta kuo jos skiriasi nuo reliacinių duomenų bazių valdymo sistemų. Ir pastebėta, kad jos gali būti labai geras įrankis saugant labai tarpusavyje susijusius duomenis.

Darbo metu buvo apžvelgtos 4 populiarios grafų duomenų bazių valdymo sistemos, išskirtos jų savybės, taip pat apžvelgtos užklausų kalbos. Pastebėta, kad Gremlin užklausų kalba galima naudoti ne vien Titan sistemai, bet ji taip pat pritaikoma ir kitoms sistemoms, galbūt ateityje ji galėtų tapti struktūrizuotos užklausų kalbos atmaina skirta grafų duomenų bazių valdymo sistemoms. Taip pat šios sistemos buvo palygintos tarpusavyje. Buvo sukurta automatizuota testavimo sistema galinti pakrauti duomenis į visas keturias duomenų bazes ir patikrinti skirtingų užklausų vykdymo trukmę. Darbo metu pastebėta, kad viena iš duomenų bazių pastebimai atsilieka nuo kitų sistemų pagal duomenų įterpimo ir užklausų vykdymo trukmę. Atlikus sistemų analizę ir testavimą buvo išskirti kriterijai į kuriuos reiktų atkreipti dėmesį pasirenkant nereliacinę grafų duomenų bazių valdymo sistemą.

Ateityje galima būtų tobulinti automatizuota sistemą, ją pildant sudėtingesniais testais tai pat praplečiant testuoti galimų sistemų sąrašą. Galima būtų sistemoje įgyvendinti įvairias duomenų terpimo ir skaitymo simuliacijas lygiagrečiose sistemos seansuose. Taip patikrinant sistemos veikimo galimybes mėginant sukurti panašias į realias sąlygas, su kuriomis sistemai gali tekti susidurti.

Taip pat galima plėsti ir detalizuoti vertinimo kriterijus sukuriant vertinimo kriterijų medį kuriuo naudojantis būtų galima atsakant į klausimus apie būsimą sistemą einant nuo vieno punkto prie kito priklausomai nuo atsakymo pasirinkti tinkamiausią sistemą.

Literatūros sąrašas

[ARA] ArangoDB dokumentacija, versija 2.8.9, https://docs.arangodb.com.

[FAH+15] Donald Feinberg, Merv Adrian, Nick Heudecker, Adam M. Ronthal, Terilyn Palanca. Magic Quadrant for Operational Database Management Systems. https://www.gartner.com/doc/reprints?id=1-2PMFPEN&ct=151013, 2015

[ME92] Priti Mishra, Margaret H. Eich. Join processing in relational databases. ACM Computing Surveys, 1992.

[NEO] Neo4j dokumentacija, versija 3.0, http://neo4j.com/docs.

[ORI] OrientDB dokumentacija, versija 2.2.x, http://orientdb.com/docs.

[RN10] Marko A. Rodriguez, Peter Neubauer. The Graph Traversal Pattern, 2010.

[RWE13] Ian Robinson, Jim Webber, Emil Eifrem. Graph Databases. O'Reilly Media, Inc, Sebastopol, 2013.

[STA] Stanford Large Network Dataset Collection, http://snap.stanford.edu/data/#socnets.

[TIT] Titan dokumentacija, versija 1.0.0., http://s3.thinkaurelius.com/docs/titan/1.0.0/.