



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ TECHNOLOGIJŲ KATEDRA

Laura Stumbraitė

**DIDELIŲ DUOMENŲ KIEKIŲ VIENTISUMO TESTAVIMAS
„HADOOP“ IR „HP VERTICA“ DUOMENŲ BAZĖSE**

**BIG DATA INTEGRITY TESTING IN „HADOOP“
AND „HP VERTICA“ DATABASES**

Baigiamasis magistro darbas

Informacinių technologijų studijų programa, valstybinis kodas 621E14004

Duomenų gavybos technologijų specializacija

Informatikos inžinerijos studijų kryptis

Vilnius, 2016

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ TECHNOLOGIJŲ KATEDRA

TVIRTINU
Katedros vedėjas

Dmitrij Šešok
(Vardas, pavardė)

(Parašas)

(Data)

Laura Stumbraitė

**DIDELIŲ DUOMENŲ KIEKIŲ VIENTISUMO TESTAVIMAS
„HADOOP“ IR „HP VERTICA“ DUOMENŲ BAZĖSE**

**BIG DATA INTEGRITY TESTING IN „HADOOP“
AND „HP VERTICA“ DATABASES**

Baigiamasis magistro darbas

Informacinių technologijų studijų programa, valstybinis kodas 621E14004

Duomenų gavybos technologijų specializacija

Informatikos inžinerijos studijų kryptis

Vadovas

doc. dr. Saulius Valentinavičius

(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

(Parašas)

(Data)

Lietuvių kalbos konsultantė

dr. Vaida Buivydienė

(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

(Parašas)

(Data)

Vilnius, 2016

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Informacinių technologijų katedra

ISBN ISSN
Egz. sk.
Data-.....-....

Antrosios pakopos studijų **Informacinių technologijų** programos magistro baigiamasis darbas

Pavadinimas **Didelių duomenų kiekių vientisumo testavimas „Hadoop“ ir „HP Vertica“ duomenų bazėse**

Autorius **Laura Stumbraitė**

Vadovas **doc. dr. Saulius Valentinavičius**

Kalba: lietuvių

Anotacija

Magistro darbo tema – „Didelių duomenų kiekių vientisumo testavimas „Hadoop“ ir „HP Vertica“ duomenų bazėse“. Šiame darbe buvo nagrinėjami pagrindiniai „Hadoop“ ir „HP Vertica“ duomenų bazių skirtumai, atlikta kelių skirtingų duomenų testavimo algoritmų analizė.

Galutinis darbo rezultatas, sukurta programa „Duomenų revizorius“, kuri skirta duomenų testavimui „Hadoop“ ir „HP Vertica“ duomenų bazėse atlikti.

Darbą sudaro: įvadas, „Hadoop“ ir „HP Vertica“ duomenų bazių analizė, didelių duomenų kiekių testavimo algoritmų analizė, „QuerySurge“ ir „Duomenų revizorius“ testavimo įrankių apžvalga.

Darbo apimtis – 46 puslapiai, 20 paveikslų, 1 lentelė ir 28 informacijos šaltiniai.

Prasminiai žodžiai: dideli duomenų kiekiai, testavimas, testavimo algoritmas, „Hadoop“, „Vertica“, duomenų saugyklos.

Vilnius Gediminas Technical University
Faculty of Fundamental Sciences
Department of Information Technologies

ISBN ISSN
Copies No.
Date-.....-.....

Master Degree Studies **Information Technologies** study programme Master Graduation Thesis

Title **Big Data Integrity Testing in „Hadoop“ and „HP Vertica“ Databases**

Author **Laura Stumbraitė**

Academic supervisor **Assoc. Prof. Dr. Saulius Valentinavičius**

Thesis language: Lithuanian

Annotation

Theme of final master thesis – „Big Data Integrity Testing in “Hadoop” and “HP Vertica” Databases“. Main differences of “Hadoop” and “HP Vertica” databases, analysis of several different Big Data testing algorithms are presented in this master thesis.

The final result is “Duomenų revizorius” program which is created for Big Data integrity testing in “Hadoop” and “HP Vertica” databases.

Thesis consists of introduction, analysis of “Hadoop” and “HP Vertica” databases, analysis of different Big Data testing algorithms, review of “QuerySurge” and “Duomenų revizorius” testing tools.

Thesis consists of 46 pages, 20 pictures, 1 table and 28 information sources.

Keywords: Big Data, testing, testing algorithms, “Hadoop”, “Vertica”, data warehouses.

Vilniaus Gedimino technikos universiteto
egzaminų sesijų ir baigiamųjų darbų rengimo bei
gynimo organizavimo tvarkos aprašo 2011–2012
m. m.

2 priedas

(Baigiamojo darbo sąžiningumo deklaracijos forma)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Laura Stumbraitė 20140980

(Studento vardas ir pavardė, studento pažymėjimo Nr.)

Fundamentinių mokslų fakultetas

(Fakultetas)

Informacinės technologijos, DGTfm-14

(Studijų programa, akademinė grupė)

BAIGIAMOJO DARBO (PROJEKTO)

SĄŽININGUMO DEKLARACIJA

2016 m. gegužės 30 d.

Patvirtinu, kad mano baigiamasis darbas tema „Didelių duomenų kiekių testavimas tarp „Hadoop“ ir „Vertica“ duomenų bazių“ patvirtintas 2014 m. lapkričio 11 d. dekanų potvarkiu Nr. 326fm, yra savarankiškai parašytas. Šiame darbe (projekte) pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Mano darbo vadovas doc. dr. Saulius Valentinavičius.

Kitų asmenų indėlio į parengtą baigiamąjį darbą (projektą) nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).

(Parašas)

Laura Stumbraitė

(Vardas ir pavardė)

Turinys

Įvadas	10
<i>Darbo tikslas</i>	11
<i>Problema</i>	11
<i>Uždaviniai tikslui pasiekti</i>	11
<i>Darbo temos aktualumas</i>	12
1. Dideli duomenų kiekiai	13
1.1. <i>Apimtis</i>	13
1.2. <i>Sparta</i>	14
1.3. <i>Įvairovė</i>	14
2. Technologijų pasirinkimas	15
2.1. <i>„Hadoop“ ekosistema (technologija)</i>	16
2.1.1. Užklausų vykdymas „Hadoop“ duomenų bazėje	17
2.2. <i>„HP Vertica“ duomenų saugykla</i>	19
3. „Hadoop“ ir „HP Vertica“ technologijų naudojimas kartu	20
3.1. <i>„Hadoop“ ir „HP Vertica“ duomenų bazių palyginimas</i>	21
4. Didelių duomenų kiekių testavimas	24
4.1. <i>Duomenų vientisumo svarba duomenų bazėse</i>	25
4.2. <i>Duomenų perkėlimo (migravimo) testavimo algoritmas</i>	25
4.3. <i>Duomenų vientisumo testavimas automatinio būdu</i>	27
4.4. <i>Kiti duomenų vientisumo testavimo metodai</i>	27
4.4.1. Užklausų atimties testavimo metodas	27
4.4.2. Duomenų atrinkimo testavimo metodas	29
5. „QuerySurge“ – įrankis didelių duomenų kiekių testavimui	31
5.1. <i>Testavimo programos plėtinys „Query Wizards“</i>	31
6. Duomenų vientisumo testavimo programa „Duomenų revizorius“	33
6.1. <i>Skirtingų duomenų bazių užklausų palyginimas</i>	37
6.2. <i>Įrankių, skirtų tarpiniams rezultatams išsaugoti, apžvalga</i>	39
6.3. <i>„Duomenų revizoriaus“ ir „QuerySurge“ įrankių palyginimas</i>	41
Išvados	43
Literatūros sąrašas:	45

Paveikslų sąrašas

1 pav. „NoSQL“ ir reliacinės duomenų bazių galimybių palyginimo grafikas	15
2 pav. „Hadoop“ ekosistemos veikimo principas	17
3 pav. „Hadoop“ ir „HP Vertica“ technologijų veikimas kartu.....	21
4 pav. 1 TB duomenų užkrovimo laikas	23
5 pav. Užklauskos vykdymo laikas	23
6 pav. Dviejų duomenų lentelių pavyzdys.....	28
7 pav. Užklausa, skirta iš pirmosios lentelės „dalyviai_1“ atimti antrąją „dalyviai_2“	28
8 pav. Gauti rezultatai, įvykdžius pirmąją užklauską	28
9 pav. Užklausa, skirta iš antrosios lentelės „dalyviai_2“ atimti pirmąją „dalyviai_1“	29
10 pav. Gauti rezultatai, įvykdžius antrąją užklauską	29
11 pav. Užklauskų, skirtų duomenims iš „Hadoop“ ir „HP Vertica“ duomenų bazių gauti, išsaugojimo testas	33
12 pav. Įrašyta WHERE sąlyga formuojant „Hadoop“ užklauską	34
13 pav. Įrašyta WHERE sąlyga formuojant „HP Vertica“ užklauską	34
14 pav. Išsaugotų užklauskų paieška naudojant „Robomongo“ įrankį.....	35
15 pav. Atfiltruotų užklauskų rezultatų pavyzdys naudojantis „Robomongo“ įrankiu, kai duomenų palyginimo statusas yra „teisingas“	36
16 pav. Atfiltruotų užklauskų rezultatų pavyzdys naudojantis „Robomongo“ įrankiu, kai duomenų palyginimo statuso reikšmė yra „klaidinga“	37
17 pav. Užklauskų, paruoštų skirtingoms duomenų bazėms, palyginimo pavyzdys.....	38
18 pav. Gauti rezultatai tikrinant duomenis dienos tikslumu	38
19 pav. Gauti rezultatai tikrinant duomenis milisekundžių tikslumu	39
20 pav. „QuerySurge“ įrankio vartotojo sąsajos pavyzdys.....	40

Santrumpų sąrašas

ETL – duomenų gavimas, transformacija ir įkėlimas (angl. *Extract, Transform, Load*)

HDFS – „Hadoop“ paskirstytųjų failų sistema (angl. *Hadoop Distributed File System*)

HQL – „Hive“ užklausų rašymo kalba (angl. *Hive Query Language*)

JDBC – JAVA duomenų bazių ryšiai (angl. *JAVA Database Connectivity*)

MPP – lygiagrečiai vykdomas didelio masto duomenų apdorojimas (angl. *Massively Parallel Processing*)

ODBC – atviri duomenų bazių ryšiai (angl. *Open Database Connectivity*)

SQL – struktūrizuota užklausų rašymo kalba (angl. *Structured Query Language*)

UDF – vartotojo apibrėžtos funkcijos (angl. *User Defined Functions*)

Išvadas

Vienas iš didžiausių iššūkių šiuolaikinėms informacinių technologijų organizacijoms yra didelių duomenų kiekių apdorojimas. Reikia priimti kas dieną generuojamus duomenis ir juos tinkamai suagreguoti, pasirūpinti, kad nekiltų problemų dėl sparčiai augančio duomenų kiekio saugojimo, užtikrinti, kad priimti duomenys yra teisingi – nėra susidubliavusių ar dingusių duomenų, jie yra tikslūs ir sutampantys. Didžiulių duomenų kiekių kokybės užtikrinimas yra didžiulis uždavinys organizacijų testuotojams. Duomenų testavimas yra labai svarbi ir privaloma programinės įrangos plėtros dalis. Labai svarbus ne tik sugebėjimas įvertinti duomenų kokybę bei tikslumą, bet taip pat ir netiesioginis jos pagerinimas nustatius defektus ir problemas. Testavimo trūkumas gali turėti daug įtakos duomenų nesutapimams, trukumams, susidubliavimo atsiradimui duomenų bazėse, taip pat gali lemti socialinių problemų ir finansinių nuostolių atsiradimą.

Nepaisant visų įdedamų pastangų siekiant pagerinti didelių duomenų kiekių kokybę ir tikslumą, testavimo veiksmingumas išlieka daug prastesnis už lūkesčius. Jis dažniausiai kainuoja labai brangiai, tačiau yra nenusipėjamas. Tam, kad testavimo veiksmingumas būtų maksimaliai pasiektas, reikia atlikti absoliučiai visų duomenų patikrą [1]. Tačiau norint ištestuoti visus duomenis susiduriama su tokiomis problemomis, kaip laiko ir resursų trūkumas, kuris gali apriboti testuotojų pastangas efektyviai užbaigti testavimą, todėl labai išsamaus testavimo tenka atsisakyti, tikrinant tik pačius svarbiausius ir aktualiausius duomenis ar jų bendrą statistiką. Reikia pasirinkti pačius tinkamiausius ir veiksmingiausius duomenų testavimo būdus, kurie kuo labiau padidintų testavimo efektyvumą ir kuo tikslingiau išnaudotų turimus ribotus resursus, skirtus didelių duomenų kiekių testavimui atlikti.

Automatinis duomenų testavimas galėtų supaprastinti ir pagreitinti testavimo procesą lyginant su duomenų testavimu rankiniu būdu. Automatinių testų vykdymo tikslas yra kiek įmanoma sumažinti žmogaus fizinio darbo (testavimo) kiekį, kurį testuotojas atlieka duomenų tikslumo tikrinimo metu, ir ištestuoti kiek įmanoma daugiau duomenų per tą patį laiką.

Nemokamų įrankių, skirtų didelių duomenų kiekių vientisumo tikrinimui „Hadoop“ ir „HP Vertica“ duomenų bazėse, nėra. Taigi, šio darbo pagrindinis tikslas yra išsiaiškinti pagrindinius „Hadoop“ ir „HP Vertica“ duomenų bazių skirtumus bei rasti didelių duomenų kiekių kokybės užtikrinimo sprendimą jose esantiems duomenims.

Darbo tikslas

Pagrindinis šio darbo tikslas – rasti sprendimą didelių duomenų kiekiams palyginti „Hadoop“ ir „HP Vertica“ duomenų bazėse. Sprendimas turėtų tenkinti šiuos reikalavimus:

- Galimybė testuoti istorinius duomenis (kelių ar keliolikos metų senumo).
- Galimybė testuoti kasdien gaunamus naujus duomenis (vėliausiai už praėjusią dieną).
- Panaudoti trečios šalies įrankį, kuriame bus laikomi iš „Hadoop“ ir „HP Vertica“ duomenų bazių gauti užklausų rezultatai, kad vėliau būtų galima atlikti gautų rezultatų palyginimą.

Taip pat išsiaiškinti pagrindinius „Hadoop“ ir „HP Vertica“ duomenų bazių skirtumus. Apibendrinti sukurtą sprendimą ir įvertinti jo pagrindinius privalumus ir trūkumus, naudą ir patikimumą testuojant didelius duomenų kiekius „Hadoop“ ir „HP Vertica“ duomenų bazėse.

Problema

Surasti geriausią sprendimą dideliems duomenų kiekiams testuoti „Hadoop“ ir „HP Vertica“ duomenų bazėse, kad būtų pasiektas maksimalus duomenų kokybės ir tikslumo užtikrinimo lygis ir būtų kuo mažesnis žmogiškųjų resursų poreikis šiam darbui atlikti.

Uždaviniai tikslui pasiekti

- Išsiaiškinti, kokios specifinės problemos gali kilti naudojantis „Hadoop“ ir „HP Vertica“ duomenų bazes, taip pat pagrindinius šių duomenų bazių skirtumus.
- Sugalvoti sprendimą dideliame duomenų kiekiui palyginti panaudojant trečios šalies įrankį testavimo rezultatams laikyti.
- Palyginti įgyvendintą sprendimą su komerciniu įrankiu „QuerySurge“, skirtu didelių duomenų kiekių kokybei užtikrinti.
- Pritaikyti, kad sukurtas įrankis veiktų testuojant tiek istorinius duomenis, tiek kasdien gaunamus naujus.
- Įvertinti sukurto sprendimo privalumus ir trūkumus, gautą naudą ir patikimumą testuojant didelius duomenų kiekius duomenų bazėse.

Darbo temos aktualumas

Kuo daugiau naujų klientų atsiranda ir įmonė labiau plečiasi, tuo greičiau duomenų kiekiai auga ir kaupiasi. Kuo daugiau duomenų duomenų bazėse, tuo lėčiau vykdomos užklausos. Vis didesnė ir didesnė našta gula ant testuotojų pečių, reikia užtikrinti pačią aukščiausią duomenų tikslumą ir kokybę. Didžiuliams duomenų kiekiams patikrinti žmogiškųjų resursų nepakanka, todėl į pagalbą pasitelkiami automatiniai įrankiai sprendimai. Duomenų vientisumo testavimo metu tos pačios užklausos abiejuose duomenų bazėse („Hadoop“ ir „HP Vertica“) yra vykdomos labai daug kartų. Pritaikius automatinio testavimo sprendimą tiek „Hadoop“ tiek „HP Vertica“ duomenų bazėms, didelių duomenų kiekių testavimo efektyvumas padidėtų, atsirastų galimybė ištestuoti visus duomenis, o ne tik tam tikrą apibendrintą duomenų dalį. Tokiu būdu pats duomenų testavimas taptų daug vertingesnis, taip pat būtų sutaupomi laiko ir žmogiškieji resursai. Automatinio duomenų testavimo sprendimo sukūrimas kainuoja taip pat ne pigiai, tačiau ilgalaikėje perspektyvoje tai padeda labai stipriai optimizuoti darbo laiką ir užtikrinti daug aukštesnį duomenų tikslumą ir kokybę, taip pat ir daug platesnį testavimo mastą. Sukūrus tinkamą automatinį sprendimą dideliems duomenų kiekiams testuoti, jis gali būti pritaikytas UAB „Adform Lithuania“ duomenų kokybės užtikrinimui gerinti.

1. Dideli duomenų kiekiai

Griežtas didelių duomenų kiekių apibrėžimas neegzistuoja. Pirmiausia atsirado idėja, kad informacijos apimtys auga taip greitai, jog norint išanalizuoti tokį kiekį duomenų nebeužtenka kompiuterių atminties, kurie yra naudojami duomenų apdorojimui atlikti. [2] Dideli duomenų kiekiai nėra atskira technologija, technika ar iniciatyva. Greičiau, tai yra daugelio verslo ir technologijų sričių tendencija. Dideli duomenų kiekiai remiasi technologijomis ir iniciatyvomis, kuriuose dalyvaujantys duomenys yra per daug įvairūs, greitai besikeičiantys arba per daug masyvūs įprastoms technologijoms, įgūdžiams ir infrastruktūroms, kad būtų galima perduoti juos efektyviai. Kitais žodžiais tariant, duomenų apimtys, greitis ir įvairovė yra per dideli. Tačiau naujos technologijos suteikia galimybę suprasti didelių duomenų kiekių reikšmę. Didelius duomenų kiekius nuolatos generuoja viskas, kas yra aplink mus. Kiekvienas skaitmeninis procesas ir socialinės žiniasklaidos keitimasis gamina didžiulius duomenų srautus. Sistemos, jutikliai ir mobilieji įrenginiai perduoda juos. Dideli duomenų kiekiai gaunami iš daugybės skirtingų šaltinių be galo greitai, labai dideliais kiekiais ir įvairove. Pavyzdžiui, mažmenininkai gali sekti vartotojų paspaudimus jo internetiniame puslapyje ir tokiu būdu identifikuoti vartotojų elgsenos tendencijas, dėl kurių galima patobulinti kampanijas, kainų nustatymą ir duomenų kaupimą. Komunalinės paslaugos gali užfiksuoti, koks namų ūkio energijos naudojimo lygis – taip galima numatyti prastovas ir paskatinti efektyvesnę energijos vartojimą. Vyriausybės ir netgi „Google“ gali aptikti ir sekti ligos protrūkių atsiradimą per socialinės žiniasklaidos signalus. Naftos ir dujų bendrovės gali imtis jutiklių produkcijos savo gręžimo įrangoje, siekiant priimti daug efektyvesnius ir saugesnius gręžimo sprendimus. Didelių duomenų kiekių rinkiniai yra labai dideli ir sudėtingi, todėl labai nepraktiška bandyti juos apdoroti naudojant tradicinius programinės įrangos įrankius. [3] Apibendrinant, dideli duomenų kiekiai yra susiję su duomenų kūrimu, laikymu, surinkimu ir puikia duomenų greičio, apimties bei įvairovės analize.

1.1. Apimtis

Tipinis kompiuteris 2000-aisiais metais turėjo maždaug apie 10 Gb atminties. Šiomis dienomis vien socialinis tinklapis „Facebook“ kasdien naujiems duomenims sunaudoja apie 500 Tb; „Boingas 737“ sugeneruoja 240 Tb skrydžio duomenų vieno skrydžio per JAV metu; išmaniųjų telefonų paplitimas, duomenys, kuriuos jie kuria ir sunaudoja; jutikliai, kurie yra integruoti į kasdieninius objektus kuria milijardus naujų, nuolat atnaujinamų duomenų kanalų, kurių sudėtyje yra aplinka, vieta ir kita informacija įskaitant ir filmuotą medžiagą.

1.2. Sparta

Daugumai programų duomenų gavimo greitis yra daug svarbesnis nei apimtis. Informacija gauta realiu laiku arba realiu laiku su keletu sekundžių paklaida leidžia kompanijoms dirbti daug lanksčiau ir greičiau. [4] Paspaudimų srautai ir skelbimų (reklamų) parodymai (angl. *impressions*) užfiksuoja milijonus vartotojo elgsenos įvykių per sekundę; aukšto dažnio vertybinių popierių prekybos algoritmai atspindi rinkos pokyčius mikrosekundžių tikslumu; iš vienos mašinos į kitą yra vykdomi duomenų mainai tarp milijardų įrenginių; infrastruktūra ir jutikliai skleidžia didžiulius kiekius žurnalo duomenų (angl. *log data*) realiu laiku; tiesioginių interneto (angl. *on-line*) žaidimų sistemos palaiko milijonus vartotojų vienu metu, kiekvienam atliekant daug ėjimų per sekundę.

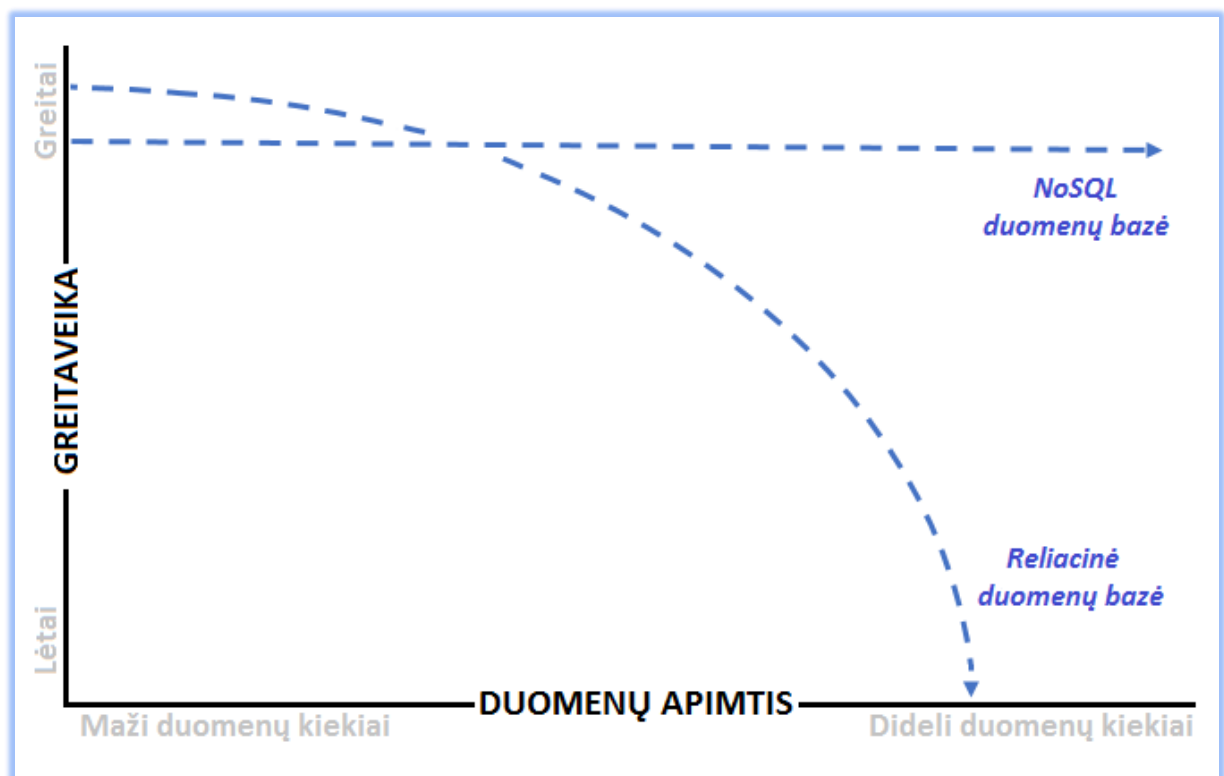
1.3. Įvairovė

Didžiuliai duomenų kiekiai – tai nėra vien tik skaičiai, datos ar sekos. Didžiuliai duomenų kiekiai yra taip pat ir geografinė bei erdvinė informacija, 3D duomenys, garso ir vaizdo medžiaga, nenuoseklus (nerišlus) tekstas įskaitant ir žurnalų failus (angl. *log files*) ir socialinę žiniasklaidą. Tradicinės duomenų bazių sistemos buvo skirtos dirbti su mažesnių apimčių struktūrizuotais duomenimis, lėčiau atsinaujinančiomis arba nuspėjamomis, nuosekliomis duomenų struktūromis. Tradicinės duomenų bazių sistemos taip pat yra skirtos veikti viename serveryje, todėl talpos padidinimas yra brangus ir ribotas. Plėtojantis programoms pradėjo didėti ir jų vartotojų kiekis, todėl programų kūrimo praktikos pradėjo tapti vis judresnės, tradicinis reliacinės duomenų bazės naudojimas tapo daugiau atsakomybe daugeliui įmonių nei galimybės suteikimo veiksmu jų versle. Didelių duomenų kiekių duomenų bazės išsprendžia pastarąsias problemas ir suteikia įmonėms įrankius, kurie gali sukurti milžinišką verslo vertę. [5]

2. Technologijų pasirinkimas

Darbai su dideliais duomenų kiekiais yra naudojamos dviejų klasių technologijos: sistemos, kurios pateikia operacines galimybes realiuoju laiku, interaktyvūs darbo krūviai, kur duomenys pirmiausia yra surenkami ir saugomi; sistemos, kurios pateikia analitines galimybes atgaline data, sudėtingas analizes, kurios apima daugumą arba visus duomenis. Šios technologijų klasės yra viena kitą papildančios ir dažnai suderinamos kartu.

Operacinės ir analitinės apkrovos dideliems duomenų kiekiams turi skirtingus reikalavimus ir sistemų vystymąsi siekiant išspręsti savo konkrečius poreikius atskirai ir labai skirtingais būdais. Kiekvienas iš jų paskatino naujų technologijų architektūros kūrimą. Operacinės sistemos, tokios kaip „NoSQL“ duomenų bazės, gali apdoroti ir labai skirtingas užklaudas ir tuo pačiu metu sumažinti vėlavimą vykdant tas užklaudas su daug kriterijų. Kita vertus, analitinės sistemos yra labiau akcentuojamos dėl didelio našumo, didelių duomenų srautų apdorojimo; vykdomos užklaudos gali būti labai sudėtingos ir naudoti daugumą arba netgi visus duomenis, esančius sistemoje, bet kuriuo metu. Abi sistemos vienu metu dirba su daug serverių, veikiančių sugrupuotose klasteriuose, ir valdyti dešimtis ar šimtus terabaitų duomenų iš milijardų įrašų. [6]



1 pav. „NoSQL“ ir reliacinės duomenų bazių galimybių palyginimo grafikas [7]

2.1. „Hadoop“ ekosistema (technologija)

„Hadoop“ technologija – tai yra bendrinė duomenų apdorojimo sistema, kuri skirta vykdyti užklausoms ir taip pat dideliems duomenų paketams skaityti, kurie gali būti labai dideli ir užimti dešimtis arba šimtus terabaitų arba netgi petabaitų. Duomenys yra įkeliami arba pridedami prie „Hadoop“ duomenų paskirstymo sistemos (angl. *Hadoop Distributed File System - HDFS*). [8] Tada „Hadoop“ technologija nuskaitys visus ten esančius duomenis, o gauti rezultatai yra iškeliami į kitus failus. Tai dar negali būti laikoma duomenų baze, kol nėra atliekami jokie duomenų atnaujinimai ar operacinis duomenų apdorojimas. „Hadoop“ technologija taip pat nepalaiko tokių pagrindinių funkcijų, kaip indeksavimas arba SQL sąsaja, tačiau yra papildomų atviro kodo projektų, kurie leidžia pridėti šias galimybes.

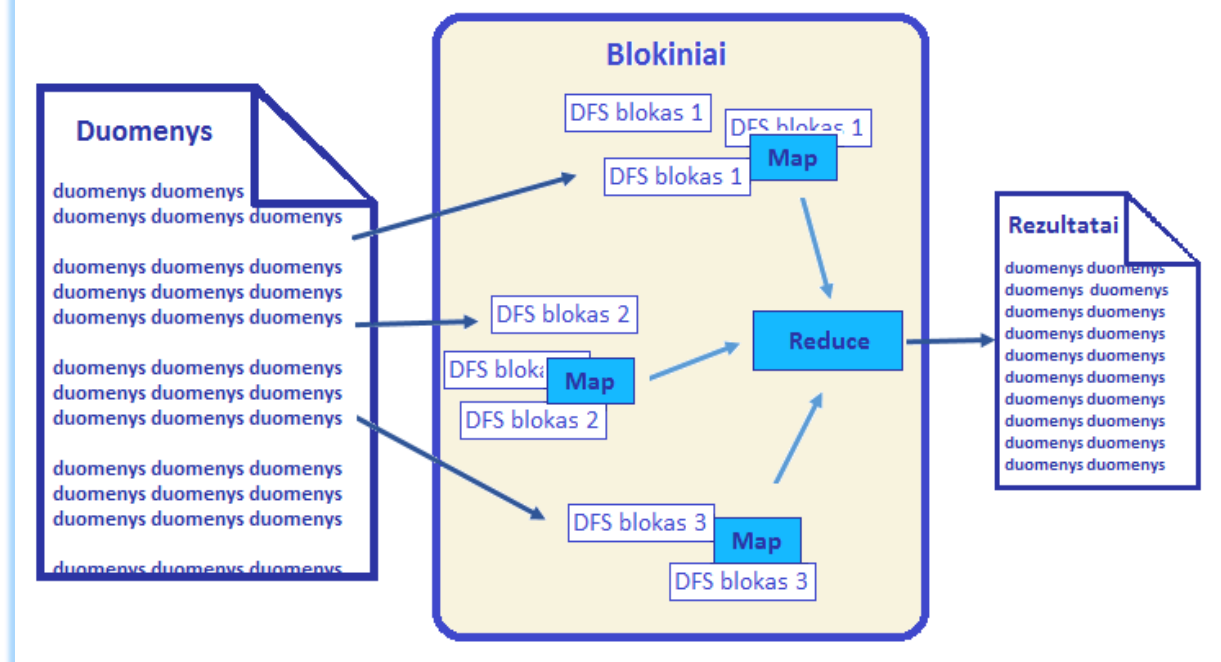
Naudojantis „Hadoop“ technologiją galima operuoti masyviais duomenų rinkiniais pagal horizontalų duomenų apimties paskirstymą, tada duomenų apdorojimas vyksta daugelyje serverių vienu metu pagal metodą vadinamą „MapReduce“. Vertikalus duomenų apimties paskirstymas (pavyzdžiui, kai duomenų apdorojimas vyksta viename galingame serveryje) taip pat yra galimas, tačiau toks metodas yra labai brangus ir ribotas. Dabartiniu laikotarpiu arba artimiausioje ateityje nėra ir nebus vieno prieinamo serverio, kuris turėtų reikalingą pajėgumą ir įgaliojimus, kad galėtų apdoroti tokius didelius kiekius duomenų tinkamu laiku.

Šimtai ar netgi tūkstančiai mažų, nebrangių, produktyvių serverių yra labai galingi ir pajėgūs apdoroti didžiulius duomenų kiekius, jeigu duomenų operacijos yra atliekamos taikant horizontalų duomenų apimties paskirstymą ir vykdant duomenų apdorojimą lygiagrečiai. Naudojant „MapReduce“ metodą, „Hadoop“ technologija padalina sudėtingas duomenų operacijas į mažesnes ir siunčia jas skirtingiems serveriams, tada kiekvienas serveris išsprendžia savo gautas mažas operacijas lygiagrečiai. Vėliau išspręstos mažesnės operacijos yra apjungiamos kartu ir gautas sprendimas įrašomas į failą, kurie savo ruožtu vėliau gali būti naudojami kaip įvesties duomenys sekančio/papildomo „MapReduce“ metodo vykdymo metu.

„Hadoop“ technologija buvo ypač naudinga aplinkose, kur duomenų rinkimui naudojami masiniai serverių ūkiai. Ši technologija gali apdoroti tiek daug lygiagrečiai vykdomų užklausų, kiek foninėms partijų užduotims būti vykdomoms užtenka vietos tame pačiame serverių ūkyje. Tai apsaugo vartotojus nuo būtinybės įsigyti naujos papildomos kompiuterinės įrangos duomenų bazių sistemai, kad būtų galima apdoroti duomenis. Svarbiausia, kad pačiam vartotojui taip pat nereikia gaišti laiko perkėlinėjant duomenis į kitą sistemą. [9]



Apžvalga



2 pav. „Hadoop“ ekosistemos veikimo principas [10]

2.1.1. Užklausų vykdymas „Hadoop“ duomenų bazėje

Užklausų vykdymui ir darbui su duomenimis esančiais „Hadoop“ duomenų bazėje yra naudojama „Hive“ sąsaja. „Hive“ yra duomenų saugojimo infrastruktūra paremta „Hadoop“ technologija. Ji palaiko SQL dialektą, todėl SQL programuotojai gali rašyti užklausas naudodami „Hive“ užklausų kalbos formuluotes (angl. *Hive Query Language - HQL*), kurios yra labai panašios į standartines SQL užklausų formuluotes. HQL kalba palaiko ribotą kiekį komandų, tačiau vistiek yra gana naudinga. HQL formuluotės yra suskirstomos pagal „Hive“ serviso veikimą į „MapReduce“ darbus ir vykdoma visame „Hadoop“ klasteryje.

„Hive“ užklausos gali būti paleidžiamos vykdytis keletu skirtingų būdų. Jos gali būti leidžiamos per komandinės eilutės sąsają, gali būti leidžiamos naudojant JAVA duomenų bazių ryšius (angl. *JAVA Database Connectivity - JDBC*) arba atvirus duomenų bazių ryšius (angl. *Open Database Connectivity - ODBC*) susietus su „Hive“ JDBC/ODBC draiveriais, taip pat per „Hive Thrift Client“. „Hive Thrift Client“ yra duomenų bazės klientas, kuris turi suinstaliuotą kliento kompiuteryje (arba viduriniame lygyje, jeigu kalbėtume apie trijų lygių architektūrą). Šis „Hive“

klientas palaiko ryšį su serveryje veikiančiais „Hive“ servisais. „Hive Thrift Client“ gali būti naudojamas aplikacijų parašytų programavimo kalbomis C#, JAVA, PHP, Python, Ruby. [11]

„Hive“ atrodo labai panašiai kaip tradicinis duomenų bazės kodas su SQL prieiga. Tačiau, yra keletas pagrindinių skirtumų dėl to, kad „Hive“ duomenų bazė yra pritaikyta „Hadoop“ technologijai ir „MapReduce“ operacijų vykdymui. Pirmasis skirtumas yra ta, kad „Hadoop“ skirtas ilgam ir nuosekliam duomenų nuskaitymui, o kadangi „Hive“ yra pagrįstas „Hadoop“ technologija, galima tikėtis, kad užklauskos turėtų būti vykdomas žymiai ilgiau. Tai reiškia, kad „Hive“ duomenų bazė netiktų užklauskų vykdymui, kurių rezultatus reikia gauti taip greitai, kaip būtų galima tikėtis naudojant SQL duomenų bazę. Galiausiai, „Hive“ duomenų bazė yra paremta duomenų nuskaitymu ir todėl netinka tranzakcijų apdorojimui, kuris užima labai daug laiko įrašymo operacijų įvykdymui.

„Hive“ duomenų bazė yra suprantama kaip tiesiog katalogas arba lentelės su pavadinimais. Tačiau didžiuliams klasteriams, kuriais naudojami keletas skirtingų komandų ir vartotojų, tai yra labai naudinga todėl, kad tokiu būdu yra išvengiama pasikartojančių lentelių pavadinimų. Taip pat įprastas būdas naudoti duomenų bazes produkcijos lentelių suskirstymui į logines grupes.

„Hive“ turi suskirstytų lentelių sampratą. Vėliau įsitikinsime, kad jos duoda didelę naudą užklauskų vykdymo greitaveikai ir tai gali padėti norint suskirstyti duomenis kokia nors logiška tvarka, pavyzdžiui hierarchiškai.

Turbūt svarbiausia duomenų pasiskirstymo priežastis yra greičiau vykdomos užklauskos. Dažnai nereikia nuskaityti visų katalogų turinių, kad gauti reikalingus duomenis, tiesiog užklauskos sąlygoje reikia nurodyti tikslų žinomą katalogą ar keletą jų. Net jei mes turime tūkstančius katalogų, visi jie, išskyrus vieną, gali būti ignoruojami. Dėl labai didelių duomenų rinkinių, duomenų suskirstymas gali žymiai pagerinti užklauskų veikimą, bet tik tais atvejais, kai suskirstymo schema yra atvaizduojama pagal bendrų rėžių filtravimą (pavyzdžiui, vietą ar laiko žymos intervalus). Tačiau užklauskos vykdymas per visus suskirstymus gali pradėti vykdyti labai daug „MapReduce“ funkcijų, jei duomenų kiekis esantis lentelėse ir suskirstymų skaičius yra labai dideli. [12]

2.2. „HP Vertica“ duomenų saugykla

„HP Vertica“ buvo sugalvota legendinio duomenų guru Michael Stonebraker. „HP Vertica“ analitinė (angl. *Vertica Analytics*) platforma buvo planuojama kurti analitiniais didelių duomenų kiekiams (angl. *Big Data Analytics*) nuo pat pirmos programinio kodo eilutės. Kodėl buvo nuspręsta kurti viską nuo nulio? Taip daryti buvo nuspręsta todėl, kad buvo aišku, jog duomenų saugyklos ir įprasto verslo praktikos apriboja technologijas ir dėl to įmonės priima skausmingus sprendimus.

„HP Vertica“ analitinė platforma buvo sąmoningai sukurta taip, kad jos branduolį sudarytų greitis, mastelis, paprastumas, aiškumas, o architektūriškai paruošta taip, kad galėtų dirbti su analitiniais darbo krūviais. „HP Vertica“ technologija suteikia labai didelį užklausų vykdymo greitį (užklausos vykdomos 50 – 1000 kartų greičiau), petabaitai atminties (kiekviename serveryje gali laikyti 10 – 30 kartų daugiau duomenų).

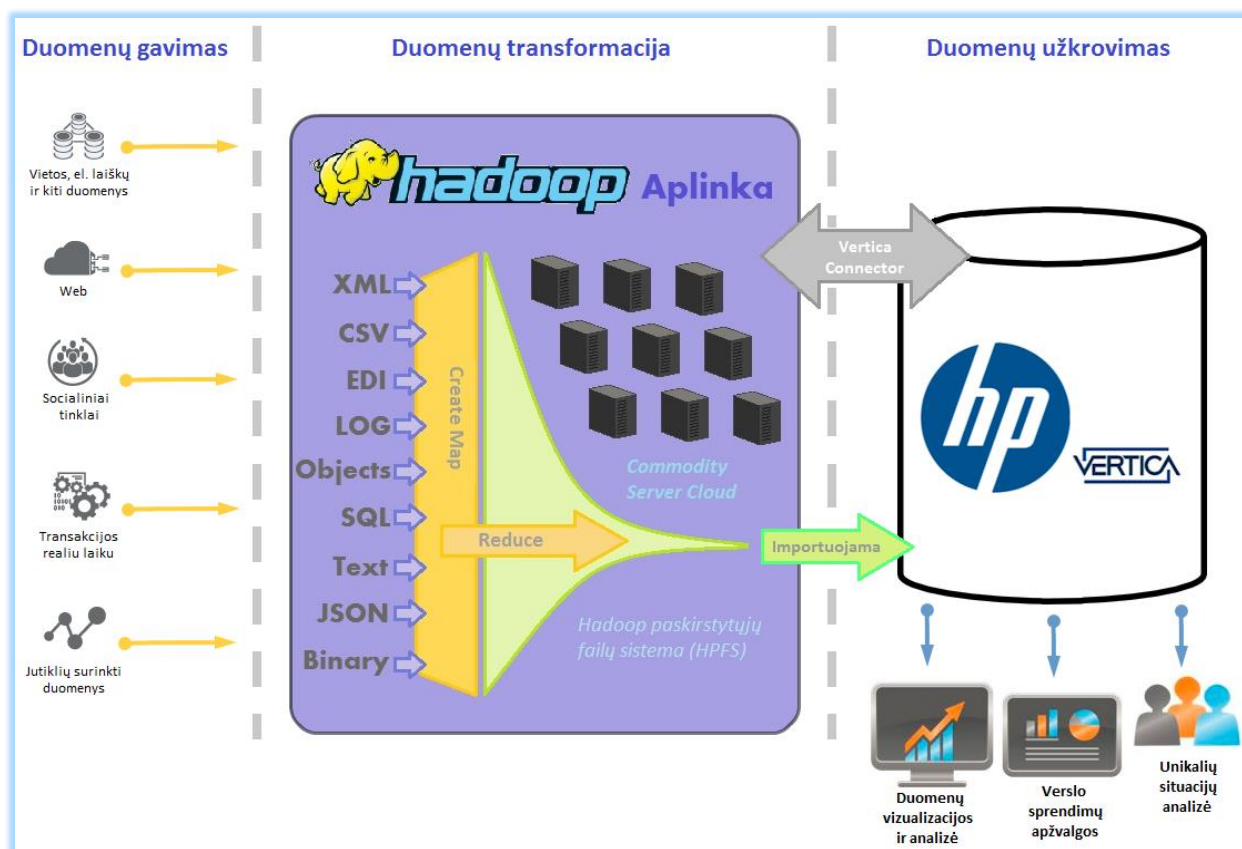
„HP Vertica“ yra grindžiamas paremta standartinėmis, reliacinėmis duomenų bazėmis. Todėl palaiko populiariąją užklausų kalbą SQL, JDBC / ODBC. Tai leidžia vartotojams sutaupyti daug laiko, skirtą šių technologijų investicijoms ir mokymams, nes visos dažniausiai naudojami SQL programavimo įrankiai ir kalbos veikia sklandžiai.

„HP Vertica“ yra optimizuotas didelio masto analitinis įrankis. Jis yra unikalios suprojektuotas naudojant atminties ir disko subalansuotą suspaustų stulpelių paskirstymo pavyzdį, kuris leidžia eksponentiškai greičiau dirbti su šiuolaikiniais duomenų analizės darbo krūviais, nei senesni metodai ir technikos.

„Vertica“ yra platforma, kuri masiškai lygiagrečiai vykdo apdorojimą (angl. *Massively Parallel Processing - MPP*) ir paskirsto savo darbo krūvį per keletą produktų serverių. „HP Vertica“ leidžia vartotojams pridėti UDF plėtinius (UDx), vartotojų parašyta kodą, kuris veikia lygiagrečiai „Vertica“ viduje. Šis vartotojų parašytas kodas gali būti parašytas C, C++ ar JAVA programavimo kalbomis. Tai leidžia pasiekti visus šiuos plėtinius naudojant SQL kalbą ir visus esamus SQL pagrindu padarytus įrankius. [13]

3. „Hadoop“ ir „HP Vertica“ technologijų naudojimas kartu

„HP Vertica“ ir „Hadoop“ kartu suteikia greičiausią ir pilnai išvystytą būdą vykdyti SQL užklausas naudojantis „Hadoop“ duomenų saugykloje esančiais duomenimis. Naudotojai gali daryti analizes ir duomenų tyrimus nepaisydami nei duomenų formato nei „Hadoop“ duomenų pasiskirstymo sistemos naudojimo. [14] Klientai gali patys analizuoti savo duomenis realiu laiku, kai tik jie atsiduria „Hadoop“ klasteryje. 3 pav. esančioje schemoje pavaizduotos 3 pagrindinės didelių duomenų kiekių apdorojimo algoritmo dalys. Pirmoji dalis – duomenų gavimas. Duomenys gali būti gaunami iš įvairiausių duomenų šaltinių: tai gali būti žemėlapių koordinatės, elektroninių laiškų informacija, internetinės reklamos, socialinių tinklų duomenys, realaus laiko duomenų tranzakcijos, taip gali būti renkami jutiklių perduodami duomenys, bei daugybė kitų duomenų šaltinių. Surinkti Raw duomenys gali būti išsaugomi įvairiausiais formatais: XML, CSV, LOG, *Objects*, SQL, *Text*, JSON, *Binary*. [15] Gauti duomenys yra įkeliami į „Hadoop“ paskirstytųjų failų sistemą, transformuojami, paskirstomi į atitinkamas lenteles, kurios yra sukurtos pagal skirtingus verslo poreikius. Vėliau naudojant MapReduce funkciją jie yra suimportuojami į „HP Vertica“ duomenų bazę. Duomenų paieška bei analizė vykdant SQL užklausas gali būti vykdoma ir „Hadoop“ duomenų bazėje, tačiau reikia naudotis papildomais įrankiais, pavyzdžiui „Hive“, o pats užklausų vykdymo laikas yra gana ilgas, kartais galintis užtrukti netgi iki keleto valandų. „HP Vertica“ duomenų bazėje visi duomenys suskirstyti stulpeliais. Vykiant SQL užklausas šioje duomenų bazėje duomenų gavimas gali užtrukti nuo kelių sekundžių iki keleto minučių, priklausomai nuo duomenų apimtys. Dėl greito veikimo „HP Vertica“ duomenų bazę pravartu naudoti duomenų atvaizdavimui vartotojo sąsajoje. Šia duomenų baze taip pat patogiau naudotis atliekant duomenų analizę, kuriant duomenų vizualizacijas, norint padaryti verslo sprendimų apžvalgą ar išanalizuoti duomenis už atitinkamą laikotarpį. [16]



3 pav. „Hadoop“ ir „HP Vertica“ technologijų veikimas kartu [17]

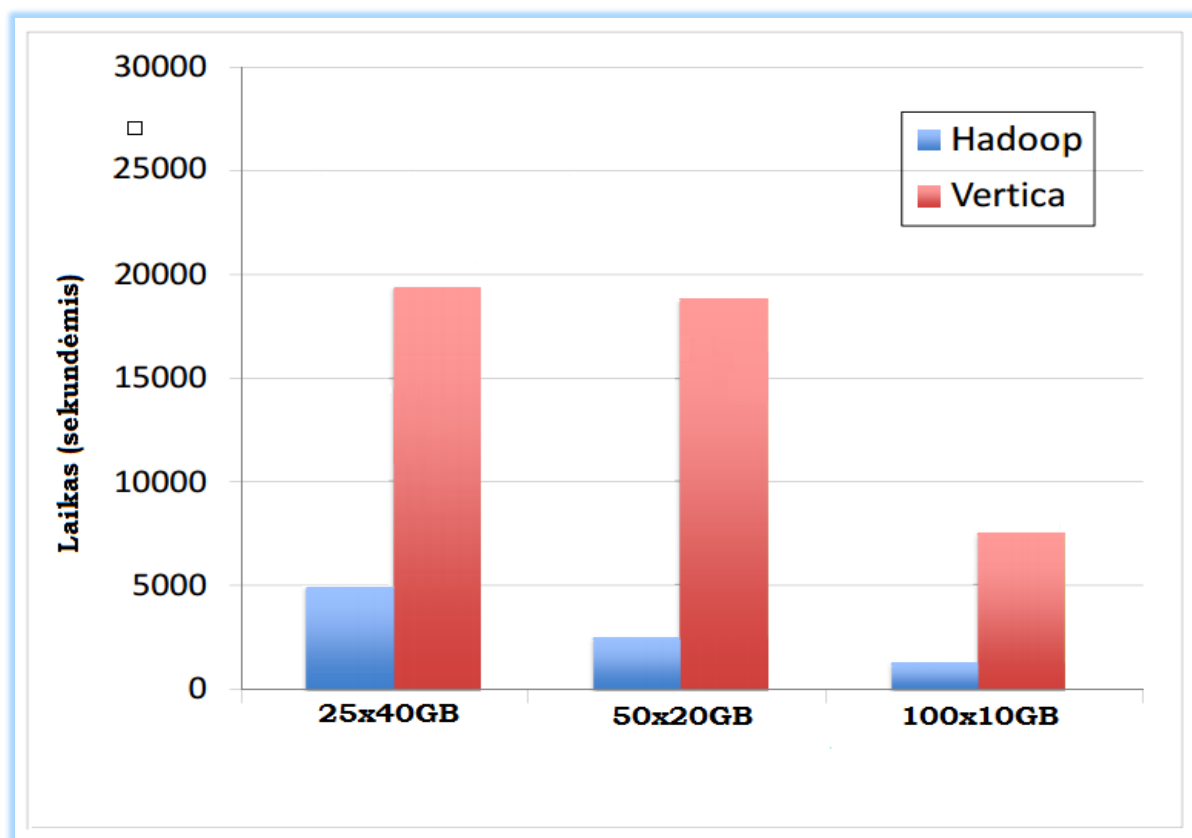
Labai svarbu, kad duomenys būtų teisingi ir nepasimestų nei viename iš schemoje pavaizduotų etapų. Visos grandinės testavimas yra labai sudėtinga užduotis, visam procesui ištestuoti reikėtų ne vieno skirtingo testavimo įrankio. Todėl šiame darbe bus analizuojama siauresnė grandinės dalis – didžiulių duomenų kiekių perkėlimo iš „Hadoop“ duomenų saugyklos į „HP Vertica“ duomenų bazę testavimas, užtikrinimas, kad perkeltant duomenis jie nepasimetė, nesusidubliavo ir yra vienodi abiejuose duomenų bazėse.

3.1. „Hadoop“ ir „HP Vertica“ duomenų bazių palyginimas

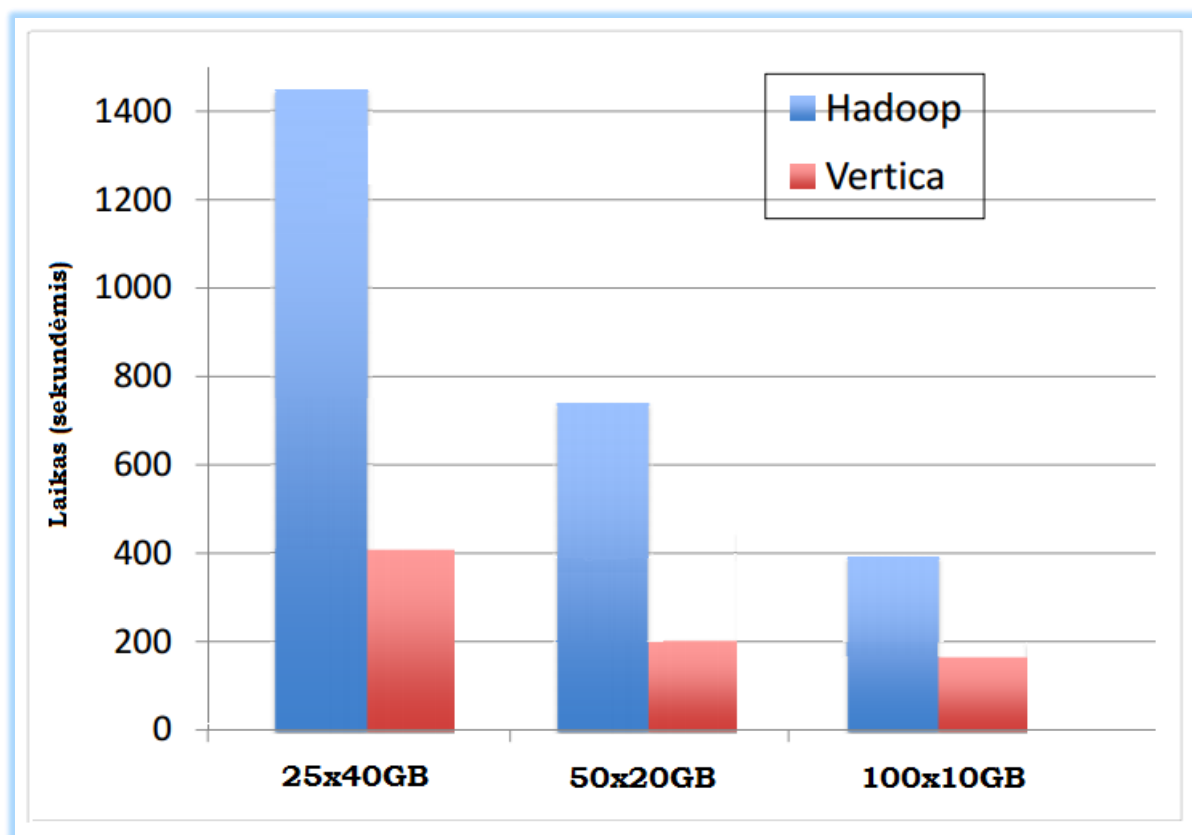
„Hadoop“ ir „HP Vertica“ yra labai skirtingi įrankiai, skirti visiškai skirtingiems verslo poreikiams aptarnauti. „HP Vertica“ yra keičiamo dydžio SQL duomenų bazė, kurioje duomenys paskirstyti stulpeliuose. Pagrindinis šios duomenų bazės privalumas yra tas, kad duomenų bazė yra optimizuota konkrečioms užduotims ir visos užklausos yra vykdomos tiksliai apibrėžtose duomenų srityse. O „Hadoop“ susideda iš dviejų pagrindinių komponentų: HDFS – paskirstytųjų failų sistemos, kuri gali laikyti šimtais petabaitų duomenų ir YARN – išteklių valdymo sistema, kuri skirta aplikacijų, skirtų duomenų naudojimui iš HDFS, kūrimui.

„Hadoop“ ir „HP Vertica“ sistemų parametrų palyginimas pateiktas žemiau esančioje lentelėje. [18]

Pavadinimas	„Hadoop“ duomenų bazė	„HP Vertica“ duomenų bazė
Duomenų bazės modelis	Plataus mastelio stulpelių saugykla	Reliacinė duomenų bazių valdymo sistema (DBVS)
Tinklapis	www.hbase.apache.org	www.vertica.com
Paskutinė išleista versija	1.1.4, 2016 m. kovas	7.1.x, 2015 m. birželis
Licencija	Atviras kodas	Mokama
Serverio operacinės sistemos	Linux Unix Windows (naudojant „Sygwin“)	Linux
XML palaikymas	Nepalaiko	Palaiko
SQL	Nepalaiko	Palaiko
APIs ir kiti prieinamumo metodai	JAVA API RESTful HTTP API Thrift	ADO.NET JDBC ODBC
Palaikomos programavimo kalbos	C C# C++ Groovy JAVA PHP Python Scala	C++ JAVA Perl Python R
Trigeriai	Palaiko	Nepalaiko
Išoriniai raktai	Nepalaikomi	Palaikomi



4 pav. 1 TB duomenų užkrovimo laikas



5 pav. Užklauso vykdomo laikas

4. Didelių duomenų kiekių testavimas

Didelių duomenų kiekių testavimas yra vienas iš didžiausių iššūkių su kuriuo susiduria organizacijos, dėl to, kad nėra aišku ką testuoti ir kiek daug duomenų testuoti. Organizacijos susiduria su iššūkiais bandydamos sugalvoti testavimo strategijas struktūrizuotų ir nestruktūrizuotų duomenų patikrinimui, kuriant optimalią testavimo aplinką, dirbant su nereliacinėmis duomenų bazėmis ir atliekant nefunkcinį testavimą. Dėl šių iššūkių bloga duomenų kokybė, netikslumai produkcinėje aplinkoje ir kaštų padidėjimas yra neišvengiami. Reikia sugalvoti patikimas testavimo būdą tam, kad struktūrizuoti ir nestruktūrizuoti duomenys būtų patikrinti. Testavimas turi būti pradėtas kuo anksčiau, kad galimi duomenų trūkumai (defektai) būtų pastebėti dar ankstyvoje gyvavimo ciklo įgyvendinimo stadijoje ir taip sumažinami visi kaštai ir duomenų išleidimo į rinką laikas.

Skirtingi testavimo tipai, tokie kaip funkcinis ir nefunkcinis, turi būti atliekami kartu su griežtu duomenų testavimu ir testavimo aplinkų priežiūra bei valdymu tam, kad būtų galima įsitikinti, jog duomenys ateinantys iš skirtingų duomenų šaltinių yra apdorojami be jokių klaidų ir yra geros kokybės, teisingi, tinkami atlikti analizei. Funkcinio testavimo funkcijos, tokios kaip MapReduce (liet. plano sumažinimas) proceso patikrinimas, struktūrizuotų ir nestruktūrizuotų duomenų patikrinimas, duomenų saugyklų (angl. *storage*) patikrinimas yra taip pat labai svarbi testavimo dalis norint įsitikinti, kad duomenys yra teisingi ir geros kokybės. Neskaitant funkcinio patikrinimo, kiti nefunkciniai testavimo būdai tokie, kaip greitaveikos arba duomenų perkėlimo testavimas užima pačią svarbiausią rolę ir užtikrina viso proceso lankstumą. [19]

Didelių duomenų kiekių testavimas – tai patvirtinimas, kad saugomi duomenys yra tikslūs, juose nėra pasikartojančių tų pačių įrašų, nėra duomenų praradimų ar iškraipymų, turimi duomenys yra teisingi. Duomenų testavimas „Hadoop“ ir „HP Vertica“ duomenų bazėse gali būti atliekamas tada, kai visi duomenys yra pilnai gauti abiejuose duomenų bazėse, nevėluoja, nėra užstrigę kituose duomenų perdavimo grandyse. Jis dažniausiai atliekamas vakarykštės dienos duomenų patikrinimui, kadangi yra žinoma, jog duomenys ilgiau vėluoti negali. Duomenų testavimas yra reikalingas tam, kad būtų išsiaiškinti ir apibrėžti duomenų neatitikimai, trūkumai, klaidos. Tikslas yra patenkinti suinteresuotų šalių poreikius ir garantuoti aukščiausią duomenų kokybę atliekant testavimą.

4.1. Duomenų vientisumo svarba duomenų bazėse

Duomenų vientisumas siejamas su informacijos, esančios duomenų bazėse, teisingumu ir išsamumu. Turi būti užtikrinami šie reikalavimai: reikšmių būtinumas (NOT NULL), reikalavimai reikšmėms, lentelės raktų vientisumas, nuorodų vientisumas, dalykinės taisyklės, duomenų neprieštarinumas.

Neįtikėtinai dideli duomenų kiekiai yra gaunami kasdien iš daugybės skirtingų šaltinių be galo greitai, labai dideliais kiekiais ir įvairove. Kai kalbama apie labai didelio masto, nuolat augančius duomenų kiekius – jų valdymas turi būti prioritetas. Gaunami duomenys gali būti įvairiausių formatų: viskas iš skaičių ir formulių, atskirų žodžių ir teksto fragmentus. Kad patikrintų, surinktų duomenų vientisumą ir tikslumą, kai kurie darbuotojai remiasi savo sukurtas skaičiuokles ir "Word" dokumentus, o kiti, daugiau patirties su duomenimis turintys darbuotojai naudotų automatinis duomenų vizualizacijos įrankius. [20]

Dėl to, kad duomenys ateina į dvi skirtingas platformas iš daugybės skirtingų duomenų šaltinių, surinktos informacijos patikimumas negali būti patikimai nustatytas. Tačiau, kadangi tie patys duomenys ateina į dvi skirtingas duomenų platformas – juose esančius duomenis galima palyginti tarpusavyje.

4.2. Duomenų perkėlimo (migravimo) testavimo algoritmas

Duomenų bazėse duomenys yra laikomi daugybėje skirtingų lentelių, sukurtų kiekvienam skirtingam verslo sprendimui ar vidiniams organizacijos poreikiams. Tai reiškia, kad įvairaus dydžio skirtingų lentelių duomenų bazėje yra be galo daug. Didelių duomenų kiekių migravimo testavimas yra atliekamas kiekvienai duomenų bazei iškart po duomenų perkėlimo. Nagrinėjamu atveju turime dvi skirtingas duomenų bazes – „Hadoop“ ir „HP Vertica“, kur didžiuliai duomenų srautai keliauja kasdien iš „Hadoop“ duomenų bazės į „HP Vertica“ duomenų bazę. Norint patikrinti, ar dviejų skirtingų duomenų bazių duomenys sutampa, reikia naudotis toliau pristatytu algoritmu. Pirmasis algoritmo žingsnis yra sukurti užklausas, kurios išrenka visus norimus patikrinti duomenis iš abiejų duomenų bazių. Vykdomos užklausos gali skirtis, tačiau turi grąžinti tą patį rezultatą, jeigu duomenų klaidų tikrinamuose duomenyse nėra. Užklausos visada šiek tiek skiriasi, nes yra vykdomos skirtinguose duomenų bazėse, kur gali skirtis stulpelių pavadinimai, duomenų formatas, rūšiavimas ar netgi pati duomenų talpinimo lentelėje struktūra, vienur duomenys gali būti transformuoti, o kirtu ne. Kitas žingsnis yra įvykdyti paruoštas

užklausas, išsaugoti gautus rezultatus ir vėliau juos palyginti. Gautas eilučių skaičius ir kiekvienas stulpelis turi būti identiški. Galima gauti tokius palyginimo rezultatus:

- Viena iš duomenų dalių arba visi duomenys lentelėje yra tik toje duomenų bazėje, iš kurios duomenys buvo perkelti – vadinasi, ta duomenų dalis (arba visi duomenys) nebuvo perkelti visai. Tikėtina, kad šiuo atveju duomenų perkėlimo procese galėjo įvykti klaida.
- Viena iš duomenų dalių arba visi duomenys lentelėje yra tik toje duomenų bazėje, į kurią duomenys buvo iškeliami – tai reiškia, kad jie galėjo būti perkelti neteisingai (pavyzdžiui, ne į tą duomenų bazės stulpelį) arba įvyko perkėlimo proceso klaida.
- Duomenų dalis yra abiejuose duomenų bazėse, tačiau duomenys yra skirtingi, nesutampa jų reikšmės – taip gali atsitikti dėl duomenų dubliavimosi arba perkėlimo ne į tą stulpelį.
- Duomenys sutampa ir yra identiški. Tai reiškia, kad duomenų perkėlimas įvyko sėkmingai, be klaidų.

Tokiu pat būdu tikrinimą turėtų būti kiekviena duomenų bazės lentelė. Taigi, norint patikrinti visus duomenis, perkeltus į naują duomenų bazę, reikia parašyti daug užklausų tiek vienai, tiek kitai duomenų bazei bei jas suderinti tarpusavyje, kad būtų užtikrinta aukščiausia duomenų kokybė.

Duomenų perkėlimo testavimas gali būti atliekamas tiek automatinio, tiek rankinio būdu. Testuojant rankiniu būdu reikėtų kažkur turėti išsaugotas užklausas, vėliau kažkur išsaugoti gautus rezultatus. Pats gautų rezultatų palyginimas būtų atliekamas naudojantis „MS Excel“ programa arba koku nors kitu įrankiu, kuriuo būtų galima palyginti dviejų lentelių duomenis. O radus duomenų nesutapimų – rastas klaidas reikėtų kur nors registruoti, kad vėliau galima atlikti tolimesnę jų analizę ir ištaisyti – tam reikėtų dar vieno papildomo įrankio. Kitaip tariant, tai niekada nesibaigiantis ir sunkus darbas, nes nauji duomenys yra generuojami kiekvieną akimirką. Rankiniam testavimo būdui reikia labai daug žmogiškųjų resursų arba galima ištestuoti tik labai mažą dalį duomenų, todėl pastarasis testavimo būdas yra netinkamas. Šiame darbe bus analizuojamas didžiulių duomenų kiekių automatinis testavimas. Automatiniam duomenų perkėlimo testavimui atlikti reikalingi papildomi įrankiai, kurie galėtų įvykdyti anksčiau minėtą duomenų perkėlimo testavimo algoritmą. 4 skyriuje pateikiami keletas metodų, kuriais naudojantis galima testuoti didelių duomenų kiekių migravimą, o 5 skyriuje pristatomas komercinis įrankis „QuerySurge“ yra vienas iš populiariausių automatinio testavimo įrankių, skirtų šiam tikslui įgyvendinti. Tačiau, tokią programą, kuri automatiškai atliktų visus veiksmus aprašytus anksčiau pateiktame testavimo algoritme, galima sukurti ir įgyvendinti savarankiškai. 6 skyriuje aprašytas jos įgyvendinimas. [21]

4.3. Duomenų vientisumo testavimas automatiškai būdu

Testuojant duomenų vientisumą automatiškai būdu, testavimo efektyvumas padidėtų, būtų išvengta žmogiškųjų klaidų, atsirastų galimybė duomenų vientisumo kokybę užtikrinti žymiai didesniems duomenų kiekiams iš karto, o ne testuoti atskiras duomenų dalis. Tokiu būdu pats testavimas taptų vertingesnis, o taip pat būtų sutaupomi laiko ir žmogiškieji resursai. Automatinio duomenų vientisumo testavimo sprendimo sukūrimas kainuoja ne pigiai, tačiau ilgalaikėje perspektyvoje tai padėtų labai stipriai optimizuoti darbo laiką, ir užtikrinti daug aukštesnę duomenų vientisumo kokybę „Hadoop“ ir „HP Vertica“ duomenų bazėse.

4.4. Kiti duomenų vientisumo testavimo metodai

„QuerySurge“ įrankio kūrėjų teigimu, yra keletas kitų metodų (algoritmų), skirtų dideliems duomenų kiekiams testuoti: užklausų atimties (angl. *minus queries*) ir duomenų atrinkimo (angl. *sampling*) metodai. Tai labai dažnai duomenų testuotojų naudojami algoritmai, tačiau pasak „QuerySurge“ įrankio kūrėjų, norint pasinaudoti jais neišvengiamai reikia mokėti SQL programavimo kalbą, o taip pat reikia ir daug žmogiškųjų resursų bei laiko. Šių duomenų testavimo metodų pagrindiniai veikimo principai aprašyti žemiau pateiktuose skyriuose.

4.4.1. Užklausų atimties testavimo metodas

Užklausų atimties duomenų testavimo metodas yra atliekamas rankiniu testavimo būdu atimant duomenų rinkinius vieną iš kito 2 kartus. Reikia parašyti SQL užklausą, kurioje turi būti naudojamas atimties veiksmo operatorius tam, kad palygintų dviejų duomenų bazės lentelių duomenis. Pirmą kartą iš pirmosios užklausos rezultatų reikia atimti antrosios užklausos rezultatus (pavyzdžiui, rezultatus gautus iš „Hadoop“ duomenų bazės atimti iš „HP Vertica“ duomenų bazės rezultatų). Jeigu atlikus šį veiksmą lieka laisvų duomenų eilučių, kurios nesieliminavo, su jomis reikia atlikti tolimesnę duomenų analizę. Tokiu būdu galima nustatyti duomenų nesutapimus ir klaidas. Jeigu nėra jokių duomenų nesutapimų, atėmus vieną duomenų rinkinį iš kito, turi nebelikti nei vienos duomenų eilutės ar rinkinio. Atlikus pirmąją užklausų atimties algoritmo dalį, reikia atlikti antrąją atimtį: pirmosios užklausos rezultatus reikia atimti iš antrosios užklausos rezultatų, o tada analogiškai, kaip ir pirmosios algoritmo dalies atveju, turi būti atliekama tolimesnė analizė

ir duomenų tikslumo įvertinimas. Tokiu būdu, galima surasti duomenų neatitikimus, susidubliavimus ar kitas klaidas.

Žemiau pateikiamas užklausų atimties testavimo metodo pavyzdys. Tarkime, kad turi dvi duomenų bazės lentelės su duomenimis (žr. 6 pav.)

Pirmoji lentelė - dalyviai_1			Antroji lentelė - dalyviai_2		
Vardas	Pavarde	Pozicija	Name	Surname	Position
Jonas	Beržaitis	LF	Jonas	Beržaitis	LF
Darius	Klevaitis	1B	Darius	Klevaitis	1B
Lina	Liepaitė	CF	Lina	Liepaitė	CF
Tomas	Guobėnas	CF	Tomas	Guobėnas	CF
Justas	Alksnys	SS	Justas	Alksnys	SS
Petras	Lapėnas	C	Petras	Lapėnas	C
Antanas	Klevėnas	2B	Antanas	Klevėnas	2B
Toma	Lapienė	3B	Giedrė	Beržaitė	P
Jonas	Alksnietis	P			

6 pav. Dviejų duomenų lentelių pavyzdys

Sekantis veiksmas yra parašyti užklausą, kuri įgyvendintų pirmąjį algoritmo žingsnį – iš pirmosios lentelės atimtų antrosios lentelės duomenų rinkinį. Užklausa, kuri įvykdo duomenų rinkinių atimtį ir grąžina reikiamus rezultatus, pavyzdys pateiktas žemiau (žr. 7 pav.)

```
SELECT vardas, pavarde, pozicija FROM dalyviai_1
MINUS
SELECT name, surname, position FROM dalyviai_2
```

7 pav. Užklausa, skirta iš pirmosios lentelės „dalyviai_1“ atimti antrąją „dalyviai_2“

Įvykdžius šią užklausą, gautas rezultatas turi būti dvi duomenų eilutės, kurios pirmojoje lentelėje „dalyviai_1“ yra pažymėtos raudona spalva (žr. 8 pav.)

Toma	Lapienė	3B
Jonas	Alksnietis	P

8 pav. Gauti rezultatai, įvykdžius pirmąją užklausą

Sekantis žingsnis yra parašyti užklausą, kurią įvykdžius iš antrosios lentelės duomenų bus atimti pirmosios lentelės duomenys. Užklausa yra formuojama analogiškai kaip ir pirmuoju algoritmo atveju (žr. 9 pav.)

```
SELECT name, surname, position FROM dalyviai_2  
MINUS  
SELECT vardas, pavarde, pozicija FROM dalyviai_1
```

9 pav. Užklausa, skirta iš antrosios lentelės „dalyviai_2“ atimti pirmąją „dalyviai_1“

Įvykdžius antrąją užklausą, gausime skirtingą rezultatą nei pirmuoju atveju. Gautas skirtumas antrojoje lentelėje „dalyviai_2“ yra pažymėtos žalia spalva (žr. 10 pav.)

Giedrė	Beržaitė	P
--------	----------	---

10 pav. Gauti rezultatai, įvykdžius antrąją užklausą

Aukščiau pateiktas užklausų atimties testavimo algoritmas atrodo gana paprastas ir nesudėtingas. Tačiau, tai tampa žymiai sudėtingesniu algoritmu, kai duomenų kiekis yra daug didesnis – keletas ar keliolika milijonų duomenų eilučių ir kelios dešimtys stulpelių. O kadangi šiam duomenų testavimo būdui atlikti reikalinga papildoma duomenų analizė, tai dirbant su didžiuliais duomenų kiekiais tai ne tik užima labai daug darbo laiko, bet ir žmogiškųjų resursų. Todėl šis duomenų testavimo būdas nėra labai efektyvus būdas testuoti duomenis. O šiame darbe analizuoju atveju yra naudojamos dvi skirtingos duomenų bazės „Hadoop“ ir „HP Vertica“, kurioms yra rašomos atskiros užklausos, kurių negalima panaudoti kartu panaudojant atimties operatorių. [22] Todėl norit pritaikyti šį algoritmą, reikėtų naudoti papildomus įrankius arba įvykdyti užklausas atskirai, išsaugoti gautus jų rezultatus atskiroje duomenų bazėje, kurioje jau būtų galima vykdyti atimties užklausas.

4.4.2. Duomenų atrinkimo testavimo metodas

Duomenų atrinkimo (angl. *sampling*) metodas dar kitaip vadinamas akylo stebėjimo ir palyginimo metodu. Šio testavimo metodo idėja yra gera, tačiau ji yra beveik neįmanoma įgyvendinti norint patikrinti visus duomenis. Paprastai duomenų atrinkimo metodo vykdymo eiga yra tokia:

- Verslo (arba duomenų atvaizdavimo) taisyklių analizė. Verslo taisyklės (angl. *business rules*) dar kitaip yra vadinamos duomenų atvaizdavimo taisyklėmis. Sudarytos atvaizdavimo lentelėse yra reikalavimai arba taisyklės, kuriomis remiantis yra vykdomas duomenų gavimas (angl. *extracting*), transformacija (angl. *transforming*) ir įkėlimas (angl. *loading*) – ETL į duomenų saugyklas. Pagrindinės taisyklės, kurios būna pateiktos: lentelių ir laukų pavadinimai, duomenų tipai ir ilgiai; kaip duomenų šaltinių lentelės ar failai turi būti sujungti perkeliant duomenis į duomenų saugyklas; bet kokia kita transformacijų logika, kuri bus pritaikyta; visos verslo taisyklės, kurios bus pritaikytos.
- Testavimo scenarijų sukūrimas. Kiekvienas skirtingas duomenų atvaizdavimas turi turėti skirtingus testavimo scenarijus. Testavimo scenarijai dažniausiai turi dvi užklausų poras, parašytas SQL programavimo kalba arba HQL (jeigu užklausos skirtos duomenims iš „Hadoop“ duomenų bazės). Panašus užklausų porų pavyzdys yra pateiktas 6 skyriuje 6.7 pav.
- Užklausų įvykdymas ir rezultatų gavimas. Testai yra įvykdomi naudojant SQL redaktorių „Toad“, „Squirrel“ arba bet kokį kitą redaktorių. Gauti užklausų rezultatai yra išsaugomi atskiruose „MS Excel“ programos lapuose.
- Rezultatų palyginimas. Palyginti visus gautų rezultatų rinkinius akylai peržiūrint visus duomenis: tūkstančius arba milijonus eilučių ir stulpelių.

Šis metodas yra tinkamas tik darbui su mažais duomenų kiekiais, nes vizualiai palyginti milijardus duomenų rinkinių – šimtus stulpelių ir milijardus eilučių per dvi skirtingas lenteles yra neįmanoma. Todėl šis testavimo metodas yra labai neefektyvus. Apibendrinat, naudojantis šiuo metodu galima palyginti mažiau nei 1 % visų duomenų. Kadangi daugelis kompanijų naudoja verslo stebėseną (angl. *business intelligence*) siekiant priimti strateginius sprendimus ir įgyti konkurencinį pranašumą savo verslo srityje, neteisingai perkelti duomenys gali būti pagrindinė priežastis atlikti blogas įžvalgas, kurios kompanijai gali kainuoti milijonus dolerių.

[23]

5. „QuerySurge“ – įrankis didelių duomenų kiekių testavimui

QuerySurge įrankis yra vienas iš geriausių duomenų testavimo sprendimų. Šis įrankis yra skirtas duomenų saugyklų ir didelių duomenų kiekių testavimo automatizavimui užtikrinant, kad duomenys, gauti iš įvairių duomenų šaltinių, išlieka nepakitę lyginant su jau apdorotais ir paruoštais naudojimui duomenimis duomenų bazėse. QuerySurge įrankis atlieka duomenų analizę ir įvardija visus duomenų skirtumus labai greitai.

Vienas iš pagrindinių QuerySurge įrankio privalumų yra tai, kad duomenų testavimas naudojant šį įrankį yra labai lengvas ir paprastas, todėl jį atlikti gali tiek nepatyręs naujokas, tiek daug patirties ir kompetencijų turintys darbuotojai. QuerySurge įrankio pagalba duomenų testavimas ir tikrinimas yra visiškai automatizuotas, kai tuo tarpu be automatizuoto įrankio reikėtų testuoti rankiniu būdu lyginant keletą atskirų Excel failų arba naudoti MinusQuery metodą.

QuerySurge įrankis palaiko ir daugiau naudingų funkcijų, tokių kaip labai detalių ataskaitų kūrimas, duomenų kokybės būklės grafikai, testavimo valdymo įrankių integravimas ir galimybė prisijungti prie bet kokių duomenų saugyklų. [24]

5.1. Testavimo programos plėtinys „Query Wizards“

Duomenų migravimas iš vienos duomenų bazės į kitą tampa vienu iš didžiausių iššūkių informacinių technologijų specialistams. Nors tokie projektai verslui atneša didžiulę naudą (tokią, kaip išlaidų sumažinimą, produktyvumo padidinimą, ir patobulintus didelių duomenų kiekių valdymo sprendimus), jie taip pat neišvengiamai apima didžiulę riziką, dėl to, kad migruojami didžiuliai duomenų kiekiai ir jie yra labai svarbūs. Siekiant sumažinti riziką ir užtikrinti, kad duomenų transformacija ir migravimas buvo atlikti tinkamai, reikia įgyvendinti išsamų duomenų patikrinimą ir kokybės užtikrinimo strategiją.

Naujausioje „QuerySurge“ produkto versijoje yra įtrauktas įrankis „Query Wizards“. Šio įrankio pagalba galima atlikti duomenų patikrinimą ir tai nereikalauja programavimo žinių ar papildomų kompetencijų. Duomenų patikrinimo darbo krūvis yra žymiai supaprastintas ir atliekamas neįtikėtinais greitai. „Query Wizards“ gali atlikti trijų tipų duomenų palyginimus, kurie išvardinti žemiau:

- Labai greitai atlikti keletą lentelių palyginimą, patikrinant šimtus lentelių per keletą minučių. Šis palyginimas yra labiausiai tinkamas duomenų migravimo arba duomenų bazių versijos atnaujinimams testavimui, kur nėra atliktų visai jokių transformacijų.
- Užtikrinti, kad visos duomenų eilutės permigruotos be klaidų ir patikrinti eilučių skaičių. Labai naudingas viskam – duomenų saugykloms, duomenų migracijoms, duomenų bazių atnaujinimams testuoti.
- Atlikti bet kokią reikalingą keletą stulpelių palyginimą. Tai labai naudinga testuojant didžiules duomenų saugyklas, kur lentelės turi dalį transformuotų duomenų stulpelių ir dalį stulpelių, kurių duomenys dar nėra transformuoti. Toks palyginimas yra skirtas specifiniams stulpeliams, kurių duomenys dar nėra transformuoti ir jie vidutiniškai sudaro 80% visų duomenų. Stulpeliai, kurių duomenys yra transformuoti reikalauja SQL užklausų rašymo kalbos žinių. Daugybė stulpelių gali būti palyginami vienu metu per labai trumpą laiko tarpą.

Query Wizards taip pat palaiko automatizuotas filtravimo („Where“ sąlyga) ir rūšiavimo („Order By“ sąlyga) funkcijas. [25]

6. Duomenų vientisumo testavimo programa „Duomenų revizorius“

„Duomenų revizorius“ – tai naujai sukurtas įrankis (programa), kuris atlieka didelių duomenų kiekių perkėlimo testavimą. Šios programos kūrimui buvo pasirinkta JAVA programavimo kalba. Šiam įrankiui kurti buvo panaudotas anksčiau aprašytas duomenų perkėlimo testavimo algoritmas. Pirmiausia reikia pasirinkti įrankį, kuriame kiekvieną kartą bus saugojami gauti užklausų rezultatai (platesnė įrankių apžvalga yra pateikta skyriuje „Įrankių tarpiniams rezultatams išsaugoti, apžvalga“). Pasirinkau atvirojo kodo įrankį „MongoDB“ – dokumentų duomenų bazę. „MongoDB“ duomenų bazėje bus laikomi ne tik gauti užklausų rezultatai, bet ir pačios užklausos. Užklausos išsaugojamos „MongoDB“ duomenų bazėje, įvykdžius automatinio užklausų išsaugojimo testą. Dažniausiai šis testas yra vykdomas vieną kartą arba vykdomas pakartotinai, jeigu užklausos buvo paredaguotos arba pakeistas testuojamas laikotarpis (žr. 11 pav.).

```
public class testCube {

    @Autowired
    private CheckGroupsService checkGroupsService;

    @Test
    public void testSave() throws Exception {
        String cubeName = "testCube";

        List<CheckItem> items = new ArrayList<>();
        items.add(checkTestCube());

        checkGroupsService.setCheckItemsForCheckGroup(cubeName, items);
    }

    private CheckItem checkTestCube() {
        List<CheckSetup> setups = new ArrayList<>();

        setups.add(new SqlCheckSetup(new HiveSource("hive"),
            "SELECT SUBSTRING(logdate, 0, 10) as dt, \n" +
            "COUNT(distinct id_programs) as d_id_campaigns, \n" +
            "CASE when uniqueorderid=1 then 'true' else 'false' end as uniqueorderid, \n" +
            "SUM(total) as total \n" +
            "FROM Cubes.TestCube WHERE yyyyymmdd >= {dateValueFrom} AND yyyyymmdd < {dateValueTo} \n" +
            "GROUP BY SUBSTRING(logdate, 0, 10), uniqueorderid \n" +
            "ORDER BY dt, d_id_campaigns, uniqueorderid"));

        setups.add(new SqlCheckSetup(new VerticaSource("vertica"),
            "SELECT CAST(logtime as VARCHAR(10)) as dt, \n" +
            "COUNT(distinct id_campaigns) as d_id_campaigns, \n" +
            "CASE when uniqueorderid=1 then 'true' else 'false' end as uniqueorderid, \n" +
            "SUM(total) as total \n" +
            "FROM HCubes.Test WHERE LogTime >= '{dateValueFrom}' AND LogTime < '{dateValueTo}' \n" +
            "GROUP BY CAST(logtime as VARCHAR(10)), uniqueorderid \n" +
            "ORDER BY dt, d_id_campaigns, uniqueorderid"));

        return new CheckItem(
            new DayStepIterationValue(DateHelper.dateFor(2015, 1, 1)), setups);
    }
}
```

11 pav. Užklausų, skirtų duomenims iš „Hadoop“ ir „HP Vertica“ duomenų bazių gauti, išsaugojimo testas

Po sėkmingo užklausų išsaugojimo jas galima rasti „MongoDB“ duomenų bazėje. Paieška vykdoma labai paprastai, užtenka nurodyti tik išsaugojimo teste nurodytą lentelės vardą – *cubeName* pagal kurį iškart išfiltruojamos išsaugotos užklausos (žr. 14 pav.). Išsaugotų užklausų paieškai galima naudoti ir kitus požymius. Pavyzdžiui, jeigu žinome užklausoms suteiktą unikalų ID (*_id*, žr. pirmasis ID pateiktame pavyzdyje 14 pav.), užklausų sukūrimo (*created*) arba jų redagavimo (*modified*) datą, ar bet kurį kitą požymį, paiešką galima vykdyti pagal tai arba netgi pagal keletą iš šių požymių vienu metu. Apibendrinant užklausų paieškos vykdymą, galima teigti, kad šis įrankis yra labai lankstus ir patogus naudoti, o atfiltruotus rezultatus galime peržiūrėti 3 skirtingais režimais: medžio struktūros, lentelės ir tekstiniu režimu. Pastarasis šablonas yra pavaizduotas 14 pav. Minėtus duomenų atvaizdavimo šablonus galima keisti vieno mygtuko paspaudimu, visi jie matomi viršutiniame dešiniajame pavyzdžio kampe. Manau, kad šis įrankis puikiai tiktų ir galėtų būti naudojamas žmonių, kurie neturi ypatingai gilių techninių žinių.

Datos (*date*) požymis – tai data, kuri buvo nurodyta mūsų užklausų išsaugojimo teste (žr. 11 pav.). Ši data reiškia, kad užklausa vykdomas testas automatiškai įrašys nurodytą datą į užklausos sąlygą vietoj *{dateValueFrom}* reikšmės, o vietoj *{dateValueTo}* bus įrašyta viena diena vėlesnė data. Pavyzdžiui, jeigu teste nurodėme datą 2015-01-01, pirmą kartą vykdomas užklausa jų *WHERE* sąlygos bus suformuluotos taip:

```
WHERE yyyyymmdd >= 20150101 AND yyyyymmdd < 20150102
```

12 pav. Įrašyta *WHERE* sąlyga formuojant „Hadoop“ užklausa

```
WHERE LogTime >= '2015-01-01' AND LogTime < '2015-01-02'
```

13 pav. Įrašyta *WHERE* sąlyga formuojant „HP Vertica“ užklausa

Tai reiškia, kad bus tikrinami vienos dienos (pateikto pavyzdžio atveju 2015 m. sausio 1 dienos) duomenys.

Jeigu norime patikrinti labai ilgo, keleto ar keliolikos metų, laikotarpio duomenis, kad duomenų testavimas vyktų greičiau galima pasirinkti testuoti duomenis ne kas vieną dieną, o kas savaitę. Tokiu atveju, savo užklausų išsaugojimo teste turime nurodyti ne užklausų vykdymą padieniui (angl. *DayStepIterationValue*), o kas savaitinį intervalo pasirinkimą (angl. *WeekStepIterationValue*). Pastaruoju atveju, po užklausos išsaugojimo testo, „MongoDB“ duomenų bazėje atsifiltravę savo išsaugotas užklausas, intervalo požymiui (*interval*) bus priskirtas ne skaičius 1 (1 dienos intervalas), o skaičius 7 (7 dienų – savaitės intervalas) ir užklausų vykdymo

metu atitinkamai bus parenkamos kitos – 7 dienų intervalo datos. Taigi, pasirinkus 1 dienos intervalą bus gaunami vienos dienos užklausų rezultatai, pasirinkus 7 –savaitės laikotarpio užklausų rezultatai. Pasirinkus didesnę – savaitės intervalą, klaidos aptikimo atveju yra daug sunkiau analizuoti klaidos atsiradimo laiką, ir priežastis. Tačiau tokiu atveju, kaip ir buvo minėta anksčiau, galima paredagavus intervalą ir išsaugojus testą iš naujo, nesunkiai gauti rezultatus už 7 dienas atskirai, kad klaidos analizės apimtis sumažėtų.

```

db.getCollection('checkGroup').find({'name':'testCube'})

```

checkGroup 0.149 sec. 0 50

```

{
  "_id" : ObjectId("5708c7a0df1d849e78cf9b1a"),
  "_class" : "checks.CheckGroup",
  "created" : ISODate("2016-04-09T09:13:04.766Z"),
  "modified" : ISODate("2016-04-09T09:13:04.766Z"),
  "name" : "testCube",
  "checkItems" : [
    {
      "_id" : ObjectId("5708c7a0df1d849e78cf9b19"),
      "stepIterationValue" : {
        "_class" : "iteration.DayStepIterationValue",
        "date" : ISODate("2015-01-01T00:00:00.000Z"),
        "interval" : 1,
        "hash" : NumberLong(20150101)
      },
      "checkSetups" : [
        {
          "sql" : "SELECT SUBSTRING(logdate, 0, 10) as dt, COUNT(distinc
          "source" : {
            "_class" : "checks.setups.sources.HiveSource",
            "dataSource" : "hive"
          },
          "_class" : "checks.setups.SqlCheckSetup"
        },
        {
          "sql" : "SELECT CAST(logtime as VARCHAR(10)) as dt, COUNT(dist
          "source" : {
            "_class" : "checks.setups.sources.VerticaSource",
            "dataSource" : "vertica"
          },
          "_class" : "checks.setups.SqlCheckSetup"
        }
      ]
    }
  ]
}

```

14 pav. Išsaugotų užklausų paieška naudojant „Robomongo“ įrankį

14 pav. pateiktame pavyzdyje testą sudaro tik viena pora užklausų. Tačiau norint patikrinti didesnę kiekį duomenų galima į vieną testą sudėti daugiau užklausų porų. Kad testai nesusimaišytų, jų rezultatai nepersipintų ir nebūtų pradėti neteisingai lyginti, kiekviena užklausų pora turi savo unikalų ID (*checkItems* – *_id*, žr. 14 pav. Išsaugotų užklausų paieška naudojant

„Robomongo“ įrankį). Naudojant šį ID (angl. *checkItemId*) gautų rezultatų lentelėje labai lengvai galima atsifiltruoti duomenis konkrečiai užklausų porai. O panaudojus ir datos filtrą kartu – gausime atfiltruotus tik konkrečios nurodytos datos ar dienų intervalo rezultatus. 15 pav. pateiktame pavyzdyje išfiltruoti užklausų rezultatai pagal *checkItemId* ir konkrečią datą 20150101. Gauti paieškos rezultatai yra 2 įrašiai: vienas iš jų yra gauti rezultatai įvykdžius užklausą „Hadoop“ duomenų bazėje (*dataSource = hive*), kitas įrašas – gauti rezultatai įvykdžius užklausą „HP Vertica“ duomenų bazėje (*dataSource = vertica*). Ar abiejų užklausų gauti atsakymai yra vienodi ir duomenys sutampa galima pamatyti prie palyginimo (*verified*) atributo esančios statuso reikšmės. Jeigu palyginimo statusas yra „teisingas“ (angl. *true*) – reiškia duomenys sutampa. Tokiu atveju gautos rezultatų reikšmių eilutės (*resultSet - rows*) yra ištrinamos (žr. 15 pav.). Jeigu palyginimo statuso reikšmė yra „klaidinga“ (angl. *false*) – reiškia duomenys nesutampa ir rezultatų reikšmių eilutės (*resultSet - rows*) yra paliekamos, kad būtų galima atlikti tolimesnę analizę. Pastarasis atvejis pavaizduotas 16 pav.

```
db.getCollection('generationResult').find({"checkItemId":"5708c7a0df1d849e78cf9b19",
"stepIterationValue.hash":20150101})
```

generationResult 0.037 sec.

Key	Value	Type
(1) ObjectId("56897f7520a38968a8403292")	{ 9 fields }	Object
_id	ObjectId("56897f7520a38968a8403292")	ObjectId
_class	generations.GenerationResult	String
checkItemId	5708c7a0df1d849e78cf9b19	String
created	2016-04-10 20:07:17.983Z	Date
modified	2016-04-10 02:56:08.824Z	Date
source	{ 2 fields }	Object
_class	checks.setups.sources.HiveSource	String
dataSource	hive	String
stepIterationValue	{ 5 fields }	Object
resultSet	{ 1 fields }	Object
rows	Array [0]	Array
verified	true	Boolean
(2) ObjectId("56897f7520a38968a8403293")	{ 9 fields }	Object
_id	ObjectId("56897f7520a38968a8403293")	ObjectId
_class	generations.GenerationResult	String
checkItemId	5708c7a0df1d849e78cf9b19	String
created	2016-04-10 20:08:20.983Z	Date
modified	2016-04-10 02:58:12.824Z	Date
source	{ 2 fields }	Object
_class	checks.setups.sources.VerticaSource	String
dataSource	vertica	String
stepIterationValue	{ 5 fields }	Object
resultSet	{ 1 fields }	Object
rows	Array [0]	Array
verified	true	Boolean

15 pav. Atfiltruotų užklausų rezultatų pavyzdys naudojantis „Robomongo“ įrankiu, kai duomenų palyginimo statusas yra „teisingas“

```
db.getCollection('generationResult').find({"checkItemId": "5708c7a0df1d849e78cf9b19", "stepIterationValue.hash": 20150101})
```

Key	Value	Type
(1) ObjectId("56897f7520a38968a8...")	{ 9 fields }	Object
(2) ObjectId("56897f7520a38968a8...")	{ 9 fields }	Object
_id	ObjectId("56897f7520a38968a8403293")	ObjectId
_class	generations.GenerationResult	String
checkItemId	5708c7a0df1d849e78cf9b19	String
created	2016-04-10 20:08:20.983Z	Date
modified	2016-04-10 02:58:12.824Z	Date
source	{ 2 fields }	Object
stepIterationValue	{ 5 fields }	Object
resultSet	{ 1 fields }	Object
rows	Array [2]	Array
0	{ 1 fields }	Object
cells	{ 4 fields }	Object
dt	2015-01-01	String
d_id_campaigns	13048	Int64
uniqueorderid	true	String
total	2669221	Int64
1	{ 1 fields }	Object
cells	{ 4 fields }	Object
dt	2015-01-01	String
d_id_campaigns	16732	Int64
uniqueorderid	false	String
total	1410668	Int64
verified	false	Boolean

16 pav. Atfiltruotų užklausų rezultatų pavyzdys naudojantis „Robomongo“ įrankiu, kai duomenų palyginimo statuso reikšmė yra „klaidinga“

Kad leidžiant užklausų vykdymo servisą nebūtų generuojami to paties laikotarpio rezultatai, užklausų vykdymo servisas automatiškai patikrina paskutinę datą visiems išsaugotiesiems rezultatams pagal jų unikalų ID ir užklausų sąlygose nurodo sekančią datą. Tokiu būdu yra išvengiama rezultatų dubliavimosi.

6.1. Skirtingų duomenų bazių užklausų palyginimas

Pirmiausia sukuriamos dvi užklausos, grąžinančios rezultatus iš duomenų bazių „Hadoop“ ir „HP Vertica“. Nors užklausos yra skirtingos atskiroms duomenų bazėms, tačiau jų grąžinamas rezultatas yra tas pats. Paanalizuokime užklausų palyginimo pavyzdį, kuris yra pateiktas 17 pav. Pirmoji (pažymėta raudona spalva) užklausa yra skirta duomenų gavimui iš „Hadoop“ duomenų bazės naudojantis „Hive“ įrankiu. Antroji (pažymėta žalia spalva) užklausa yra skirta duomenų gavimui iš „HP Vertica“ duomenų bazės. Spalvotai pažymėtos eilutės nėra identiškos, tamsiau paryškintos vietos skiriasi. Pirmoje eilutėje matosi 2 skirtumai: skiriasi

laukelių pavadinimai *logdate* („Hadoop“ duomenų bazėje) ir *logtime* („HP Vertica“ duomenų bazėje) bei *string* tipo lauko iškirpimo sintaksė.

```

1. SELECT SUBSTRING(logdate, 0, 10) as dt,
2. COUNT(distinct id_programs) as d_id_campaigns,
3. CASE when uniqueorderid=1 then 'true' else 'false' end as uniqueorderid,
4. SUM(total) as total
5. FROM Cubes.TestCube WHERE yyyyymmdd >= 20150101 AND yyyyymmdd < 20150102
6. GROUP BY SUBSTRING(logdate, 0, 10), uniqueorderid
7. ORDER BY dt, d_id_campaigns, uniqueorderid

1. SELECT CAST(logtime as VARCHAR(10)) as dt,
2. COUNT(distinct id_campaigns) as d_id_campaigns,
3. CASE when uniqueorderid=1 then 'true' else 'false' end as uniqueorderid,
4. SUM(total) as total
5. FROM HCubes.Test WHERE LogTime >= '2015-01-01' AND LogTime < '2015-01-02'
6. GROUP BY CAST(logtime as VARCHAR(10)), uniqueorderid
7. ORDER BY dt, d_id_campaigns, uniqueorderid

```

17 pav. Užklausų, paruoštų skirtingoms duomenų bazėms, palyginimo pavyzdys

Iškirpimas naudojamas tam, kad mūsų gautų rezultatų apimtis būtų mažesnė ir gautus rezultatus būtų lengviau palyginti (žr. 18 pav.). Žinoma, tuo atveju, kai testai aptinka klaidą, analizuoti rezultatus tampa žymiai sudėtingiau, nes mes gaunami tik bendrus visos dienos duomenis. Turėti detalesnius bei tikslesnius duomenis klaidos atveju yra žymiai geriau, nes iš karto būtų galima pamatyti kurią valandą / minutę / sekundę / milisekundę buvo pastebėtas duomenų nesutapimas ir atlikti tolimesnę klaidos analizę. Tačiau saugoti tokį didžiulį kiekį duomenų yra per brangu, o ir gautų duomenų palyginimas užtruktų žymiai ilgiau. Todėl klaidos atvejais galima užklausą įvykdyti iš naujo paliekant valandos ar minutės tikslumo duomenis ir palyginti gautus rezultatus iš naujo, jau gavus detalesnę informaciją.

Recent queries Query Log Columns Results Chart				
	dt	d_id_campaigns	uniqueorderid	total
0	2015-01-01	13048	true	2669221
1	2015-01-01	16732	false	1410668

18 pav. Gauti rezultatai tikrinant duomenis dienos tikslumu

Recent queries Query Log Columns Results Chart				
	dt	d_id_campaigns	uniqueorderid	total
0	2015-01-01 10:47:41.550	1	true	1
1	2015-01-01 10:47:41.563	1	true	1
2	2015-01-01 10:47:41.567	1	true	1
3	2015-01-01 10:47:41.603	1	true	1
4	2015-01-01 10:47:41.610	1	true	1
5	2015-01-01 10:47:41.650	1	true	1
6	2015-01-01 10:47:41.667	1	true	1
7	2015-01-01 10:47:41.683	1	true	1
8	2015-01-01 10:47:41.693	1	false	1
9	2015-01-01 10:47:41.727	1	true	1
10	2015-01-01 10:47:41.737	1	false	1
11	2015-01-01 10:47:41.737	1	true	1
12	2015-01-01 10:47:41.793	1	true	1
13	2015-01-01 10:47:41.807	1	false	1
14	2015-01-01 10:47:41.817	1	true	1

19 pav. Gauti rezultatai tikrinant duomenis milisekundžių tikslumu

6.2. Įrankių, skirtų tarpiniams rezultatams išsaugoti, apžvalga

Tarpiniams užklausų rezultatams išsaugoti pasirinkta „NoSQL“ duomenų bazė. Lyginant su reliacinėmis, šios duomenų bazės yra lankstesnės ir pasižyminčios geresne greitimeika. Vienas iš „NoSQL“ duomenų saugojimo struktūros pranašumų lyginant su reliacinėmis duomenų bazėmis, yra didelių duomenų kiekių saugojimas, kai jų struktūra nuolatos keičiasi, yra pusiau struktūrizuoti arba visai nestruktūrizuoti duomenys.

Viena populiariausių „NoSQL“ duomenų bazių yra „MongoDB“ – dokumentų duomenų bazė. Pagrindiniai šios duomenų bazės privalumai yra greitimeika, dokumentų modelis ir lanksti schema. „MongoDB“ įrankis yra pagrįstas „raktas – reišmė“ (angl. *key value*) duomenų saugojimo struktūra, todėl veikia labai greitai. Pagrindinis duomenų išsaugojimo vienetas yra dokumentas. Tai yra paprastas JSON objektas, kuris „MongoDB“ duomenų bazėje yra vadinami BSON. Pavyzdžiui, jeigu straipsnis turi pavadinimą, tekstą ir komentarus. Reliacinėje duomenų bazėje komentarai būtų išsaugoti atskiroje lentelėje ir norint gauti duomenis reikėtų apjungti straipsnio lentelę ir komentarų lentelę, kad visi duomenys būtų vienoje vietoje. Dokumentų modelis šiuo atveju yra žymiai patogesnis, nes visas straipsnis kartu su komentarais yra

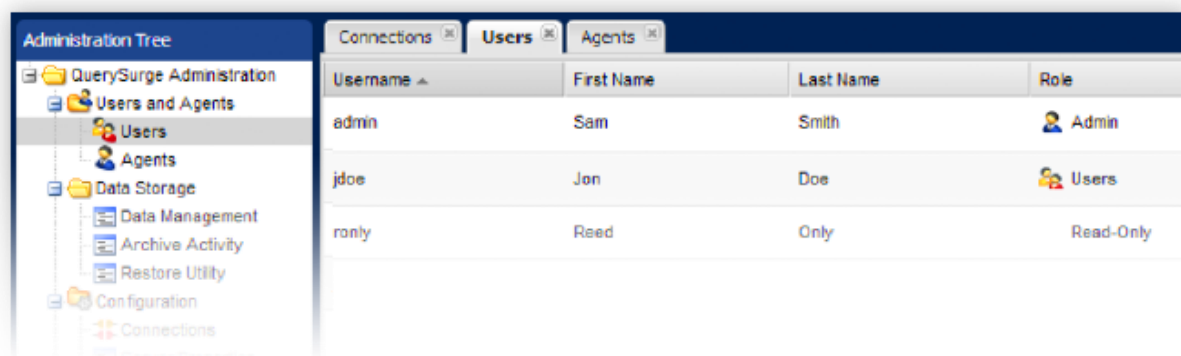
išsaugomas viename dokumente. Dokumentų modelio struktūra taip pat palengvina programavimo užduotis. Dokumentai neturi stulpelių, eilučių, SQL programavimo kalbos. Jeigu reikia apdoroti daug nestruktūrizuotų duomenų, „MongoDB“ duomenų bazė yra geriausias pasirinkimas. Schemas „MongoDB“ yra sukuriamos iškart po duomenų išfiltravimo, o naujo tipo lauką galima pridėti bet kuriuo metu. Taigi, „MongoDB“ neturi jokių schemų, dokumentas gali turėti neribotą kiekį laukų, nauji laukai gali būti pridedami dinamiškai prie bet kurio iš dokumentų, nereikia perskaičiuoti indeksų. Tačiau siekiant išlaikyti puikią greitaveiką, „MongoDB“ įrankis nepalaiko tranzakcijų ir duomenų sujungimo operacijų (angl. *join operation*). Duomenų sujungimą galima atlikti tik sukūrus atskiras užklausas. Dar vienas „MongoDB“ duomenų bazės trūkumas yra atminties apribojimas. Naudojamos atminties dydis priklauso nuo virtualios operacinės sistemos arba programinės įrangos apimties. 34 bitų mašinoje galima saugoti tik apie 4GB duomenų, todėl klientinėms aplinkoms 64 bitų mašinos yra privalomos. Kai atmintis yra pilnai užpildyta, naujų duomenų įterpimai bus nesėkmingi be jokių papildomų perspėjimų. Apibendrinant, „MongoDB“ duomenų bazė neturi kai kurių galimybių, kurias turi reliacinės duomenų bazės, tokių kaip tranzakcijų palaikymo ar duomenų sujungimo (angl. *join operation*), tačiau turi galimybę būti lengvai plečiamą ir lanksčią schemą, kad būtų galima nesudėtingai manipuluoti JSON duomenų formatu. „MongoDB“ įrankis yra labai efektyvus, jeigu jis yra naudojamas tinkamai, bet tai nėra vienintelis įrankis galintis išspręsti visas problemas. Šis įrankis nepakeis SQL duomenų bazės, jis gali būti naudojamas kaip papildomas įrankis greta. [26]

Kita analizuojama atvirojo kodo „NoSQL“ duomenų bazė – „Redis“. Ši duomenų bazė taip pat kaip ir „MongoDB“ yra pagrįsta „raktas – reikšmė“ (angl. *key value*) duomenų saugojimo struktūra. „Redis“ duomenų bazė yra labai greita, visos vykdomos operacijos tiesiogiai užima vietą atmintyje, priešingai nei kitos duomenų bazės, kurios užpildo diską nuskaitant ir įrašinėjant duomenis. Tačiau „Redis“ duomenų saugyklos dydis yra ribotas priklausomai nuo laisvos atminties kiekio. Dauguma duomenų bazių, kurios yra pagrįstos „raktas – reikšmė“ duomenų saugojimo struktūra, turi ribotą duomenų tipų kiekį, tačiau šis įrankis palaiko didžiulį kiekį skirtingų duomenų tipų, galima laikyti tekstinių laukų sąrašus ir rinkinius. Palaikomas skirtingų duomenų tipų kiekis gali būti palyginamas su aukšto lygio programavimo kalbų, kurios yra naudojamos internetinių svetainių kūrimui, palaikomais duomenų kiekiais. Todėl duomenų bazės ir programos kodo apjungimas yra daug paprastesnis nei naudojantis reliacinėmis duomenų bazėmis, nors žinoma, tai labai priklauso nuo konkretaus programuotojo pirmenybių nusistatymo. Kadangi „Redis“ duomenų bazė užima vietą saugyklos atmintyje, ji yra labiau tinkama laikyti santykinai mažiems duomenų rinkiniams. Jeigu tikslas yra laikyti didžiulius duomenų rinkinius, tai balansas tarp greitaveikos ir kainos greičiausiai baigtųsi suteikiant pirmenybę kainai. „Redis“ duomenų bazė dažniausiai naudojama greitam duomenų išsaugojimui laikinojoje atmintyje. [27]

Jeigu užklausų kūrimas užima didžiąją darbo dalį, patogiau naudoti „Redis“ duomenų bazę, nes joje galima naudoti keletą skirtingų duomenų struktūrų, kurios atitiks rašomas užklausas. Lygiai tas pats galioja ir „MongoDB“ duomenų bazei. Kita vertus naudojant „Redis“ visada reikia papildomo darbo, kad šios duomenų bazės greitaveika būtų puiki. „MongoDB“ duomenų bazė specialistams, kurie anksčiau yra susidūrę su SQL programavimo kalba, yra suprantama labai greitai. O „Redis“ yra netradicinė duomenų bazė ir darbuotojams nėra taip lengva iš karto pradėti ja naudotis, tačiau ji pasižymi labai dideliu lankstumu. Duomenims, kurių išsaugojimas laikinojoje atmintyje yra svarbus, geriau rinktis naudoti „Redis“, tačiau duomenims, kuriems reikia apdoroti daugiau schemų – geriau būtų pasirinkti „MongoDB“ duomenų bazę. Dėl galimybės išsaugoti didelį kiekį naudojantis lanksčia duomenų saugojimo schema, nestruktūrizuotų duomenų apdorojimu viename dokumente, šiame rašto darbe, užklausų rezultatų išsaugojimui, buvo pasirinkta būtent „MongoDB“ duomenų bazė. [28]

6.3. „Duomenų revizoriaus“ ir „QuerySurge“ įrankių palyginimas

„QuerySurge“ – tai komercinis įrankis, kuriuo galima atlikti duomenų palyginimą nenaudojant SQL programavimo kalbos. Tačiau šiame darbe analizuojamu atveju duomenys yra pusiau struktūrizuoti, todėl užklausų rašymas naudojantis SQL kalba yra neišvengiamas. „Duomenų revizoriaus“ įrankyje duomenų užklausos taip pat rašomos SQL kalba. „QuerySurge“ įrankiu galima sukurti diagramas ar grafikus, kurie vaizduoja duomenų testavimo rezultatus, to atlikti naudojantis „Duomenų revizoriaus“ programa kol kas nėra galimybės. Naudotis komercine programa gali keletas vartotojų iš karto, minusas tik tas, kad kiekvienam vartotojui reikia pirkti atskiras įrankio licencijas, kurios yra labai brangios. „Duomenų revizoriaus“ programa taip pat gali naudotis keletas vartotojų iš karto, bet tai galima daryti tik rašant naujus užklausų testus ir juos saugojant. Užklausų vykdymo servisas veikia vietiniame kompiuteryje, todėl jį įjungti vienu metu gali tik vienas žmogus, nes priešingu atveju „MongoDB“ duomenų bazėje pradedami saugoti susidubliavę užklausų rezultatai, kurie vėliau negali būti tinkamai palyginami. „QuerySurge“ testavimo įrankyje galima konfigūruoti vartotojus ir jų teises, duomenų saugyklų parametrus bei prisijungimus prie jų – visa tai galima atlikti naudojantis patogia vartotojo sąsaja, kuri yra labai lengvai suprantama didelės patirties neturintiems IT darbuotojams. „Duomenų revizoriaus“ programa priešingai – neturi savo vartotojo sąsajos, todėl šia programa gali naudotis su jos veikimo principais supažindinti darbuotojai.



20 pav. „QuerySurge“ įrankio vartotojo sąsajos pavyzdys

Naudojantis „QuerySurge“ įrankiu galima sukurti skirtingų testų vykdymo tvarkaraštį. Pavyzdžiui, jeigu vykdomos užklausos yra didelės ir labai stipriai padidina duomenų bazės apkrovą, tai galima sukurti tvarkaraštį, kad tokie testai būtų vykdomi naktį, kai duomenų bazės apkrova yra mažesnė ir pan. Lygiai tą patį galima padaryti ir naudojantis „Duomenų revizoriaus“ įrankiu. Apibendrinant, „QuerySurge“ įrankis yra žymiai patogesnis naudoti dėl to, kad jam yra sukurta vartotojo sąsaja ir juo naudojantis galima patogiau analizuoti klaidas stebint jas grafikuose. Tačiau, didelių duomenų kiekių vientisumo testavimui atlikti buvo pasirinkta specialiai tam sukurta „Duomenų revizoriaus“ programa, kuri yra nemokamas įrankis ir atitinka visus duomenų vientisumo testavimui atlikti keliamus reikalavimus, o atsiradus naujiems reikalavimams įrankis gali būti plėtojamas toliau.

Išvados

Atlikus „Hadoop“ ir „HP Vertica“ duomenų bazių analizę, buvo išsiaiškinta, kad „HP Vertica“ duomenų bazėje užklausos įvykdomos daug greičiau nei „Hadoop“ duomenų bazėje. Taip yra todėl, kad „HP Vertica“ yra SQL duomenų bazė, kurioje visi duomenys yra saugomi stulpeliuose, o „Hadoop“ yra „NoSQL“ duomenų bazė, kurios duomenys yra saugojami HDFS failų sistemoje. Užklausos vykdymo metu duomenų gavimui yra naudojamas „MapReduce“ metodas.

Išsiaiškinus pagrindinius nagrinėjamų duomenų bazių skirtumus, buvo pradėta analizuoti duomenų vientisumo testavimo sprendimus jose. Užklausų atimties ir duomenų atrinkimo testavimo metodai, apžvelgti 4 skyriuje, buvo atmesti, nes jie yra skirti dirbti su daug mažesniais duomenų kiekiais. Taip pat buvo apžvelgtas komercinis įrankis „QuerySurge“, kuris yra vienas iš populiariausių įrankių skirtų didelių duomenų kiekių testavimui atlikti. Tačiau jis taip pat nebuvo pasirinktas, nes tai yra labai brangus komercinis įrankis kainuojantis apie 9 tūkstančius dolerių vienam vartotojui.

Pagrindinis šio darbo tikslas – rasti sprendimą didelių duomenų kiekių testavimui „Hadoop“ ir „Vertica“ duomenų bazėse, buvo įgyvendintas sukūrus naują testavimo įrankį „Duomenų revizorius“, kuris atitinka šiuos testavimo kriterijus:

- Turi galimybę testuoti tiek istorinius duomenis už labai didelį laikotarpį, tiek kasdien gaunamus naujus duomenis. Šis kriterijus buvo įgyvendintas sukurtame įrankyje padarius galimybę keisti testuojamų duomenų laiko intervalus. Patys naujausi, šios dienos, duomenys gali būti testuojami dienos dažnumu, o istorinius duomenis galima testuoti kiek abstrakčiau – pasirinkus savaitės ar mėnesio testavimo laikotarpį.
- Turi galimybę ištrinti testavimo rezultatus ir vykdyti užklausas iš naujo. Tai atlikti labai paprasta naudojantis „MongoDB“ dokumentų duomenų baze. Ši duomenų bazė yra pasirinktas trečios šalies įrankis, skirtas užklausoms ir jų rezultatams išsaugoti, kad vėliau būtų galima atlikti duomenų palyginimą tarpusavyje. „MongoDB“ įrankis buvo pasirinktas dėl galimybės išsaugoti didelį duomenų kiekį naudojantis lanksčia duomenų saugojimo schema, nestruktūrizuotų duomenų apdorojimu viename dokumente.
- Turi galimybę nurodyti vykdomų užklausų kiekį kiekvienoje duomenų bazėje atskirai. Tai yra reikalinga tam, kad užklausos, skirtos atlikti duomenų testavimui klientinėje aplinkoje, nepadidintų darbo apkrovos labai stipriai ir klientai to nepajautų.

Sukurta „Duomenų revizoriaus“ programa buvo palyginta su komerciniu įrankiu „QuerySurge“. Pastarasis įrankis turi vartotojo sąsają, dėl kurios juo yra daug patogiau naudotis tiek labai patyrusiems, tiek mažiau patirties turintiems specialistams. Be to, šis įrankis turi galimybę kurti diagramas bei grafikus, kuriais naudojantis daug paprasčiau stebėti testavimo rezultatus. Tačiau ši programa „Duomenų revizoriaus“ programa neturi sukurtos vartotojo sąsajos, taip pat neturi ir galimybės kurti diagramų ar grafikų. Tačiau, „Duomenų revizoriaus“ programa puikiai atlieka didelių duomenų kiekių vientisumo testavimą ir yra specialiai tam sukurta programa, kuri yra nemokama ir atitinka visus duomenų vientisumo testavimui atlikti keliamus reikalavimus, o atsiradus naujiems reikalavimams įrankis gali būti plėtojamas toliau.

Pagrindinis šio sprendimo privalumas yra žmogiškųjų resursų kaštų sumažinimas iki minimumo, didelių duomenų kiekių testavimo efektyvumo padidinimas ir galimybė ištestuoti visus duomenis, o ne tik tam tikras duomenų dalis. Galimybė atsirasti kartotiniams užklausų rezultatams yra pagrindinis šios programos trūkumas. „Duomenų revizorius“ atlieka užklausų vykdymą atitinkamose duomenų bazėse, o vėliau automatiškai palygina gautus rezultatus ir testą pažymi, kaip „sėkmingą“ arba „klaidingą“. Naudojantis šiuo įrankiu kas dieną yra testuojami milijardai skirtingų duomenų tranzakcijų. Šis įrankis yra sėkmingai taikomas įmonės UAB „Adform Lithuania“ veikloje, duomenų kokybės užtikrinimui gerinti.

Literatūros sąrašas:

1. Kaushal A. 2014. *Big Data: Practical Tips for ETL Testing* [interaktyvus]. [žiūrėta 2015-01-04]. Prieiga per internetą: <<http://www.developer.com/db/big-data-practical-tips-for-etl-testing.html>>.
2. V. Mayer-Schonberger; K. Cukier. 2013. *Big Data. A Revolution That Will Transform How We Live, Work, and Think*. USA. 244p. ISBN: 978-0-544-00269-2
3. A. McAfee; E. Brynjolfsson. 2011. *Big Data: The Management Revolution* [interaktyvus]. [žiūrėta 2016-05-03]. Prieiga per internetą: <http://www.rosebt.com/uploads/8/1/8/1/8181762/big_data_the_management_revolution.pdf>.
4. S. Mills; S. Lucas; L. Irakliotis; M. Rappa; T. Carlson; B. Perlowitz. 2004. *Demystifying Big Data. A Practical Guide To Transforming The Business of Government* [interaktyvus]. [žiūrėta 2015-01-04]. Prieiga per internetą: <<https://www-304.ibm.com/industries/publicsector/fileserv?contentid=239170>> .
5. MongoDB. *What Is Big Data?* [interaktyvus]. [žiūrėta 2015-01-07]. Prieiga per internetą: <<https://www.mongodb.com/big-data-explained>>.
6. World Wide Technology. 2014. *Emerging Big Data Technologies how to identify what you need and integrate with your Legacy Architecture* [interaktyvus]. [žiūrėta 2015-01-09]. Prieiga per internetą: <<https://www2.wwt.com/wp-content/uploads/2015/03/Brief-Emerging-Big-Data-Technologies.pdf>>.
7. Frank L. 2016. *Big Data Technology. What is Hadoop? What is MapReduce? What is NoSQL?* [interaktyvus]. [žiūrėta 2016-04-12]. Prieiga per internetą: <<https://datajobs.com/what-is-hadoop-and-nosql>>.
8. T. White. 2012. *Hadoop. The Definitive Guide*. USA. 659 p. ISBN: 978-1-449-31152-0
9. Gordon S. 2011. *Hadoop and NoSQL Technologies and the Oracle Database* [interaktyvus]. [žiūrėta 2015-02-05]. Prieiga per internetą: <<http://www.oracle.com/technetwork/database/hadoop-nosql-oracle-twp-398488.pdf>>.
10. IBM. *An example of what a Hadoop Cluster infrastructure diagram might look like* [interaktyvus]. [žiūrėta 2015-02-05]. Prieiga per internetą: <<http://www-01.ibm.com/software/ebusiness/jstart/hadoop/>>.
11. E. Capriolo; D. Wampler; J. Rutherglen. 2012. *Programming Hive*. USA. 329 p. ISBN: 978-1-449-31933-5
12. IBM. *What is Hive?* [interaktyvus]. [žiūrėta 2016-04-05]. Prieiga per internetą: <<https://www-01.ibm.com/software/data/infosphere/hadoop/hive/>>.
13. Hewlett-Packard Development Company, L.P. 2014. *HP Vertica Analytics Platform Overview* [interaktyvus]. [žiūrėta 2016-04-08]. Prieiga per internetą: <<http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA5-8917ENW.pdf>>.
14. Hewlett-Packard Development Company, L.P. 2016. *Vertica for SQL on Apache Hadoop Advanced ANSI SQL analytics for your enterprise* [interaktyvus]. [žiūrėta 2016-04-08]. Prieiga per internetą: <<http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA5-7834ENW.pdf>>.
15. Hewlett-Packard Development Company, L.P. 2012. *Make all your information matter. Hadoop and HP Vertica Analytics Platform* [interaktyvus]. [žiūrėta 2016-04-11]. Prieiga per internetą: <http://www.hp.com/hpinfo/newsroom/press_kits/2012/HPDiscoverFrankfurt2012/HP_Vertica_HadoopAnalyticsPlatform_WhitePaper.pdf>.

16. Hewlett-Packard Development Company, L.P. 2015. *HP Vertica enables faster, more sophisticated market insight* [interaktyvus]. [žiūrėta 2016-04-11]. Prieiga per internetą: <<https://www.hpe.com/h20195/v2/GetPDF.aspx/4AA5-9079ENW.pdf>>.
17. Hinchcliffe D. 2009. *Using Hadoop in the Enterprise* [interaktyvus]. [žiūrėta 2016-04-11]. Prieiga per internetą: <http://www.ebizq.net/blogs/enterprise/2009/09/10_ways_to_complement_the_enterprise.php>.
18. DB-Engines. 2016. *System Properties Comparison HBase vs. Vertica* [interaktyvus]. [žiūrėta 2016-04-11]. Prieiga per internetą: <<http://db-engines.com/en/system/HBase%3BVertica>>.
19. Ashlesha S. N.; Tugnayat R. M.; Tembhurkar M. P.; *Overview on Performance Testing Approach in Big Data* [interaktyvus]. [žiūrėta 2016-04-12]. Prieiga per internetą: <http://www.academia.edu/9883288/Overview_on_Performance_Testing_Approach_in_Big_Data>.
20. Oracle. *Data Integrity* [interaktyvus]. [žiūrėta 2016-04-21]. Prieiga per internetą: <https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch7.htm>.
21. Oracle. 2010. *Data Comparison Methods Overview* [interaktyvus]. [žiūrėta 2016-04-12]. Prieiga per internetą: <<http://planet.mysql.com/entry/?id=23819>>.
22. Real-Time Technology Solutions. 2016. *Minus Queries* [interaktyvus]. [žiūrėta 2016-05-06]. Prieiga per internetą: <<http://www.querysurge.com/solutions/minus-queries>>.
23. Real-Time Technology Solutions. 2016. *Sampling Method of Data Validation* [interaktyvus]. [žiūrėta 2016-05-06]. Prieiga per internetą: <<http://www.querysurge.com/solutions/sampling>>.
24. Real-Time Technology Solutions. 2016. *What is QuerySurge?* [interaktyvus]. [žiūrėta 2016-05-09]. Prieiga per internetą: <<http://www.querysurge.com/product-tour/what-is-querysurge>>.
25. Real-Time Technology Solutions. 2016. *Query Wizards* [interaktyvus]. [žiūrėta 2016-05-09]. Prieiga per internetą: <<http://www.querysurge.com/product-tour/querysurge-query-wizards>>.
26. MongoDB. 2016. *Document Databases* [interaktyvus]. [žiūrėta 2016-05-14]. Prieiga per internetą: <<https://www.mongodb.com/document-databases>>.
27. Redis. *Introduction to Redis* [interaktyvus]. [žiūrėta 2016-05-16]. Prieiga per internetą: <<http://redis.io/topics/introduction>>.
28. Kovacs K. 2016. *MongoDB vs Redis comparison* [interaktyvus]. [žiūrėta 2016-05-19]. Prieiga per internetą: <<http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis>>.