KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Marija Rygalovskaja

RELIACINĖS DUOMENŲ BAZĖS OPTIMIZACIJA GREITAVEIKAI

Baigiamasis magistro projektas

Vadovas
Lekt. Julijus Jakutavičius

KAUNAS, 2018

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

RELIACINĖS DUOMENŲ BAZĖS OPTIMIZACIJA GREITAVEIKAI

Baigiamasis magistro projektas
Informatikos studijų programa (kodas 621|10003)

Vadovas
Lekt. Julijus Jakutavičius

Recenzentas
Doc. Liudas Motiejūnas

Projektą atliko
Marija Rygalovskaja

KAUNAS, 2018

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Marija Rygalovskaja

(Studento vardas, pavardė)

Informatika 621|10003

(Studijų programos pavadinimas, kodas)

Reliacinės duomenų bazės optimizacija greitaveikai

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 18 m. _____ ____ d.

Kaunas

Patvirtinu, kad mano, **Marijos Rygalovskajos**, baigiamasis projektas tema „Reliacinės duomenų bazės optimizacija greitaveikai“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____

(vardą ir pavardę įrašyti ranka) *(parašas)*

Rygalovskaja, Marija. Reliacinės duomenų bazės optimizacija greitaveikai. *Magistro* baigiamasis projektas / vadovas lekt. Julijus Jakutavičius; Kauno technologijos universitetas, Informatikos fakultetas.

Mokslo kryptis ir sritis: Technologijos mokslai, Informatikos inžinerija

Reikšminiai žodžiai: *reliacinė duomenų bazė, optimizacija greitaveikai, denormalizacija, duomenų bazės modelis, užklausos vykdymo laiko vertinimas*.

Kaunas, 2018. 76 p.

SANTRAUKA

Daugelis egzistuojančių programų yra sukurtos arba vis dar yra kuriamos naudojantis reliacinės duomenų bazės valdymo sistemos technologijomis. Reliacinės duomenų bazės dažniausiai būna normalizuotos – lentelės ir laukai yra atitinkamai organizuojami. Šis procesas užtikrina geresnį duomenų vientisumą, tačiau duomenų gavimo metu yra naudojamas kelių lentelių sujungimai (angl. *join*). Sujungimai yra lėta operacija, todėl dažnai duomenų bazė yra denormalizuojama.

Egzistuoja įvairūs denormalizacijos metodai, kurių metu duomenis organizuojami taip, kad būtų lengviau prieinami užklausoms. Dažniausiai jų metu yra mažinamas lentelių, kurios turi būti skenuojamos užklausose, skaičius taip mažinant sujungimų kiekį užklausoje. Tokie metodai yra lentelių sujungimas į vieną, pasikartojančių stulpelių pridėjimas, pasikartojančių laukų pridėjimas bei skaičiuojamųjų laukų saugojimas. Taip pat denormalizacijos proceso metu lentelės gali būti skaidomos, norint sumažinti lentelių dydį.

Nors denormalizacija yra dažnai pritaikoma praktikoje, tačiau šioje srityje vis dar trūksta aiškių strategijų. Egzistuojančiuose tyrimuose denormalizacijos poveikis duomenų bazės veikimui nurodomas tik teoriniu aspektu. Šio darbo esmė yra išbandyti denormalizacijos metodus ir nustatyti jų poveikį. Darbe atlikti 4 eksperimentai, kurių metu buvo matuojami laikai prieš optimizaciją ir po jos. Matavimai įvykdyti su trijų dydžių duomenų imtimis – maža, vidutine ir didele. Apibendrinimai pateikti rezultatų skyriuje.

Rygalovskaja, Marija. *Optimizing Performance Of Relational Database*: *Master's* thesis in Informatics / supervisor lect. Julijus Jakutavičius. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: Technological Sciences, Informatics Engineering

Key words: *relational database, optimizing performace, denormalization, database model, query execution time measurement.*

Kaunas, 2018. 76 p.

SUMMARY

Majority existing programs were created or are still being developed using relational database management system technologies. Relational databases are usually normalized – tables and fields are organized respectively. This process ensures better data integrity, however multiple table joins are used when retrieving data. Join is a slow operation, therefore database is often denormalized.

There are various denormalization methods in which data is organized in such a way that would be easier accessed to queries. Most often number of tables that must be scaned in queries is reduced, thus reducing the number of joins in query. These methods include joining tables into one, adding duplicate columns, adding duplicate fields and storing calculated fields. Also, during denormalization process, the table can be split to reduce table size.

Although denormalization is often applied in practice, there is still a lack of clear strategies in this area. In existing researches the denormalization impact on the operation of the database is indicated only in a theoretical basis. The essence of this work is to test denormalization methods and to determine their impact. 4 experiments were carried out in the work, during which time and other parameters before and after the optimization were measured. Measurements are performed with three data sampling – small, medium and big. Summaries are provided in the results section.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

Denormalizacija	procesas, kurio metu normalizuotoje duomenų bazėje atliekami tam tikri veiksmai, norint pagreitinti duomenų išrinkimą
Duomenų anomalijos	problemos, kurios atsiranda blogai suprojektuotoje ar nenormalizuotoje duomenų bazėje (pavyzdžiui, duomenų nevientisumas)
Duomenų bazės valdymo sistema (*DBVS*)	specializuotos programos skirtos kurti ir organizuoti duomenų bazes
Duomenų elementas	konkreti duomenų reikšmė lentelėje
Išorinins raktas	vienos lentelės laukas, kuris kitoje lentelėje yra pirminis
Metodas-1	denormalizacijos metodas, kurio metu lentelėje saugomi pasikartojantys stulpeliai
Metodas-2	denormalizacijos metodas, kurio metu kelios lentelės sujungiamos į vieną
Metodas-3	denormalizacijos metodas, kurio metu lentelėje saugomi skaičiuojamieji laukai
Metodas-4	denormalizacijos metodas, kurio metu lentelėje saugoma pasikartojanti informacija
Normalinė forma (*NF*)	sąlygos, kurias turi atitikti duomenų bazės reliacinė schema, kad duomenų bazė išvengtų tam tikrų nepageidaujamų savybių
Normalizacija	procesas, kurio metu taikomos įvairios taisyklės, siekiant supaprastinti sąryšius reliacinėje duomenų bazėje
Optimizacija	procesas, kurio metu stengiamasi sumažinti užklausų vykdymo laiką ir pagerinti duomenų bazės greitaveiką
Pirminis raktas	laukai ar laukų grupės, kurios vienareikšmiškai identifikuoja lentelių įrašus
Reliacinė duomenų bazė	duomenų bazė, kurioje informacija saugoma dvimatėse lentelėse
SQL	struktūrinė užklausų kalba, skirta darbui su duomenų bazėmis
Sujungimas (angl. *join*)	*SQL* užklausos sąlyga, kuri reliacinėje duomenų bazėje sujungia stulpelius iš kelių lentelių
Tuščia reikšmė (angl. *null*)	specialus žymėjimas, naudojamas *SQL*, kuriuo parodoma, kad neegzistuoja duomenų reikšmė
Užklausos vykdymo laikas	laiko tarpas, per kurį grąžinamas atsakymas, įvykdžius užklausą

ĮVADAS

Daugelis interneto sistemų naudoja reliacines duomenų bazes. Remiantis *db-engines* 2018 metų gegužės mėnesio duomenimis, reliacinių duomenų bazių populiarumas tarp duomenų bazių kategorijų yra 77,6 % [1]. Šis reitingas sudarytas pagal tai, kaip dažnai duomenų bazė yra minima paieškos rezultatuose ir neatsižvelgiama į tai, koks yra atsiliepimas ar koks yra instaliacijų kiekis. Tačiau tai nekeičia fakto, jog reliacinės duomenų bazės dominuoja duomenų bazių rinkoje.

Nors jų populiarumas didelis, bet reliacinių duomenų bazių greitaveika ne visada pasiteisina. Egzistuoja įvairūs greitaveikos gerinimo metodai: duomenų modelio optimizacija (pavyzdžiui, normalizacija, denormalizacija, indeksų ar procedūrų naudojimas), *SQL* užklausų optimizacija, fizinė optimizacija (konfigūraciniai parametrai, vietos naudojimas, procesoriai). Šiame darbe nagrinėjama duomenų modelio optimizacija – atsižvelgiama į tai, kaip gerinti greitaveiką denormalizuojant duomenų bazę.

Įvairūs duomenų bazės denormalizacijos metodai yra plačiai naudojami bei šaltiniuose pateikiamos skirtingos technikos, tačiau vis dar trūksta aiškių optimizacijos principų. Literatūroje nurodoma, kaip galima denormalizuoti, tačiau egzistuoja mažai tyrimų, kurie iliustruotų, kaip pritaikyti pakeitimai pakeičia duomenų bazės greitaveiką. Šaltiniuose poveikis labiau pateikiamas teoriniu aspektu (kaip denormalizacija turėtų paveikti), o ne praktiniu – nėra aprašyti eksperimentai, iš kurių galima matyti, kaip iš tikrųjų pasikeitė duomenų bazės greitaveika bei kiti parametrai. Šis darbas skirtas parengti išsamias gaires apie tai, kaip ir kokias būdais galima denormalizuoti.

Darbo problematika ir aktualumas

Kadangi vis daugiau vartotojų persikelia į internetinę erdvę, prieinamumas tampa pagrindiniu rūpesčiu. Be to, vartotojai tikisi, kad naudojama sistema veiks greitai. Jie nėra nusiteikęs ilgai laukti atsakymo ir gali išjungti programą, jeigu ji veikia lėčiau negu vartotojas tikisi. Galiausiai, programos ir servisai turi palaikyti vis didėjantį vartotojų ir duomenų kiekį – o didėjant sistemoms ir duomenų kiekiams, atsiranda greitaveikos problema.

Dažniausiai reliacinės duomenų bazės schema būna normalizuota. Tai lemia, kad duomenys lentelėse nesikartoja ir yra suskaidyti į daugelį tarpusavyje susijusių lentelių. Toks duomenų smulkinimas ar perkomponavimas lėtina skaitymo operacijas, kadangi duomenis tenka gauti iš daugybės susijusių lentelių. Jeigu duomenų bazę sudaro daug lentelių, užklausos pasidaro sudėtingos ir ilgos, o juose egzistuoja daug sujungimų (angl. *joins*), kurie daro neigiamą įtaką greitaveikai.

Literatūroje minima, jog normalizuota schema nepasiteisina ir yra pritaikoma denormalizacija, norint pagerinti greitaveiką. Įvairiose šaltiniuose yra nurodyti skirtingi denormalizacijos metodai. Tačiau šios technikos nėra teoriškai pagrįstos – dauguma jų išvestos iš praktinės patirties projektuojant duomenų bazes. Tai lemia, kad dažniausiai duomenų bazės pokyčiai yra aprašomi tik teoriniu aspektu, o praktinio iliustravimo nėra. Be to, aprašuose trūksta detalumo.

Taigi, egzistuoja daug įvairių denormalizacijos metodikų, tačiau tyrimų šioje srityje stinga. Informacija yra pateikiama nepagrįstai – poveikiai nustatomi teoriškai, o praktinių bandymų beveik nėra. Todėl šiuo darbu siekiama išanalizuoti keletą denormalizacijos metodų ir matuojant duomenų bazės parametrus – įvertinti, ar metodas pasiteisina. Tai gali padėti susisteminti ir išbandyti optimizacijos metodus.

Darbo tikslas ir uždaviniai

Darbo tikslas – atlikti reliacinės duomenų bazės modelio efektyvumo optimizavimo tyrimą.

Uždaviniai:

1. išanalizuoti esamus reliacinės duomenų bazės modelio greitaveikos optimizavimo metodus;
2. ištirti denormalizacijos įtaką duomenų operacijoms reliacinėje duomenų bazėje;
3. atlikti optimizacijos tyrimą, pritaikant ištirtus metodus;
4. apibendrinti tyrimo rezultatus ir palyginti duomenų bazės dydį bei greitaveiką prieš ir po optimizacijos.

Darbo struktūra

Šiame darbe yra nagrinėjami galimi reliacinės duomenų bazės modelio optimizacijos būdai, kurie pagerintų jos greitaveiką. Visų pirma yra pateikiama analizės dalis – informacija apie tyrimo objektą, įvairias reliacinės duomenų bazės modelio optimizavimo technikas bei nurodomas galimų reliacinės duomenų bazės optimizacijos denormalizuojant metodų sąrašas. Antrame skyriuje pateikiama projektavimo eiga. Šiame skyriuje sudaroma optimizacijos metodika ir vizualiai iliustruojamos denormalizacijos technikų schemos. Sudarytų metodikų realizavimo eiga nurodyta trečiame skyriuje. Ketvirtame skyriuje atliekami matavimai – pateikiamos metrikos prieš ir po optimizacijos. Penktame skyriuje iliustruojami rezultatai – pateikiami kiekvieno metodo rezultatai bei apibendrinami eksperimentų skyriuje pateikti matavimai. Galiausiai, šeštoje dalyje apibendrinamas visas darbas ir pateikiamos išvados.

1. DUOMENŲ BAZĖS OPTIMIZACIJOS GREITAVEIKAI ANALIZĖ

Egzistuoja įvairios duomenų bazės optimizacijos technikos – fizinė, duomenų modelio bei *SQL* kodo optimizacija. Fizinės duomenų bazės optimizacija apima techninės įrangos resursų naudojimą, tinklą (angl. *networking*) bei kitas administracines užduotis (tokias, kaip konfigūracija bei failų paskirstymas). Ši technika apima tai, kaip duomenų bazė yra instaliuojama ir konfigūruojama. G. Powell [2] nurodė, kad ši optimizacija dažniausiai pagerina našumą nuo 10 % iki 20 %. Fizinė technika yra standartizuota, todėl sprendimas yra lengviau realizuojamas nei kitos optimizacijos [2, 3].

SQL kodo optimizacija susijusi su užklausų rašymo ypatumais. Tokia technika labai priklauso nuo programos. Prastai sudarytos *SQL* užklausos sukelia daug našumo problemų. Pasak G. Powell [2] tokia optimizacija gali pagerinti greitaveiką daugiau nei 25 procentais [2, 3].

Duomenų modelio optimizacijoje keičiama pati schema. Į šią techniką įeina normalizacija, denormalizacija, indeksų ar procedūrų naudojimas. Duomenų bazės greitaveika labai priklauso nuo duomenų modelio. Schemos pakeitimai gali paveikti visą sistemą bei *SQL* kodą. Dėl šios priežasties duomenų modelio optimizacija yra sudėtingiausias ir brangiausias sprendimas. Reliacinės duomenų bazės, normalizacijos bei denormalizacijos sąvokos yra gerai žinomos, tačiau denormalizacijos principai nėra aiškūs ir plačiai išanalizuoti [2, 3].

Šiame darbe nagrinėjama duomenų modelio optimizacija. Analizuojant literatūros šaltinius pastebėta, jog problematiški yra sujungimai užklausose. Juos siūloma mažinti denormalizuojant duomenų bazę. Dėl šios priežasties šiame skyriuje analizuojami įvairūs denormalizacijos metodai.

Prieš kalbant apie optimizaciją, skyriuje pateikiamos pagrindinės reliacinės duomenų bazės sąvokos. Kadangi dažniausiai denormalizacija yra priešingas procesas normalizacijai, todėl šiame skyriuje taip pat kalbama apie normalizaciją. Literatūroje rasti optimizavimo sprendimai pateikiami po to, kai nurodomas normalizacijos bei denormalizacijos procesas ir jie tarpusavyje palyginami.

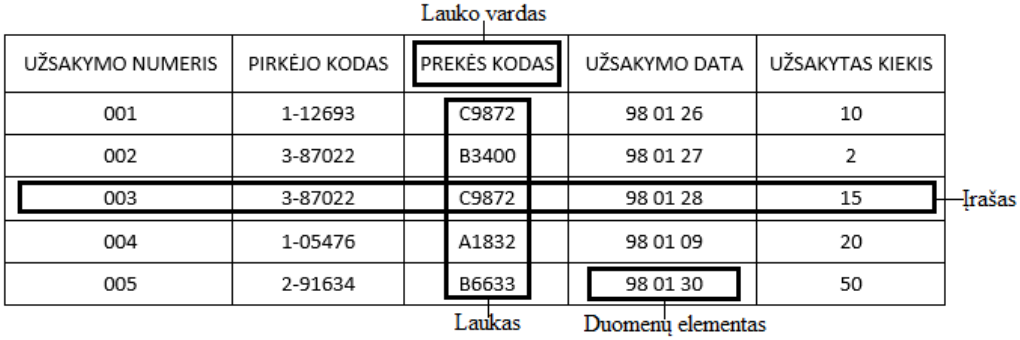
1.1 Reliacinės duomenų bazės analizė

Šiame poskyryje visų pirma apibrėžiama reliacinė duomenų bazė. Pateikiami tokie terminai, kaip duomenų bazės lentelė, duomenų bazės raktai, ryšiai tarp lentelių. Poskyryje taip pat nagrinėjamas normalizacijos ir denormalizacijos procesai bei yra tarpusavyje palyginami.

1.1.1. Reliacinės duomenų bazės apibrėžimai

Reliacinė duomenų bazė – tai tokia duomenų visuma, kurioje informacija saugoma dvimatėse lentelėse. Lentelė yra vienodo tipo įrašų rinkinys. Reliacinės duomenų bazės bazinis komponentas yra vadinamas stulpeliu arba lauku, kuris turi prasmę realiame pasaulyje (pavyzdžiui, pavardė). Susijusių stulpelių grupė, kuri traktuojama kaip vienas dydis, yra eilutė (arba įrašas). Kiekvienas įrašas yra identifikuojamas pagal lauką (atributą), kuris turi unikalią reikšmę. Įrašo pavyzdžiai – klientas, produktas ar užsakymas. Stulpelių ir eilučių susikirtimuose yra konkrečios duomenų reikšmės (duomenų elementai) [4-6].

Anksčiau aptartos reliacinės duomenų bazės sąvokos iliustruojamos 1.1 pav. Šiame paveiksle pateikta pavyzdinė užsakymo lentelė. Joje viena eilutė atitinka vieną objektą (šiuo atveju – vieną užsakymą), o stulpeliai – objektų reikšmes, pagal ką užsakymas identifikuojamas.



1.1 pav. Pavyzdinė dvimatė reliacinės duomenų bazės lentelė su sąvokomis [5]

Reliacinės duomenų bazės lentelėms yra nustatomi raktai – laukai ar laukų grupės, kurių reikšmės yra nepasikartojančios. Šios reikšmės vienareikšmiškai identifikuoja tų lentelių įrašus. Toks stulpelis ar jų grupė yra vadinamas pirminiu raktu (angl. *primary key*) [4, 5].

Eilutės vienoje lentelėje gali būti susietos su eilutėmis kitoje lentelėje. Sąryšį tarp atskirų lentelių nustato bendri, sutampantys atitinkamų lentelių laukai. Pagrindinėje lentelėje šis stulpelis (ar laukas) vadinamas pirminiu raktu, o kitoje lentelėje (toje, su kuria susijusi reikšmė) – išoriniu raktu (angl. *foreign key*) [4, 5].

Dvi lentelės (A ir B) tarpusavyje gali būti susijusios trimis būdais: vienas su vienu (1:1), vienas su daug (1:N) ir daug su daug (M:N). Jeigu A lentelės pirminio rakto nurodyta reikšmė atitinka tik vieną įrašą B lentelėje, tuomet ryšys yra 1:1. Jeigu A lentelės pirminio rakto nurodyta reikšmė atitinka vieną ar daugiau reikšmių lentelėje B, tuomet lentelės susijusios 1:N ryšiu. Jeigu A lentelės pirminio rakto nurodyta reikšmė atitinka bent vieną reikšmę B lentelėje ir atvirkščiai, tuomet lentelės susijusios M:N ryšiu [7, 8].

Reliacinės duomenų bazės modelis gali būti dviejų tipų [9]:

- normalizuotas. Normalizacija yra žingsnių seka, kuria remiantis reliacinės duomenų bazės modelis yra sukuriamas ir modifikuojamas. Šio proceso tikslas yra sumažinti duomenų kiekį ir jų dubliavimąsi. Be to, yra stengiamasi sumažinti galimybę turėti nesuderinamumą tarp duomenų. Plačiau apie normalizaciją aprašoma poskyryje Reliacinės duomenų bazės normalizacija;
- denormalizuotas. Denormalizacija yra priešingas procesas normalizacijai – dažniausiai jos metu yra atkuriami kai kurie duomenų dublikatai, anuliuojamos anksčiau pritaikytos normalinės formos. Šis procesas padidina riziką turėti nevientisus duomenis. Denormalizacija dažniausiai realizuojama norint pagerinti greitaveiką. Plačiau apie denormalizaciją aprašoma poskyryje Reliacinės duomenų bazės denormalizacija.

1.1.2. Reliacinės duomenų bazės normalizacija

Normalizacija yra procesas, kurio metu mažinamas duomenų dubliavimasis duomenų bazėje. Šis procesas dažniausiai apima lentelių padalinimą į mažesnes (ir mažiau dublikatų turinčias) lenteles ir sąryšių tarp jų nustatymą. Tikslas yra izoliuoti duomenis taip, kad įrašų įterpimas, šalinimas ir atnaujinimas būtų atliekamas tik vienoje lentelėje, o pakeitimai perduodami likusioms lentelėms per nustatytus ryšius. Normalizacija susistemina duomenis į stabilias struktūras, taip sumažinant duomenų anomalijų tikimybę ir padidinant duomenų prieinamumą (ang. *accessibility*) [10, 11].

Taisyklės, užtikrinančios optimalų duomenų bazės projektavimą (be duomenų dubliavimosi), vadinamos normalinėmis formomis. Jos numato būdus, kaip galima sugrupuoti atributus. Pirmoji normalinė forma (1NF), antroji normalinė forma (2NF) ir trečioji normalinė forma (3NF) yra dažniausiai naudojamos formos. Aukštesnės normalizavimo formos (Boyce / Codd normalinė forma, ketvirta normalinė forma (4NF) ir penktoji normalinė forma (5NF)) yra svarbios akademiniu požiūriu, tačiau nėra plačiai taikomos praktikoje [10]. Dėl šios priežasties šiame darbe apžvelgiamos tik pirmosios trys normalinės formos.

Normalizacija vykdoma etapais – nuo pirmosios normalinės formos iki norimos. Normalizacijos lygis pasirenkamas individualiai. Kuo aukštesnė normalinė forma pritaikoma, tuo labiau mažėja duomenų anomalijų tikimybė, atsiranda galimai sudėtingesnis sistemos palaikymas ir sistemos greitaveika žymiai pablogėja. Bendru atveju, dažniausiai sistemose normalizacijos lygis būna trečios normalinės formos. Būtent jis skaitomas optimaliausiu [7, 12, 13].

Pirma normalinė forma yra bazinė. Ši taisyklė reikalauja pašalinti pasikartojančių duomenų reikšmes ir nurodo, kad jokios dvi eilutės lentelėje negali būti identiškos. Vadinasi, kiekviena lentelė turi turėti loginį pirminį raktą, kuris unikaliai identifikuoja eilutę lentelėje. Kiekvieno stulpelio reikšmė turi būti atominio vieneto (mažiausios įmanomos komponentės). Kitaip sakant, kiekvienos lentelės lauke yra galima tik viena reikšmė, o ne jų rinkinys (pavyzdžiui, masyvas) [5, 11, 12, 14, 15].

1.2 pav. kairėje pusėje pateikta lentelė nėra pirmos normalinės formos. Eilutėse negali būti stulpelių, kuriose saugoma daugiau nei viena reikšmė (pavyzdžiui, reikšmės, kurios išskirtos kableliais). Šiame pavyzdyje *Subject* lauke yra saugoma tokia reikšmė (*Biology, Maths*). Susidarius

nurodytai situacijai, tokius duomenis reikia išskirti į atskiras eilutes. Perdaryta į pirmąją normalinę formą lentelė pateikta 1.2 pav. dešinėje pusėje [14].

STUDENT	AGE	SUBJECT
Adam	15	Biology, Maths
Alex	14	Maths
Stuart	17	Maths

STUDENT	AGE	SUBJECT
Adam	15	Biology
Adam	15	Maths
Alex	14	Maths
Stuart	17	Maths

1.2 pav. Pakeitimai po pirmosios normalinės formos pritaikymo [14]

Lentelė yra antrosios normalinės formos, jeigu ji yra 1NF ir kiekvienas jos neraktinis laukas pilnai priklauso nuo viso pirminio rakto. Jeigu stulpelis priklauso tik nuo pirminio rakto dalies ar nuo kito stulpelio, kuris nepriklauso pirminiam raktui, tuomet lentelė neatitinka antrosios norminės formos. Šios taisyklės esmė yra duomenų, kurie yra iš dalies priklausomi nuo pirminio rakto, perkėlimas į kitą lentelę [5, 11, 12, 14, 15].

1.2 pav. dešinėje pusėje esančioje lentelėje pateiktos dvi eilutės, kurių stulpelio *Student* reikšmė sutampa (*Adam*). Toks sprendimas neefektyviai išnaudoja vietą. Be to, šios lentelės galimas pirminis raktas būtų {*Student, Subject*}. Tačiau *Age* stulpelis priklauso tik nuo pirminio rakto dalies (*Student*). Tai neatitinka antrosios norminės formos taisyklių. Norint turėti antrąją norminę formą, šią lentelę reikia padalinti į dvi – vienoje saugant studento informaciją (1.3 pav. a)), o kitoje atvaizduojant kursus (1.3 pav. b)). Šiuo atveju studento lentelėje *Student* stulpelis yra raktas, pagal kurį šios lentelės susijungia. Kurso lentelėje, pirminis raktas yra {*Student, Subject*} pora [14].

STUDENT	AGE
Adam	15
Alex	14
Stuart	17

a)

STUDENT	SUBJECT
Adam	Biology
Adam	Maths
Alex	Maths
Stuart	Maths

b)

1.3 pav. Išskaidytos lentelės pagal antrąją normalinę formą [14]

Lentelė yra trečiosios normalinės formos, jeigu ji yra 2NF ir kiekvienas neraktinis stulpelis yra netranzityviai priklausomas nuo pirminio rakto. Tranzityvus priklausymas reiškia, kad iš stulpelio A galima gauti B, iš kurio tranzityviai galima gauti C atributo reikšmę. Tokiu būdu ši taisyklė nurodo, kad lentelėje neturi būti jokių funkcinių priklausomybių tarp neraktinių stulpelių. Neraktiniai stulpeliai turi priklausyti tik nuo raktų. Jei taip nėra, turi būti sukurta nauja lentelė, kuri neturėtų neraktinių laukų tarpusavio priklausomybės. Šioje formoje atributai, kurie nurodo skirtingus objektus, dažniausiai yra perkeliami į skirtingas lenteles [5, 12, 14, 15].

1.4 pav. kairėje pavaizduota pavyzdinė lentelė. *Student_id* yra pirminis raktas, tačiau *street, city* ir *state* priklauso nuo *Zip*. Norint šiai lentelei pritaikyti trečiąją normalinę formą, *street, city* ir *state* laukai yra perkeliami į naują lentelę, kurioje *Zip* yra pirminis raktas (1.4 pav. dešinėje).

STUDENT_ID	STUDENT_NAME	DOB	STREET	CITY	STATE	ZIP

STUDENT_ID	STUDENT_NAME	DOB	ZIP

ZIP	STREET	CITY	STATE

1.4 pav. Pakeitimai po trečiosios normalinės formos pritaikymo [14]

Norint sistemai parinkti normalizacijos lygį, reikia įvertinti jos paskirtį. Tai galima daryti, įvertinant, ar duomenų bazė dažniausiai naudojama duomenų gavimo, ar duomenų apdorojimo veiklai. Esant duomenų apdorojimo sistemoms, normalizacija yra būtina norint užtikrinti duomenų vientisumą ir pagreitinti duomenų įterpimo, atnaujinimo ir šalinimo operacijas. Priešingai, duomenų apdorojimo sistemose skaitymo našumas yra svarbiausias ir tokiose situacijose dažniausiai yra pritaikoma mažesnio laipsnio normalizacija. Todėl sistemoms, kuriose dažnai vykdomos rašymo operacijos, geriau turėti aukštesnį normalizacijos lygį, o sistemoms, kuriose pagrindinis veiksmas yra duomenų gavimas, – mažesnį [16].

Kalbant apie normalizaciją ir duomenų bazės optimizaciją, atsižvelgiama į keletą aspektų [9]:

- per mažas normalizacijos lygis lemia per didelį duomenų dubliavimąsi. Dėl šios priežasties duomenų bazė yra per didelė. Per didelis duomenų kiekis reiškia lėtą priėjimo (angl. *access*) laiką vien todėl, kad disko vieta, kurioje galima ieškoti informacijos, yra per didelė;
- netinkama normalizacija gali reikšti nepasiekiamus duomenis ar neteisingus ryšius;
- per didelis normalizacijos lygis veda prie pernelyg sudėtingo (angl. *overcomplex*) *SQL* kodo, kurį sunku ar visai neįmanoma optimizuoti;
- jeigu schema yra pernelyg normalizuota, duomenų bazę sudaro daug mažų lentelių. Dėl šios priežasties užklausose reikia naudoti sujungimus, kurie neigiamai veikia greitaveiką.

Normalizacijos privalumai [11, 12, 17]:

- besidubliuojančių duomenų mažinimas. Tai supaprastina duomenų struktūras ir mažina duomenų bazės dydį (taip pat saugo disko vietą);
- dėl duomenų dubliavimosi pašalinimo, normalizacija gali lemti greitą lentelės skenavimą ir duomenų paiešką;
- lentelės yra mažesnės, todėl indeksų kūrimas ir duomenų rūšiavimas vyksta greičiau;
- lentelės yra siauresnės. Taip galima išsaugoti daugiau eilučių puslapyje;
- duomenų integralumo ir vientisumo užtikrinimas duomenų bazėje. Dėl panaikintų duomenų dublikatų, išlaikomas duomenų vientisumas;
- greičiau vykdoma duomenų atnaujinimo, įterpimo ir šalinimo operacijos. Nes atitinkama informacija yra saugoma vienoje lentelėje, todėl šie veiksmai atliekami tik joje.

Trūkumai [12, 13, 17]:

- duomenų bazėje egzistuoja daugiau lentelių. Išskaidant lentelę į kelias, kiekvienoje papildomai atsiranda stulpelis (-iai) ryšiams sudaryti. Be to, kiekviena lentelė papildomai turi antraštę (angl. *overhead*). Todėl, nors ir dublikatų šalinimas mažina duomenų bazės dydį, bet papildomos lentelės didina duomenų bazė užimą vietą;
- sudėtingesnis duomenų bazės projektavimas;
- užklausos tampa sudėtingesnės (duomenis reikia išrinkti iš daugelio lentelių);
- sujungimų naudojimas užklausose. Tokios užklausos vykdymas reikalauja daugiau laiko. Be to, indeksavimo strategijos nesiderina su sujungimais.

Apibendrinant, normalizacija išlaiko duomenų vientisumą. Tačiau jos metu lentelės išskaidomos į daugiau lentelių. Kuo daugiau lentelių, tuo daugiau sujungimų užklausose, o sujungimai turi neigiamą įtaką greitaveikai. Be to, normalizacijos pritaikymas gali sukurti tokias situacijas, kai bandant gauti mažą duomenų kiekį tenka ieškoti informacijos daugelyje lentelių. Normalizuota schema labiau tinka rašymo operacijoms, o ne skaitymo. Taigi, normalizacija turi vieną didelį trūkumą – prastą sistemos greitaveiką skaitymo operacijoms. Dėl šios priežasties, dažnai normalizacijos lygmuo yra mažinamas [10]. Kitame poskyryje ir kalbama apie normalizacijos lygio mažinimo procesą – denormalizaciją.

1.1.3. Reliacinės duomenų bazės denormalizacija

Denormalizacija yra procesas, kai normalizuotoje duomenų bazėje yra modifikuojama lentelių struktūra, stengiantis pagerinti duomenų bazės greitaveiką. Tai galima apibūdinti kaip normalizacijos lygio mažinimo procesą. Denormalizacijos proceso metu gali būti išskaidytos normalizacijos metu

sujungtos lentelės ar sukuriami duomenų dublikatai. Tai daroma tam, kad būtų sumažintas lentelių kiekis bei sujungimų poreikis užklausose [9-11].

Denormalizacija yra realizuota daugelyje duomenų bazių tam, kad būtų pagerinta greitaveika ir būtų sumažintas užklausų vykdymo laikas. Šis procesas dažniausiai naudojamas tokiose sistemose, kuriose duomenų skaitymas, gavimas, perskaičiavimas yra pagrindiniai veiksmai. Taip pat denormalizacija pasitarnauja tokiose situacijose, kuriose reikia pasiekti tokius agregatus (angl. *aggregates*), kaip sumos, vidurkiai, minimumai, maksimumai. Taip yra todėl, jog normalizuotoje sistemoje saugomi tik baziniai duomenys, o pastaruoju atveju – skaičiuojami laukai jau yra paruošti skubiam pateikimui, todėl tokių duomenų gavimas yra greitesnis [10, 17, 18]. Tokiu būdu, denormalizacija yra ne tik normalinių formų anuliavimas (pavyzdžiui, lentelių jungimas ar dublikatų įvedimas), bet ir procesas, kuris nėra tiesiogiai nesusijęs su normalinėmis formomis (skaičiuojamų laukų saugojimas) [19].

Egzistuoja įvairūs denormalizacijos būdai. Tam tikri privalumai ir trūkumai priklauso nuo atitinkamo metodo, tačiau galima išskirti bendro pobūdžio savybes. Pagrindinis denormalizacijos privalumas yra greitaveikos gerinimas. Pritaikius denormalizaciją, dažniausiai yra sumažinamas sujungimų kiekis užklausose, pamažėja lentelių kiekis duomenų bazėje, greičiau atvaizduojami skaičiuojami laukai, sumažėja išorinių raktų kiekis sąryšiuose ar sumažėja sąryšių kiekis tarp lentelių.

Denormalizacija turi ir trūkumų [17, 20]:

- padidėjusi disko talpa. Dėl duomenų dublikatų, duomenų bazė užima daugiau vietos;
- atsiradusios duomenų anomalijos. Po denormalizacijos, dėl atsiradusių duomenų dublikatų, duomenis tenka keisti daugiau negu vienoje vietoje. Taip duomenys gali išsikraipyti;
- lėtesnės kitos operacijos. Neigiamai paveikiamos duomenų įterpimo, modifikavimo ir šalinimo operacijos. Jeigu šios operacijos atliekamos pakankamai retai, denormalizacija yra teigiama;
- daugiau programavimo. Dėl pasikeitusios struktūros, turi keistis ir programos kodas.

Be normalizacijos tampa sunku valdyti ir modifikuoti duomenų bazę, neprarandant duomenų. Įterpimo, atnaujinimo ir šalinimo anomalijos yra dažnos, jeigu duomenų bazė nėra normalizuota arba yra pritaikyta denormalizacija. Egzistuoja tokios anomalijos [13, 14, 21]:

- atnaujinimo. Tokio tipo anomalijos atsiranda, kai dėl duomenų dublikatų atnaujinimas pasidaro sudėtingesnis. Kitaip tariant, norint lentelėje pakeisti elementą, reikia jį keisti visuose eilutėse, kuriose jis pasikartoja. Tai reikalauja atidumo, kadangi duomenų gali būti daug ir ji gali būti daugelyje lentelių. Jeigu atnaujinami ne visi laukai, gaunami prieštaringi duomenys;
- įterpimo. Tokios anomalijos atsiranda, kai įrašas negali būti įterptas, kol nėra visos informacijos. Tarkime, kad norima įkelti naują įrašą, bet kol kas jame dar nėra tam tikros informacijos. Tokios situacijos pavyzdys yra lentelė, kurioje saugoma informacija apie fakultetus ir kursus. Esant tokiai lentelės struktūrai, vedant fakultetą būtina suvesti kursą. Jeigu kurso nėra, tuomet atitinkamame lauke įrašoma neapibrėžta reikšmė;
- šalinimo. Tokio tipo anomalijos pasireiškia nenumatytu duomenų praradimu. Tarkime laikinai reikia ištrinti vieną įrašą lentelėje. Tuomet reikia ištrinti visą eilutę kitoje lentelėje, kurioje yra tas įrašas. Tai lemia, kad yra ištrinamas visas įrašas, nors objektas egzistuoja. Šiuo atveju galima būtų pritaikyti pavyzdį, paminėtą prie įterpimo anomalijos – jeigu norima ištrinti kursą, taip pat yra pašalinama ir fakulteto informacija.

Duomenų anomalijų problema gali būti išspręsta naudojant tokias duomenų bazių valdymo technikas, kaip trigeriai, taikymo logika (angl. *application logic*) ar paketo suderinimas (angl. *batch reconciliation*). Trigeriai gali atnaujinti dublikuotus ar išvestus duomenis bet kada, kai tik baziniai duomenys pasikeičia. Jie suteikia geriausią sprendimą iš integralumo pusės, tačiau taip pat gali atimti daug laiko našumo atžvilgiu. Taikymo logika gali būti įtraukta į tranzakcijas kiekvienoje programoje norint atnaujinti denormalizuotus duomenis užtikrinant, jog pokyčiai atominiai. Galiausiai, paketo suderinimo procesas gali būti vykdomas atitinkamais intervalais, bandant užtikrinti denormalizuotų duomenų teisingumą. Toks būdas yra rizikingas, nes ta pati logika turi būti palaikoma ir naudojama visose programose, kurios modifikuoja atitinkamus duomenis [10].

1.1.4. Normalizuotos ir denormalizuotos schemų palyginimas

Normalizuojant lentelės yra išskaidomos į daugiau lentelių. Tai lemia, kad gautos lentelės yra mažesnės, o lentelių kiekis duomenų bazėje yra didesnis. Tokia duomenų išskaidymo struktūra lemia, kad skirtingi faktai yra išsaugomi atskirose lentelėse. Taip yra užtikrinamas duomenų vientisumas. Be to, tokią duomenų bazės schemą lengviau prižiūrėti ir valdyti.

Denormalizuojant mažinamas sujungimų kiekis užklausose. Po denormalizacijos, lentelių kiekis gali išlikti nepakitęs, sumažėti ar padidėti. Be to, denormalizacijos proceso eigoje dažniausiai yra pridedami duomenų dublikatai. Vadinasi, lentelės duomenų bazėje padidėja. Be to, dėl pasikartojančios informacijos, duomenys tampa nebevientisi ir gali atsirasti duomenų anomalijos. Šiuo aspektu denormalizacija prideda sudėtingumo, kadangi sunkiau duomenis prižiūrėti bei suvaldyti. Taigi, tokios lentelės reikalauja daugiau priežiūros.

Normalizuotoje schemoje kiekvienas faktas saugomas atskiroje vietoje, dėl to modifikavimo operacijos (įterpimo, atnaujinimo ir šalinimo) atliekamos greičiau ir paprasčiau. Kitaip tariant – modifikacijos atliekamos tik vienoje gana mažoje lentelėje, o pakeitimai kitoms lentelėms yra perduodami per sąryšius tarp lentelių. Denormalizuotoje schemoje atsiradęs duomenų nevientisumas apsunkina ir sulėtina modifikavimo operacijas. Dėl dublikatų duomenis tenka keisti ne vienoje lentelėje. Todėl esant dažnam modifikavimo operacijoms, palaikomumas yra žemas.

Kita vertus, duomenų išskaidymas po normalizacijos proceso lemia lėtesnį duomenų gavimą. Net ir mažo kiekio įrašų (pavyzdžiui, vieno stulpelio duomenų) gavimui gali tekti skaityti duomenis iš daugelio lentelių. Denormalizacijos procese tai vyksta atvirkščiai – duomenys saugomi tokiu būdu, jog būtų lengviau pasiekiami. Tai lemia, jog skaitymo operacijos yra vykdomos greičiau.

Normalizuotoje schemoje skaitymo užklausoms atlikti gali tekti skaityti duomenis iš daugelio lentelių. Tam turi būti naudojami sujungimai, kurie turi neigiamą įtaką greitaveikai, todėl užklausų vykdymo laikas yra ilgesnis. Tačiau vykdant vienos lentelės skaitymo operaciją, užklausa yra vykdoma greičiau (kadangi lentelės yra mažos). Denormalizuotoje schemoje lentelės dažniausiai būna didesnės, todėl vienos lentelės skenavimas gali užtrukti ilgiau. Tačiau sumažėjęs sujungimų kiekis užklausose lemia greitesnį užklausų vykdymą, todėl bendru atveju skaitymo užklausos yra vykdomos greičiau.

Normalizacija labiau tinka sistemoms, kuriose dažnai atliekami modifikavimo procesai, o ne skaitymo operacijoms. Taip yra dėl to, jos skaitymo užklausos dažniausiai būna sudėtingos ir lėtos (dėl didelio sujungimų kiekio užklausose), o modifikavimas dažniausiai apsiriboja vienos lentelės rėžiuose. Tuo tarpu denormalizuota schema labiau tinka sistemoms, kuriose duomenų gavimas atliekamas dažnai. Taip yra todėl, jog duomenų bazės schema yra perdaroma taip, jog atitinkami duomenys yra lengviau pasiekiami. Bet dėl duomenų dublikatų ir nevientisumo, sulėtėja modifikavimo operacijos.

Dėl to, jog normalizacijoje duomenys išskaidomi į daugiau lentelių, duomenų bazėje padaugėja sąryšių tarp lentelių. Vadinasi, atsiranda daugiau pirminių ir išorinių raktų. Tai gali lemti, jog duomenų bazė užima daugiau vietos. Tačiau normalizacija pašalina dublikatus ir kompaktiškai saugo duomenis. Tai turi didesnę įtaką duomenų bazės dydžiui, todėl galima teigti, jog normalizuota duomenų bazė užima mažiau vietos.

Denormalizuota duomenų bazė dažniausiai užima daugiau vietos. Nors schemoje egzistuoja mažiau lentelių, o tai lemia, kad yra mažiau sąryšių ir raktų, tačiau dažnai denormalizacija atkuria duomenų dublikatus. Dėl šios priežasties duomenų bazėje egzistuoja daugiau įrašų bei elementų, o tuo pačiu ir duomenų bazė yra didesnė.

Normalizuotoje duomenų bazės schemoje, dažniausiai yra pritaikomas trečias ar aukštesnis normalizacijos lygmuo. Denormalizuotoje duomenų bazėje yra mažinamas normalizacijos lygmuo, todėl jis yra žemesnis. Dažniausiai duomenų bazė būna trečiame ar žemesniame normalizacijos lygmenyje.

Normalizacijos ir denormalizacijos palyginimas pateiktas 1.1 lent. Joje apibendrinami anksčiau minėti bruožai – lentelių, užklausų, duomenų schemos charakteristikos. Denormalizacija vertinama bendru atveju, neatsižvelgiant į konkrečius metodus.

1.1 lent. Normalizacijos ir denormalizacijos palyginimas

Parametrai	Normalizacija	Denormalizacija
Lentelių kiekis	Daugiau	Mažiau
Lentelės dydis	Mažesnės	Didesnės
Duomenų valdymas	Lengvesnis duomenų valdymas ir duomenų bazės organizacija	Sudėtingesnis valdymas. Gali būti nevientisi duomenys, atsirasti anomalijos
Labiausiai tinkančios operacijos	Modifikavimo (įterpimo, atnaujinimo, šalinimo)	Skaitymo
Užklausų charakteristikos	• Skaitymo užklausos dažniausiai sudėtingesnės ir lėtesnės • Modifikavimo užklausos paprastesnės ir greitesnės	• Skaitymo užklausos dažniausiai paprastesnės ir greitesnės • Modifikavimo užklausos sudėtingesnės ir lėtesnės
Sujungimų kiekis užklausose	Didesnis	Mažesnis
Skaitymo operacijos vykdymo laikas	• Greitesnis, kai skaitoma iš vienos lentelės • Lėtesnis, jei skaitoma iš daugelio lentelių	Dažniausiai greitesnis
Duomenų bazės dydis	Dažniausiai mažesnis	Dažniausiai didesnis
Sąryšių kiekis tarp lentelių	Didesnis	Mažesnis
Raktų kiekis duomenų bazėje	Didesnis	Mažesnis
Normalizacijos lygmuo	Bent jau 3NF	Mažesnė negu 3NF

1.2. Reliacinės duomenų bazės optimizacijos metodų analizė

Poskyryje pateikiami literatūros šaltiniuose rasti denormalizacijos sprendimai, kurie plačiai pritaikyti reliacinėse duomenų bazėse. Po literatūros apžvalgos atrasta, kad galimi denormalizacijos būdai yra:

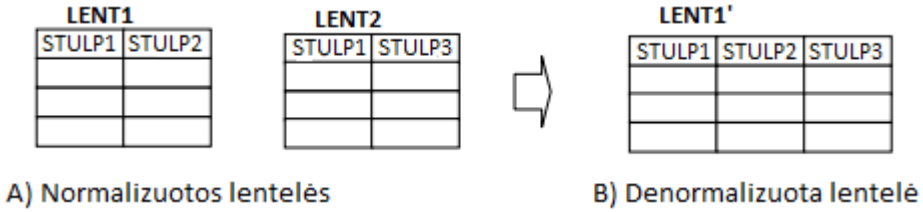
- lentelių sujungimas. Keletas lentelių sujungiama į vieną, norint išvengti sujungimo operacijų užklausose;
- lentelių skaidymas. Iš vienos lentelės padaromos kelios. Tokiu būdu išvengiama paieškos per duomenis ar stulpelius, kurie retai naudojami. Tai sumažina fizinę vietą, kurioje ieškoma, bei padeda išvengti priėjimo prie retai naudojamos informacijos. Gali būti du skaidymo būdai:
  - vertikaliai (atskiriant pagal stulpelius),
  - horizontaliai (atskiriant pagal duomenis),
- pasikartojančių stulpelių (angl. *repeating groups*) pridėjimas. Lentelėje saugomi pasikartojantys vienos rūšies stulpeliai. Tokiu būdu yra sumažinamas lentelių kiekis bei išvengiama lentelių sujungimo užklausose;
- pasikartojančios informacijos pridėjimas. Duomenys kopijuojami į kitas lenteles ir naudojami kaip nuorodos. Tai daroma tam, kad būtų išvengta daugelio kiekio lentelių sujungimų;
- skaičiuojamųjų laukų saugojimas. Išsaugant skaičiuojamus stulpelius lentelėse, išvengiama grupuojančių sujungimų (angl. *grouping joins*).

1.2.1. Lentelių sujungimas

Atskiroms lentelėms atliekamas sujungimas užklausoje reiškia, kad kiekvienam įrašui vienoje lentelėje atliekamas ciklas kitoje lentelėje tam, kad būtų rastas atitinkamas įrašas. Aptikti įrašą yra daug kainuojanti operacija, kuri gali užtrukti kelis šimtus kartų ilgiau negu paprasto įrašo skenavimas. Duomenų perkėlimas į vieną lentelę gali padėti išvengti sujungimo operacijos. Tačiau tokiu atveju lentelė užima daugiau vietos, todėl ir lentelės skenavimas užtrunka ilgiau. Bendru atveju, sujungti galima lenteles, turinčias tiek vienas su vienu, tiek daug su daug, tiek vienas su daug ryšį [6, 22, 23].

Viena iš dažniausių ir saugiausių denormalizacijos technikų yra lentelių, turinčių 1:1 ryšį, sujungimas. Lentelės su tokiu sąryšiu atsiranda, kai kiekvienai A lentelės eilutei yra tik viena (ar nei viena) susijusi eilutė lentelėje B. Nors šių lentelių raktiniai atributai gali skirtis, tačiau jų ekvivalentus dalyvavimas ryšyje nurodo, kad į jas galima žiūrėti kaip į vienetą. Pavyzdžiui, jeigu vartotojui reikia

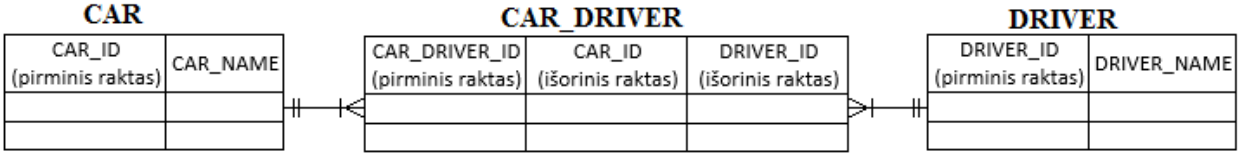
dažnai gauti *STULP1*, *STULP2* ir *STULP3* vienu metu ir duomenys iš abiejų lentelių yra 1:1 ryšyje, tuomet sprendimas yra sujungti šias dvi lenteles į vieną (1.5 pav.). Priešingu atveju, *SQL* užklausa, kuria siekiama ištraukti informaciją iš šių dviejų lentelių, turėtų naudoti dviejų lentelių sujungimą vien tam, kad būtų gauta reikiama informacija. Be to, gali būti norima gauti ir kitą informaciją (iš kitų lentelių), ko pasekoje gali gautis labai dideli sujungimai. Taigi, tokia lentelių struktūra gali vesti prie prastos greitaveikos [7, 9, 10, 24].



1.5 pav. Lentelių sujungimo schema [10]

Yra keletas tokio sprendimo privalumų: sumažėjęs išorinių raktų kiekis lentelėse, sumažėjęs indeksų kiekis (kadangi dauguma indeksų yra sukurti remiantis pirminiais ir išoriniais raktais), sumažėjęs duomenų bazės dydis ir sutrumpėjęs duomenų modifikavimo laikas. Be to, stulpelių sujungimas sumažina duomenų gavimo laiką, kadangi yra mažiau fizinių objektų ir yra sumažėjusi lentelės antraštė (angl. *overhead*). Lyginant su kitais optimizacijos metodais, šis būdas turi mažiausiai trūkumų. Tačiau sprendimas pasiteisina tik tuomet, jei originalios lentelės yra dažnai naudojamos kartu. Kitu atveju, toks sujungimas padaro lentelę didesnę, todėl užklausos turi ieškoti informacijos didesnėje fizinėje vietoje. Dėl to užklausos vykdymas gali pailgėti [9, 10, 24].

Lentelės, turinčios daug su daug ryšį, taip pat yra kandidatės lentelių sujungimui. Tipinės lentelės, turinčios N:M ryšį, fizinėje duomenų bazės struktūroje yra pateikiamos trimis lentelėmis: po vieną lentelę kiekvienam iš dviejų pagrindinių objektų, o kita lentelė yra skirta jų kryžminėms nuorodoms (angl. *cross-reference*) [10, 24]. 1.6 pav. pagrindinių objektų lentelės atvaizduotos *CAR* ir *DRIVE*, o kryžminių nuorodų lentelę atvaizduoja *CAR_DRIVER*.

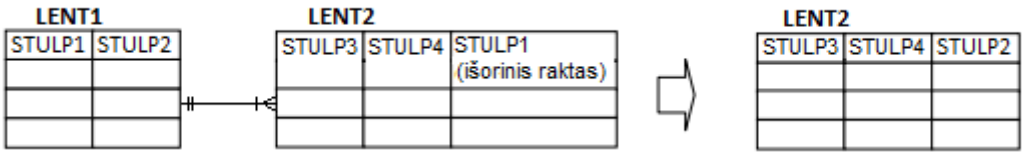


1.6 pav. Lentelės, turinčios N:M ryšį

Tokio tipo lentelės gali būti sujungtos, jeigu viena iš jų turi nedaug duomenų (išskyrus pirminį raktą) – nėra daug priklausomybių su pirminiu raktu. Tokia lentelė gali būti sujungta į kryžminės nuorodos lentelę, dubliuojant atributų duomenis. Žinoma, šis sprendimas turi trūkumų. Kadangi duomenų dubliavimas gali trukdyti duomenų atnaujinimui, gali atsirasti atnaujinimo anomalijos, kai susijungusi lentelė turi egzempliorius, kurie neturi jokių įrašų kryžminėje nuorodų lentelėje [10].

Tokio tipo sujungimas eliminuoja sujungimus užklausoje, tačiau patiriamas didelis nuostolis abstrakčiu lygmeniu, kadangi nebėra konceptualaus duomenų atskyrimo. Bendru atveju, lentelių, turinčių daug su daug ryšį, sujungimas sukuria daug daugiau problemų negu kiti būdai [10].

Panašiai, kaip ir kiti ryšiai, gali būti sujungiamos ir lentelės su 1:N ryšiu. Visos šios lentelės, nepriklausomai nuo ryšio tipo, dažniausiai turi vieną bruožą – tokios lentelės atsiranda, kai persistengiama su 4 normaline forma norint pašalinti tuščias (angl. *null*) reikšmes. Taigi, tokių lentelių sujungimas yra normalinių formų anuliavimas. Todėl sujungus lenteles galima pagerinti greitaveiką [7, 9, 24]. Lentelių, esančių 1:N ryšyje, sujungimas pateiktas 1.7 pav.



1.7 pav. Lentelių sujungimo schema [10]

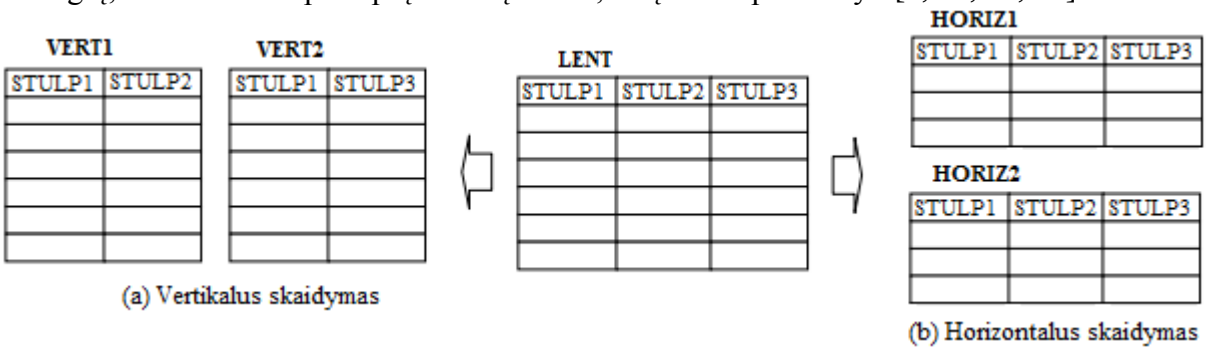
Bendru atveju, lentelių sujungimas panaikina sąryšį tarp lentelių, todėl sumažėja ir raktų kiekis. Taip pat sumažėja ir indeksų kiekis, kadangi dažniausiai indeksai yra kuriami remiantis pirminiais ir išoriniais raktais. Dėl to sumažėja sujungimo operacijų kiekis užklausose ir duomenys gaunami greičiau. Tačiau toks metodas gali pridėti sudėtingumo, kadangi atsiranda duomenų dublikatų.

1.2.2. Lentelių skaidymas

Gali būti ir tokie scenarijai, kai lentelėje yra dideli ir retai naudojami duomenys. Tuomet greitaveika gali būti pagerinta perkėlus šią informaciją į atskirą lentelę. Taip yra todėl, kad tuomet dažniau naudojami duomenys yra saugomi mažesnėje lentelėje, o rečiau pasiekiama informacija yra peržiūrima tik tuomet, kai reikia. Kitaip tariant, yra atskiriama dažniau naudojama informacija nuo rečiau pasiekiamos. Įtaka greitaveikai, atsiradusi dėl dažnų sujungimų, yra kompensuojama ieškant aktualesnės informacijos mažesnėje lentelėje. Nors sujungimai naudoja CPU laiką, tačiau didelės lentelės užima disko vietą ir disko pasiekiamumo laiką [7, 10, 12, 22, 24].

Galimi du skaidymo būdai: vertikalus ir horizontalus. Vertikalus skaidymas apima lentelių skaidymą pagal stulpelius tokiu būdu, kad stulpelių grupė yra perkeliami į vieną naują lentelę, o likę stulpeliai yra perkeliami į kitą (žr. 1.8 pav. (a)). Toks skaidymas gali būti naudojamas tada, kai tam tikri stulpeliai yra rečiau naudojami negu kiti stulpeliai arba kai lentelė turi plačias eilutes (daug stulpelių). Pavyzdžiui, 90% užklausų pasiekia tuos pačius 5 iš galimų 65 lentelės stulpelių. Duomenys būtų gaunami greičiau, jeigu būtų sukurta lentelė, kurioje saugomi tik tie 5 stulpeliai [6, 7, 9, 10, 12].

Bendras eilučių kiekis puslapyje priklauso nuo lentelės pločio. Pavyzdžiui, *SQL Server* duomenų bazę sudaro 8 KB puslapiai, o eilutės negali aprėpti daugiau nei vieną puslapį. Tai lemia, kad kuo platesnė lentelė, tuo mažesnis eilučių kiekis puslapyje. Vertikalaus dalinimo technika sumažina lentelės plotį, kadangi vienos lentelės stulpeliai yra padalindami per kelias lenteles atsižvelgiant į duomenų naudojimo dažnumą. Pagrindinis rezultatas yra tai, kad dėl sutrumpėjusių eilutės ilgių, sumažinamas puslapių / blokų kiekis, kurį reikia perskaityti [7, 10, 12, 25].



1.8 pav. Lentelių skaidymo schemos [10]

Iš dalies šis būdas primena normalizaciją, nes lentelės yra skaidomos. Tačiau normalizacijoje yra skaidomi skirtingo tipo duomenys (pavyzdžiui, užsakymai ir pirkėjai), o šiuo atveju – to pačio tipo duomenys. Kiekviena vertikaliai išskaidyta lentelė turi turėti po pirminį raktą, nes tai palengvina duomenų gavimą per lenteles. Todėl, bendru atveju, duomenų bazė padidėja. Ši technika yra efektyvi, kai ilgose eilutėse yra ilgi tekstiniai laukai. Jei tekstiniai laukai yra retai pasiekiami, jie gali būti perkelti į atskirą lentelę [7, 10, 12, 23].

Horizontalus skaidymas apima skaidymą pagal eilutes. Lentelė yra išskaidoma į kelias tokiu būdu, kad kiekvienoje naujoje lentelėje struktūra yra identiška originaliai, o tarp lentelių skiriasi tik duomenų rinkiniai (žr. 1.8 pav. (b)). Rezultate eilutės yra klasifikuojamos į grupes pagal tam tikrus rėžius. Dažniausiai ši technika yra pritaikoma, kai lentelės skaidymas atitinka natūralų eilučių atskyrimą (pavyzdžiui, skirtingos geografinės sritys ar istorinė ir dabartinė data) [6, 7, 10, 12, 25].

Kai lentelės auga, duomenų gavimo laikas auga taip pat. Užklausoms, kurios turi įvykdyti lentelės skenavimą, vykdymo laikas yra tiesiogiai proporcingas lentelės eilučių kiekiui. Pritaikius horizontalų skaidymą, lentelės dydis yra mažinamas taip mažinant indeksuotų puslapių skaitymo užklausoje kiekį. Technika gali būti naudojama tada, kai lentelė saugo didelį kiekį retai naudojamų

istorinių duomenų ir tuo pačiu metu būna atvejų, kai reikalingas greitas rezultato atvaizdavimas iš tokios lentelės [7, 10, 12, 23].

Horizontalus skaidymas gali pagerinti greitaveiką, ypač kai istoriniai ar archyviniai duomenys nusveria esamus. Tai gali būti įvardinta primityviu dalinimu (angl. *partitioning*). Tikslinga duomenis skaidyti pagal periodus, pavyzdžiui metus, ketvirčius ar mėnesius. Tačiau toks būdas pasiteisina tik tuomet, kai duomenis reikia imti iš vieno rinkinio (pavyzdžiui, vieno mėnesio). Jei, pavyzdžiui, lentelės padalintos pagal mėnesius, o reikia gauti metinius duomenis, tuomet tenka naudoti sujungimus ir užklausos kaip tik tampa sudėtingesnės ir vykdomos ilgiau [7, 10, 12].

Bendru atveju, horizontalus skaidymas prideda sudėtingumo. Jeigu išskaidytose lentelėse yra pasikartojančių eilučių, jungiant lenteles atsiranda duomenų dublikatų ir netikslumų. Po skaidymo naujos lentelės turi naujus pavadinimus, todėl užklausose turi būti naudojami lentelių vardai atsižvelgiant į reikšmes lentelėse. Tai lemia, kad užklausoje lenteles tenka rinkti pagal duomenų reikšmių kontekstą. Atsiradęs sudėtingumas dažniausiai nusveria skaidymo privalumus [7, 10, 12].

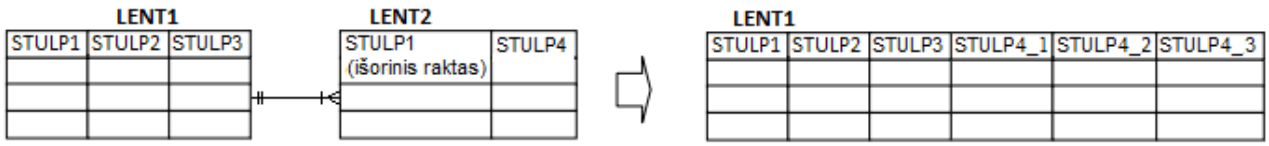
Abiem atvejais duomenų atskyrimas užtikrinama, kad retai naudojami duomenys neįsiterptų į dažnai pasiekiamus duomenis. Taip viena lentelė išskaidoma į daugiau mažesnių. Taigi, abu lentelių skaidymo būdai sukuria daugiau lentelių ir papildomų ryšių bei galimai sukurtos lentelės užima daugiau vietos. Tačiau, lentelės savaime yra mažesnės – o turint mažesnes lenteles, užklausos vykdomos trumpiau. Be to, šie sprendimai netinka, jeigu dažnai reikia naudoti duomenis iš išskaidytų lentelių kartu.

1.2.3. Pasikartojančių stulpelių pridėjimas

Pasikartojanti grupė (angl. *repeating group*) yra susijusių reikšmių kolekcija. Atsižvelgiant į normalizacijos taisykles, pasikartojanti grupė yra saugoma atskiroje lentelėje. Tokie atributai yra traktuojami kaip subjektai, todėl saugomi kaip atskiri įrašai. Tokiu būdu egzistuoja dvi 1:N sąryšį turinčios susijusios lentelės – pagrindinė ir detalių, kurioje saugoma pasikartojanti grupė. Pasitaiko situacijų, kai detalių lentelėje pasikartojanti grupė yra dažnai pasiekiama kartu su pagrindinės lentelės duomenimis ir tarp šių lentelių yra fiksuotas įrašų kiekis (arba sąryšis turi fiksuotą įrašų maksimumą). Tokiu atveju pagrįstas sprendimas yra denormalizuoti šias lenteles. Šiuo atveju pasikartojanti grupė yra realizuojama kaip atskiri įvardinti stulpeliai pagrindinėje lentelėje. Tokia denormalizacija veikia geriausiai, kai įrašų kiekis detalių lentelėje yra nedidelis [22, 23, 25-27].

Tarkime yra situacija, kai vartotojas gali turėti keletą telefono numerių. Tokiu atveju normalizuotas sprendimas yra turėti dvi lenteles – vieną vartotojo (pagrindinė lentelė) informacijai saugoti, o kitą – telefono numeriams (detalių lentelė). Kitas sprendimas yra denormalizuotas – galima turėti vieną vartotojo lentelę, kurioje papildomai saugomi ir telefono numeriai (telefonas1, telefonas2, telefonas3, ...) kaip pasikartojantys stulpeliai [26, 27].

Šio metodo pavyzdinė schema pateikta 1.9 pav. Šiuo atveju pagrindinė (remiantis anksčiau pateiktu pavyzdžiu – vartotojo) lentelė yra *LENT1*, o detalių (arba telefono numerių) lentelė yra *LENT2*. Kairėje pusėje pateikta normalizuota schema, o dešinėje – denormalizuota. Denormalizuotos schemos atveju sprendimas pasiteisina, nesvarbu pagal kurį vartotojo stulpelį ieškomi detalių stulpeliai (telefono numeriai). Tačiau normalizuotoje schemoje yra kitaip. Jeigu telefono numeriai ieškomi pagal vartotojo identifikatorių (*STULP1*), tuomet sprendimas pasiteisina ir užklausoje skenuojama viena lentelė. Jeigu užklausa remiasi kitu stulpeliu – pavyzdžiui, vartotojo vardu (*STULP2*) – tuomet užklausoje atsiranda sujungimas su vartotojų lentele, nes numerių lentelėje saugomas tik vartotojo identifikatorius [26, 27].



1.9 pav. Pasikartojančių stulpelių pridėjimo schema

Normalizuotos schemos atveju duomenų bazėje galima saugoti begalinį detalių įrašų kiekį. Šiuo atveju vienintelis apribojimas yra leidžiama disko vieta. Jeigu nusprendžiama detalių įrašus įrašyti į

stulpelius (denormalizuota schema), tuomet egzistuoja ribotas kiekvienoje eilutėje esančių stulpelių skaičius. Šiame pavyzdyje apribojimas yra trys telefono numeriai vienam vartotojui (*STULP4_1, STULP4_2* ir *STULP4*_3). Toks ribotas detalių įrašų kiekis mažina sistemos lankstumą, kadangi gali prireikti daugiau reikšmių [26, 27].

Denormalizuotu atveju visi detalių įrašai gaunami viename pagrindinės lentelės įraše. Normalizuotoje schemoje turėtų būti grąžinama N įrašų, norint gauti tokią pačią informaciją. Šiuo atveju N yra skaičius, nurodantis, kiek pagrindinės lentelės eilutė turi detalių įrašų (pavyzdinėje schemoje N = 3). Tokiu būdu denormalizuotoje schemoje yra mažinamas sujungimų kiekis užklausose, nes visa informacija yra saugoma vienoje lentelėje. Toks metodas labiausiai tinka tuomet, kai visiems pagrindinės lentelės įrašams egzistuoja fiksuotas ir statiškas detalių įrašų skaičius. Toks įrašų saugojimas ne tik sumažina sujungimų kiekį užklausose, bet padeda išsaugoti vietą, kadangi egzistuoja mažiau pirminių ir išorinių raktų [25-28].

Denormalizuojant duomenis ir saugant juos skirtinguose stulpeliuose galima pasiekti geresnę greitaveiką, tačiau tai pasiekiama lankstumo kaina. Vienas iš šio metodo trūkumų yra toks, kad tokius duomenis sunkiau valdyti bei vykdyti *SELECT* ar kitas (pavyzdžiui, rūšiavimo) užklausas detalių reikšmėms. Be to, atsiranda nepatogumų, kai užklausoje naudojamas antrinis prieigos kelias. Detalių stulpelių reikšmių patikrinimai turi būti atlikti per visus pasikartojančius stulpelius (pateiktame pavyzdyje – per *STULP4_1, STULP4_2* ir *STULP4*_3). Tokios užklausos pavyzdys būtų vartotojo, kuris turi konkretų telefono numerį X, suradimas. Normalizuotu atveju, norint gauti rezultatą, pakanka skenuoti telefono numerių lentelę (žr. užklausa (1)). Denormalizuotu atveju sistemos atsako laikas yra blogesnis, kadangi lentelėje vykdomas pilnas vartotojų lentelės skenavimas ir tikrinamos visų detalių stulpelių reikšmės (žr. užklausa (2)). Tačiau tokios užklausos nėra dažnos, todėl problema nėra didelė [25-27].

```
select * from LENT2 where STULP4 = 'X'                    (1)
```

```
select * from LENT1
where STULP4_1 = 'X' or STULP4_2 = 'X' or STULP4_3 = 'X'  (2)
```

Kitas metodo trūkumas yra toks, kad galimas tik baigtinis detalių stulpelių kiekis. Todėl sprendimas pasiteisina tik tada, kai žinoma, kad sąrašas yra ribotas, o ne begalinis. Be to, pasikartojantys stulpeliai turi būti įvardinami (pavyzdžiui, *STULP4_1, STULP4-1)*. Įvardinimas turi būti prasmingas, kad vėliau nesukeltų nepatogumų. Taip pat pasikartojantys stulpeliai negali būti indeksai [26-28].

Prieš saugant pasikartojančius sąrašus kaip stulpelius (vietoj įrašų), turi būti atsižvelgta į sekančius kriterijus [27]:

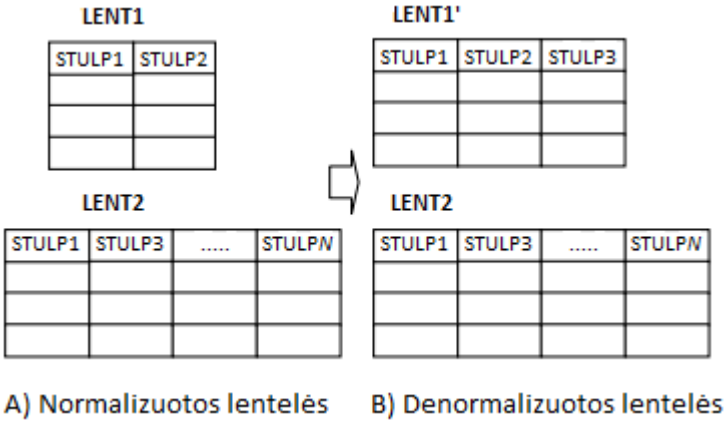
- su duomenimis eilutėje niekada arba retai yra atliekami tokie veiksmai kaip apibendrinimas, vidurkio skaičiavimas ar palyginamas;
- duomenų atsiradimas gali būti aprašytas statišku ir nuspėjamu modeliu;
- duomenys turi stabilų paplitimo (angl. *occurrence*) skaičių;
- duomenys dažniausiai yra gaunami kartu;
- duomenys turi nuspėjamą įterpimo bei šalinimo modelį.

Jeigu kuris nors iš anksčiau įvardintų kriterijų neatitinka, tuomet *SELECT* užklausos gali pasidaryti sudėtingesnės. Taip yra todėl, jog duomenys yra sunkiau prieinami dėl prasto duomenų modeliavimo. To reikėtų vengti, nes schema denormalizuojama tik tam, kad duomenys būtų lengviau ir greičiau prieinami. Bendrai, metodas tinka, kai žinomas tikslus elementų pasikartojančioje grupėje skaičius ir šis kiekis yra statiškas ir nesikeičiantis [27].

1.2.4. Pasikartojančios informacijos pridėjimas

Pasikartojančių stulpelių pridėjimas naudojamas tada, kai stulpelis iš vienos lentelės yra pasiekiamas kartu su stulpeliu iš kitos lentelės. Kitaip tariant – vienos lentelės stulpeliai yra pasiekiami kartu su kitos lentelės duomenimis. Jeigu tai pasikartoja dažnai, galima sujungti duomenis

ir turėti juos kaip dublikatus. Pavyzdžiui, jeigu dažnai vykdomas sujungimas tarp *LENT1* ir *LENT2* tam, kad būtų pasiekti *STULP1, STULP2* ir *STULP3*, tinkama strategija yra pridėti *STULP3* į *LENT1* lentelę (žr. 1.10 pav.). Tokiu atveju dažniausiai pasiekiami stulpeliai yra saugomi vienoje vietoje ir užklausose nebereikia naudoti sujungimų [6, 7, 9, 10, 20, 22].



1.10 pav. Pasikartojančios informacijos pridėjimo schema [10]

Užklausa (3) iliustruoja atvejį, kai norint gauti knygos pavadinimą ir autoriaus vardą, tenka naudoti trijų lentelių (*authors, titleauthor, title*) sujungimą. Tokia užklausa yra neefektyvi, nes kelių reikšmių gavimui, informacijos tenka ieškoti daugelyje lentelių. Norint išvengti tokios situacijos, galima pridėti atitinkamus atributus (kurie bus kopijos arba dublikatai) į lenteles – pridėti autoriaus vardo ir pavardės stulpelius į *titles* lentelę ir saugoti informaciją joje. Naujosios reikšmės nėra funkcionaliai priklausomos nuo pirminio rakto, tačiau toks sprendimas pašalina sujungimus, o ta pati reikšmė yra saugoma tiek *titles,* tiek *authors* lentelėse. Užklausa, kuri grąžina tokią pat informaciją kaip ir (3) po dublikatų įterpimo pateikta (4) [12].

```
select c.title, a.au_lname, a.au_fname from authors a join
titleauthor b on a.au_id = b.au_id join title c on              (3)
b.title_id = c.title_id where b.au_ord = 1 order by c.title
```

```
select title, au_lname, au_fname from titles order by title     (4)
```

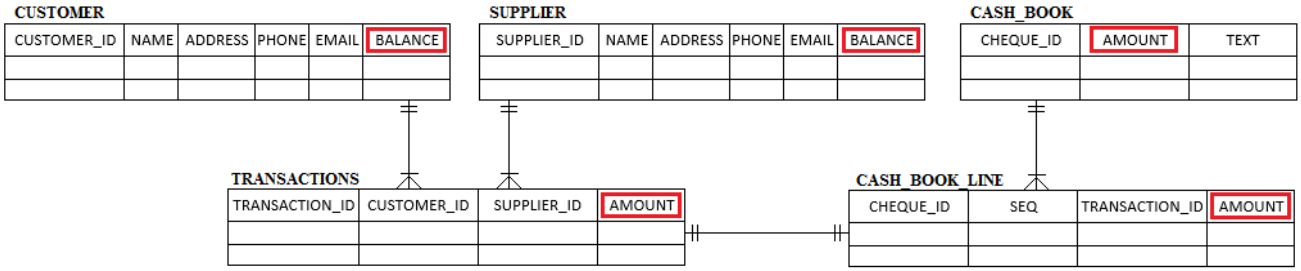
Geriausia dubliuoti nedidelį stulpelių kiekį. Renkantis stulpelius dubliavimui, geriausia imti tokius, kurie yra reliatyviai statiški ir keičiasi retai. Taip yra todėl, kad pasikeitus duomenims vienoje lentelėje, jie turi pasikeisti ir kitoje. Tai lemia, kad atnaujinant duomenis, atnaujinimo operacijos greitis gali nukentėti dramatiškai. Be to, didėja tikimybė turėti nevientisus duomenis [12].

Toks sprendimas nebūtinai sumažina lentelių kiekį. Tačiau, tai pašalina poreikį sujungti lenteles, kadangi tam tikras stulpelis yra dubliuojamas kitoje lentelėje ir informacija gali būti pasiekiama iš mažesnio lentelių kiekio [23]. Dublikato saugojimo problema yra ta, jog lentelė tampa platesnė, dėl to ji užima daugiau vietos duomenų bazėje. Toks metodas lemia, jog turi būti valdomi tiek duomenų originalai, tiek jų kopijos. Tai apsunkina duomenų modifikavimą. Be to, duomenų dublikatai padidina riziką susidurti su duomenų anomalijomis. Tačiau, kaip jau buvo minėta anksčiau, duomenų anomalijų problema gali būti išspręsta naudojant tokias duomenų bazių valdymo technikas, kaip trigeriai, taikymo logikos arba paketo suderinimai [10, 24].

1.2.5. Skaičiuojamųjų laukų saugojimas

Pasitaiko situacijų, kai suvestiniai (angl. *summary*) ir išvestiniai (angl. *derived*) duomenys yra labai svarbūs. Daugelio programų pobūdis diktuoja dažną duomenų, apskaičiuojamų iš atitinkamos informacijos, naudojimą. Dėl didelio duomenų kiekio, net paprasti skaičiavimai gali prailginti apdorojimo laiką. Be to, jei tokie skaičiavimai atliekami realiu laiku, tuomet atsako (angl. *response*) laikas yra paveikiamas neigiamai. Galiausiai, tokie skaičiavimai gali apkrauti CPU ir gali turėti didelę įtaką greitaveikai [10, 12, 22, 24].

Išeitis yra apskaičiuoti tam tikras sumas ir išsaugoti rezultatą atskiroje lentelėje (arba lentelės stulpelyje). Skaičiuojami laukai gali būti saugomi tėvinėje lentelė tam, kad būtų apibendrinta informacija, esanti vaikinėse lentelėse. Kad toks sprendimas būtų efektyvus, išvedami duomenys turi būti dažnai pasiekiami skaičiuojami kiekiai. Toks saugojimas reiškia, kad nebereikia jungti lentelių ir iš naujo vykdyti grupavimo užklausas vien tam, kad rasti skaičiuojamą lauką. 1.11 pav. *balance* stulpelis *Customer* ir *Supplier* lentelėse sumuoja atitinkamas vaikinių lentelių sumas. Taip išvengiama duomenų skaičiavimo iš trijų lentelių [7, 9, 12, 20, 23].



1.11 pav. Skaičiuojamųjų laukų saugojimas [20]

Turint išvedamus duomenis, labai svarbu sinchronizuoti reikšmes. Skaičiavimai gali būti atliekami trimis būdais [12]:

- perskaičiuojama realiu laiku. Kiekvieną kartą, kai pagrindiniai (iš kurių yra išvedami skaičiavimai) duomenys keičiasi, suminiai duomenys yra perskaičiuojami naudojant pagrindinius duomenis kaip šaltinį. Tam dažniausiai naudojamos išsaugotos (angl. *stored*) procedūros ar trigeriai;
- pridedama realiu laiku. Kiekvieną kartą, kai pagrindiniai duomenys pasikeičia, suminiai duomenys perskaičiuojami naudojant seną jų reikšmę ir naujus duomenis. Toks būdas yra sudėtingesnis, bet gali sutaupyti laiko, ypač jei nauji duomenys yra reliatyviai maži lyginant su visu duomenų rinkiniu. Tokie skaičiavimai taip pat atliekami naudojantis išsaugotas procedūras ir trigerius;
- skaičiuojama su vėlinimu. Tokiu atveju naudojamos suplanuotos užduotys, kurios perskaičiuoja suminius duomenis.

Saugant tokią išvedamą informaciją duomenų bazėje gali iš esmės pagerinti greitaveiką trumpinant tiek CPU ciklus, tiek duomenų skaitymo laiką. Be to, išvestų duomenų saugojimas gali padėti pašalinti sujungimus užklausose ir sumažinti daug laiko trunkančius skaičiavimus realiu laiku. Ši technika gali būti efektyviai derinama ir su kitomis denormalizacijos technikomis. Tačiau toks būdas pažeidžia normalizacijos principus. Taip pat, gali būti sudėtinga išlaikyti duomenų integralumą, jeigu duomenys, kurie naudojami išvedamų duomenų skaičiavimui, keičiasi nenuspėjamai ar dažnai. Taip yra todėl, kad nauja išvedama reikšmė turi būti perskaičiuota kiekvieną kartą, kai komponentiniai duomenys pasikeičia. Todėl, pasiekiamumo dažnumas, norimas atsako laikas ir padidėję palaikymo kaštai turi būti įvertinti prieš taikant šį sprendimą [10, 12].

1.2.6. Optimizacijos metodų palyginimas

Šiame poskyryje palyginami anksčiau išanalizuoti metodai. Kadangi technikos yra skirtingos ir taikomos skirtingose situacijose, todėl negalima jų vienareikšmiškai vertinti tarpusavyje. Dėl šios priežasties metodai lyginami pagal parametrus – kiekvienam iš jų įvertinamas paveiktų lentelių dydis ir kiekis, elementų kiekis, kada galima naudoti metodą, koks gaunamas rezultatas, kaip pasiekiama geresnė greitaveika, kaip pasikeičia *SQL* užklausa, kada būdas tinka ar netinka bei kokie parametrai greitėja. Šios metrikos nustatomos teoriniu aspektu, lyginant duomenų bazę prieš taikant metodą ir po to. Palyginimas pateiktas 1.2 lent.

1.2 lent. Denormalizacijos metodų palyginimas

Metodai / Parametrai	Lentelių sujungimas	Lentelių skaidymas	Pasikartojančių stulpelių pridėjimas	Pasikartojančios informacijos pridėjimas	Skaičiuojamųjų laukų saugojimas
Kada galima naudoti	Norint pagreitinti dažnai kartu naudojamų duomenų gavimą	Lentelėje egzistuoja dažnai naudojama informacija	Norint sumažinti kartu naudojamų lentelių kiekį	Norint sumažinti kartu naudojamų lentelių kiekį	Norint pašalinti skaičiavimus ir algoritmus užklausose
Paveiktų lentelių kiekis, dydis	Mažiau, bet jos didesnės	Daugiau, bet jos mažesnės	Mažiau, bet jos didesnės	Nepakinta, bet jos didesnės	Priklauso nuo situacijos (daugiau arba didesnės)
Elementų kiekis	Priklauso nuo situacijos (daugiau arba mažiau)	Daugiau, nes prisideda identifikatorius (Vertikalus skaidymas) Nepakinta (Horizontalus skaidymas)	Didesnis	Didesnis	Didesnis
Gautas rezultatas	Duomenys saugomi vienoje lentelėje	Atskirai saugomi aktualūs / neaktualūs ir aktyvūs / pasyvūs duomenys	Pagrindiniai ir detalių duomenys saugomi vienoje lentelėje	Lentelėje saugomos kitos lentelės duomenų kopijos	Skaičiuojami laukai išsaugoti duomenų bazėje
Kokiu būdu pasiekiama geresnė greitaveika	Nereikia naudoti sujungimų užklausose	Duomenys gaunami iš mažesnės lentelės	Nereikia naudoti sujungimų užklausose	Sudaromas trumpesnis duomenų pasiekimo kelias	Gaunant duomenis nereikia skaičiuoti rezultatų iš galimai kelių lentelių
SQL užklausų pokyčiai	Nėra sujungimų	Duomenys renkami pagal kontekstą	Nėra sujungimų	Nėra sujungimų	Nėra agregatų ir sujungimų
Kada būdas tinka	Kai duomenys iš skirtingų lentelių dažnai naudojami kartu	Dažnai reikalinga tik dalis lentelės duomenų	Kai yra fiksuotas susietų įrašų kiekis tarp lentelių ir duomenys naudojami kartu	Kai duomenys naudojami sąryšiui nustatyti, o ne vaizdavimui	Kai skaičiavimai atliekami dažnai, realiu laiku ir iš kelių lentelių
Kada būdas nepasiteisina	Kai reikia dažnai modifikuoti duomenis	Kai dažnai reikia gauti duomenis iš abiejų lentelių	Kai duomenys nėra statiški; nežinomas tikslus detalių įrašų kiekis	Kai duomenys dažnai keičiasi	Kai netinka atsiradę palaikymo kaštai; kai nėra daug skaičiuojamų laukų
Kas greitėja	Sutrumpėja užklausos vykdymo laikas	Sutrumpėja lentelės skaitymo laikas	Sutrumpėja užklausos vykdymo laikas	Sutrumpėja užklausos vykdymo laikas	Sutrumpėja užklausos vykdymo laikas

Sujungus lenteles, sumažėja bendras lentelių kiekis duomenų bazėje, o naujoji lentelė yra didesnė. Bendras elementų kiekis priklauso nuo situacijos. Jeigu dvi lentelės prieš sujungimą turėjo griežtą vienas su vienu ryšį (nebuvo įrašų, kurie egzistavo tik vienoje lentelėje), tuomet tikėtina, kad elementų kiekis sumažės. Taip yra todėl, kad vienos lentelės identifikatorius tampa nebereikalingas ir todėl jis gali būti pašalinamas (nebenaudojamas vienos lentelės išorinis raktas). Jeigu dvi lentelės prieš sujungimą turėjo tokių įrašų, kurie yra vienoje lentelėje, bet kitoje jų nėra, tuomet po sujungimo gali atsirasti tuščių reikšmių. Tai lemia, kad elementų kiekis gali padidėti. Taigi, sujungus lenteles, elementų kiekis gali tiek sumažėti, tiek padidėti. Tačiau duomenys po sujungimo yra saugomi vienoje vietoje, todėl užklausose nereikia naudoti sujungimų tarp lentelių. Kadangi sujungimai yra lėta operacija, jų mažinimas (ar eliminavimas) pagreitina užklausos vykdymo laiką. Žinoma, didesnė lentelė lemia ilgesnį lentelės skenavimo laiką, tačiau sujungimų mažinimas yra svarbesnis aspektas, kai analizuojama greitaveika. Taigi, toks būdas pasiteisina tik tuomet, kai duomenys dažnai naudojami kartu. Kitu atveju sujungimas sukuria didesnę lentelę ir didesnę tikimybę turėti nevientisus duomenis (todėl yra sunkiau modifikuoti duomenis).

Kitokie rezultatai gaunami naudojant lentelių skaidymo metodą. Po vertikalus ir horizontalus skaidymo duomenų bazėje atsiranda daugiau, bet mažesnių lentelių. Šie metodai tinka, kai dažnai yra

naudojami atitinkami lentelės stulpeliai ar įrašai. Kitaip tariant – kai dažnai skenuojama ne visa lentelė, o jos dalis. Tokiu atveju, skaidymas išskiria duomenis – vertikaliu atveju vieni stulpeliai yra saugomi vienoje lentelėje, o likę – kitoje, o horizontaliu atveju vieni įrašai saugomi vienoje lentelėje, o likę – kitoje. Kadangi vertikalaus skaidymo metu išskiriami pradinės lentelės stulpeliai, kiekvienoje naujoje lentelėje atsiranda po identifikatorių. Vadinasi, stulpelių ir elementų kiekis padidėja. Horizontalaus skaidymo atveju įrašų kiekis nepakinta, nes lentelės struktūra nepakinta.

Po skaidymo naujos lentelės būna mažesnės (lyginant su originaliomis). Tai lemia, kad lentelės skenuojamos greičiau, o dažnai naudojamų duomenų gavimo laikas yra trumpesnis. Tačiau, išskaidžius duomenis į atskiras lenteles, užklausose tenka atsižvelgti į duomenų kontekstą – rinkti duomenis tarp vienos iš naujų lentelių. Jeigu išskaidoma taip, jog norimi duomenys yra saugomi skirtingose lentelėse, užklausose reikia naudoti sujungimus ir metodas nepasiteisina. Tačiau metodas tinka, jeigu dažnai gaunami būtent tie duomenys, kurie patalpinti vienoje išskaidytoje lentelėje.

Turint pasikartojančių stulpelių pridėjimo metodą, pagrindinė ir detalių lentelės sujungiamos ir informacija saugoma pagrindinėje lentelėje. Tai lemia, kad duomenų bazėje sumažėja lentelių kiekis, o paveikta lentelė yra platesnė (turi daugiau elementų), kadangi lentelėje detalių informacija yra saugoma kaip atskiri stulpeliai. Duomenų saugojimas vienoje, o ne keliose lentelėse sumažina ne tik skenuojamų lentelių kiekį užklausoje, bet ir sujungimų kiekį. Kadangi užklausa supraprastėja, duomenų gavimo užklausos vykdomos greičiau.

Dėlto, kad šis metodas suteikia ribotą detalių reikšmių kiekį, metodas pasiteisina tik tuomet, jeigu yra žinomas tikslus elementų pasikartojančioje grupėje skaičius ir šis kiekis yra statiškas ir nesikeičiantis. Esant tokiai situacijai, lentelės gali būti sujungiamos. Jeigu nėra žinomas tikslus detalių įrašų kiekis, saugant juos kaip stulpelius gali neatsipirkti. Tai gali lemti, kad pasirinktas stulpelių kiekis yra per didelis ar per mažas. Be to, prarandamas lankstumas detalių įrašų atžvilgiu – jeigu duomenys dažnai keičiasi, po tokios realizacijos užklausos sudėtingėja ir metodas nepasiteisina.

Pridedant informacijos kopijas iš vienos lentelės į kitą, bendras lentelių kiekis duomenų bazėje nepakinta. Tačiau taip sukuriama daugiau elementų, todėl lentelė didėja. Vadinasi, duomenys dubliuojasi ir gali tapti nevientisi bei atsirasti duomenų anomalijos. Dėl to šis metodas netinka, jei duomenys keičiasi dažnai. Informacija, kuri kartojama kitose lentelėse, turi būti pakankamai statiška.

Metodas tinka tokiose situacijose, kai užklausose atitinkamos lentelės naudojamos sąryšiui nustatyti, o ne atvaizdavimui. Kitaip tariant – lentelė užklausoje yra sujungiama su kita lentele, bet rezultate atvaizduojama tik maža tos lentelės dalis (pavyzdžiui, vienas stulpelis) ar informacija iš lentelės net neatvaizduojami (lentelė naudojama tik tam, kad būtų surastas ryšys su kita lentele). Jeigu tokios lentelės atitinkami duomenys (tie, kurie naudojami sujungimuose ar atvaizdavimui) yra dubliuojami kitoje lentelėje, tuomet užklausose sumažėja sujungimų kiekis (o tuo pačiu ir skenuojamas mažesnis lentelių kiekis). Taip yra sudaromas trumpesnis duomenų pasiekimo kelias, todėl užklausos vykdomos greičiau.

Skaičiuojamuosius laukus galima saugoti atskiroje lentelėje arba naujame stulpelyje egzistuojančioje lentelėje. Jeigu duomenys saugomi naujoje lentelėje, tuomet bendras lentelių kiekis padidėja. Jeigu saugoma egzistuojančioje lentelėje – lentelių kiekis nepakinta, o lentelė padidėja.

Bet kuriuo atveju, skaičiuojamųjų laukų saugojimas lemia įrašų kiekio didėjimą duomenų bazėje. Be to, tai lemia paprastesnį duomenų gavimą – užklausose nereikia ieškoti duomenų iš galima daugiau nei vienos lentelės (ir naudoti sujungimus) bei atlikti matematinių veiksmų (dažniausiai agregatų) su lentelės duomenimis. Taigi, šiuo atveju pakanka tik skenuoti duomenis, nes jau jie būna apskaičiuoti. Žinoma, duomenys turi būti paruošti – tai lemia, kad jie turi būti reguliariai atnaujinami ir sinchronizuojami su baziniais duomenimis. Toks procesas sukuria papildomų priežiūros darbų bei kaštų. Todėl negalima teigti, jog toks metodas tinka visuose situacijose, kai yra skaičiuojami laukai ir reikalingas greitas ar dažnas jų atvaizdavimas. Prieš taikant metodą taip pat reikia įvertinti ir atsiradusius naujus palaikymo kaštus.

1.3. Denormalizacijos įtakos duomenų bazei analizė

Literatūros šaltiniuose denormalizacija yra nurodoma kaip sprendimas greitaveikai gerinti. Tai pateikiama kaip faktas, tačiau išsamesnių analizių beveik nėra. Denormalizacijos įtaka nagrinėjama teoriniu aspektu – pavyzdžiui, kad duomenų bazė padidėja, kadangi sukuriami duomenų dublikatai. Ar kad skaitymo užklausų vykdymo laikai sutrumpėja, kadangi sudaromi trumpesni duomenų pasiekimo keliai. Tačiau šaltiniuose stokojama duomenų bazės parametrų palyginimų skaitinėmis reikšmėmis. Retai nurodoma kaip ir kiek kartų pasikeičia duomenų bazės dydis, duomenų kiekis bei kaip tiksliai paveikiami užklausų vykdymo laikai. Palyginimas dažniausiai būna tik tarp normalizuotos ir denormalizuotos schemos. Be to, denormalizuota schema nėra detalizuojama, todėl neaišku, kas joje buvo padaryta ir kokio lygio denormalizacija pritaikyta. Tai lemia, kad nagrinėjama nebent denormalizacijos įtaka duomenų bazei, o konkretaus metodo įtaka nagrinėjama retai [19, 29].

D.Shasha ir P.Bonnet [29] pateikė normalizuotos ir denormalizuotos schemos palyginimą. Šiuo atveju matuojamas užklausų, kuriose skaitomi duomenys, pralaidumas (angl. *throughput*), kuris nurodo įvykdytų užklausų kiekį per sekundę. Denormalizuotu atveju pralaidumas yra 30 % geresnis. Tai lemia, kad denormalizuotu atveju užklausos yra vykdoma greičiau. Tokia situacija paaiškinama tuo, jog normalizuotos schemos atveju užklausoje turi būti naudojami 4 sujungimai, o denormalizuotos schemos atveju užklausoje yra vykdomas tik lauko skaitymas.

D.Shasha ir P.Bonnet [29] taip pat pateikė palyginimą, kai lentelėje yra saugomi skaičiuojami laukai. Toks sprendimas reikalauja trigerių, kurie pasikeitus atitinkamiems duomenis, atnaujina apskaičiuotus laukus. Turint įterpimo užklausas, trigerių vykdymas sulėtina įterpimo užklausas apie 60 %, todėl įtaką užklausų vykdymo laikui yra neigiama. Skaitymo užklausų vykdymo laikas sumažėja 200 kartų. Kaip ir minėta anksčiau, geresnė greitaveika gaunama, kadangi užklausoje vykdomas tik skenavimas, kol kitu atveju atliekami lentelių sujungimai.

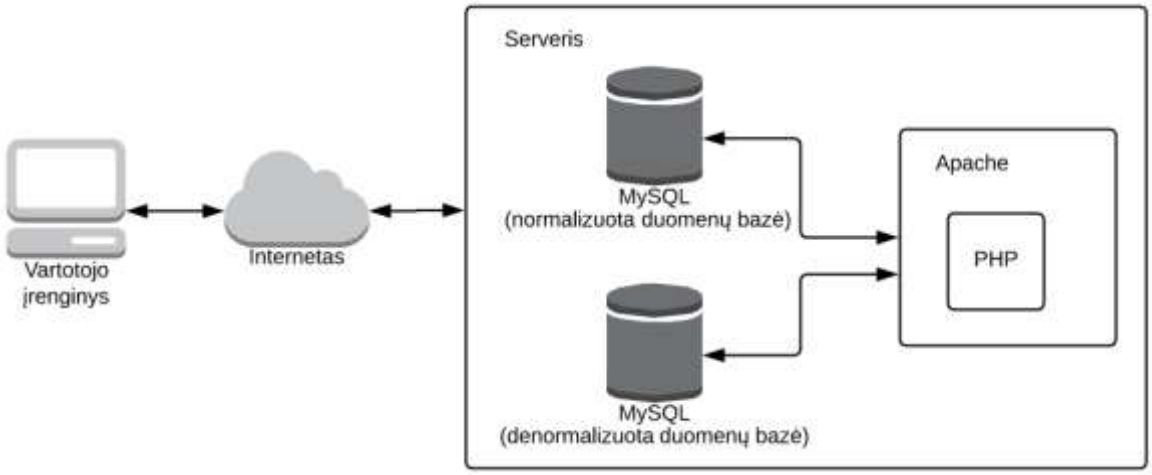
M. Zaker, S. Phon-Amnuaisuk, S. Haw [19] eksperimente nagrinėtos dvi duomenų bazės. Denormalizuotoje duomenų bazėje duomenų yra beveik 2 kartus daugiau negu normalizuotoje. Eksperimente testuojamos vienmatės (angl. *one-dimensional*) bei daugiamatės (angl. *multi-dimension*) užklausos. Vienamatės užklausos yra tokios, kuriose atliekami veiksmai vienos lentelės rėžiuose. Daugiamatėse užklausose skenuojamos bent dvi lentelės. Vienmatėse užklausose denormalizacijos atveju vykdymo laikai yra apie 1,5 kartus ilgesni. Taip yra todėl, jog lentelės yra didesnės ir turi būti skenuojama daugiau įrašų. Daugiamatėse užklausose vykdymo laikai yra geresni. Pirmoji užklausa denormalizacijos atveju pagreitėja 5190 kartus. Viena iš priežasčių, kodėl pasiekiamas toks rezultatas yra toks, jog nebereikalinga sujungimo operacija užklausoje. Kita užklausa, nagrinėja situaciją, kai saugomi skaičiuojami laukai lentelėje. Tokios užklausos vykdymo laikas pagreitėja apie 10,8 kartus.

2. DUOMENŲ BAZĖS MODELIO OPTIMIZACIJOS METODŲ PROJEKTAVIMAS

Darbo esmė yra keisti duomenų bazės struktūrą ir stebėti, kaip keičiasi įvairūs duomenų bazės parametrai, tokie, kaip duomenų kiekis, lentelių dydis ir greitaveika. Šiame skyriuje pateikiamas eksperimento kontekstas. Taip pat yra iškeliami reikalavimai optimizacijos procesui ir rezultatams. Galiausiai, remiantis analizės skyriuje rastais optimizacijos metodais, sudaroma reliacinės duomenų bazės modelio optimizavimo metodika. Į šią metodiką įeina pasirinktos denormalizacijos technikos bei jų projektavimas. Kadangi optimizuojama keliais būdais, todėl kiekvieno metodo projektavimas yra išskaidytas į atskirus poskyrius.

2.1. Kontekstas

Eksperimento kontekstinė diagrama pateikta 2.1 pav. Vartotojas, naudodamasis interneto prieiga, jungiasi prie serverio. Serveryje yra dvi duomenų bazės (*MySQL*) bei *Apache* su *PHP*. Viena duomenų bazė yra normalizuota (pradinė), o kita yra su denormalizuotomis lentelėmis. Šiame darbe optimizuojama duomenų bazės dalis, o visa kita yra priimama kaip yra. Denormalizuotoji duomenų bazė yra gaunama atlikus optimizaciją.



2.1 pav. Konteksto diagrama

2.2. Reikalavimai

Eksperimento eigai bei rezultatams yra iškelti reikalavimai:

- atliekami tyrimai neturi keisti esamo sistemos funkcionalumo;
- naudoti kelių tipų *SELECT* užklausas (su sąlygomis ir be jų);
- tyrimas atliekamas su trijų tipų duomenų kiekiais – mažais, vidutiniais ir dideliais. Su visais kiekiais turi būti pasiektas optimalus rezultatas;
- siekiamas trumpesnis užklausų vykdymo laikas;
- po optimizacijos duomenų logika turi išlikti (negali dingti duomenys ar ryšiais tarp jų);
- sprendimai turi leisti peržiūrėti atitinkamus duomenis (sudaryti ataskaitas);
- tyrimo sprendimai turi būti atliekami tik su *SQL* duomenų bazėmis.

2.3. Optimizacijos metodų projektavimo eiga

Analizuojant galimas denormalizacijos technikas šaltiniuose, pastebėta, kad dažniausiai naudojami metodai yra lentelių sujungimas, lentelių (horizontalus ir vertikalus) skaidymas, pasikartojančios informacijos pridėjimas, skaičiuojamųjų laukų saugojimas bei sąrašai lentelėje. Atsižvelgiant į esamą duomenų bazę, modelio optimizacijai pasirinkti 4 būdai:

1. pasikartojančių stulpelių saugojimas lentelėje;
2. kelių lentelių sujungimas į vieną lentelę;
3. skaičiuojamų laukų saugojimas lentelėje;
4. pasikartojančios informacijos saugojimas lentelėje.

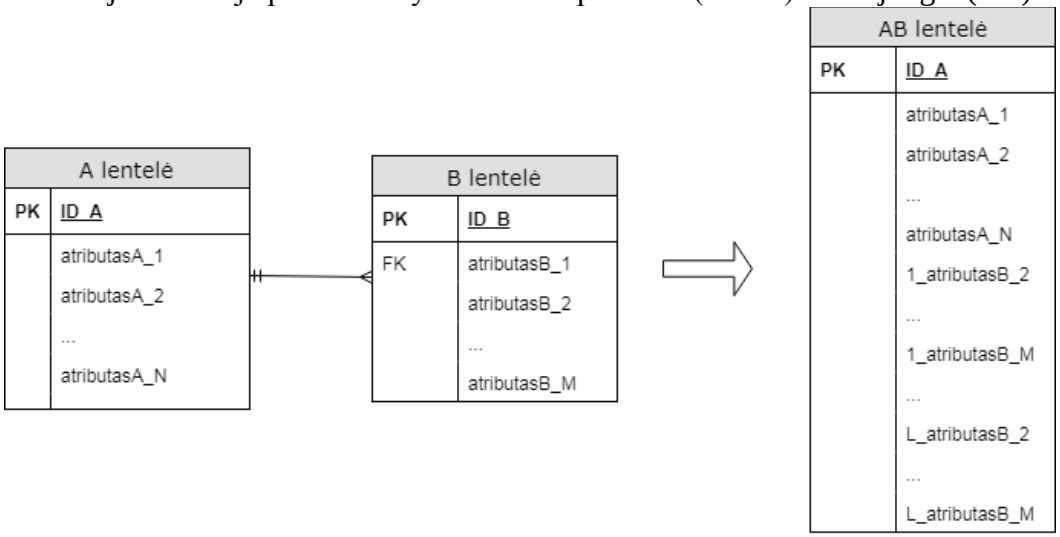
Supaprastinimui, kiekviena iš metodikų yra vadinama pagal tai, kokiu eiliškumu buvo išvardyta anksčiau esančiame sąraše – pavyzdžiui, pasikartojančių stulpelių saugojimo lentelėje metodas traktuojamas kaip metodas-1, lentelių sujungimas – metodas-2, skaičiuojamų laukų saugojimas – metodas-3, o pasikartojančių laukų saugojimas lentelėse įvardinamas kaip metodas-4. Toliau kiekvienas iš metodų yra pateiktas atskirame poskyryje, kiekviename aprašant atitinkamos technikos pagrindinius aspektus.

Projektavimas iliustruojamas naudojantis esybių – ryšių diagramomis (angl. *Entity Relationship Diagram*). Tokia diagrama naudojama tam, kad būtų parodyta lentelių struktūra (stulpelių pavadinimai, pirminiai ir išoriniai raktai) bei sąryšiai tarp lentelių.

2.3.1. Metodo-1 projektavimas

Pirmas duomenų bazės optimizacijos metodas yra pasikartojančių stulpelių saugojimas lentelėje. Tai tokia technika, kai vietoje 1:N sąryšio, vienoje lentelėje yra saugomas pasikartojantis sąrašas. Kitaip tariant, šiuo metodu kelios lentelės yra sujungiamos į vieną platesnę lentelę.

2.2 pav. pateikta metodo projektavimo schema. Lentelėje *PK* žymi pirminį raktą, o *FK* – išorinį. Kairėje paveikslo pusėje yra pateikiamas pradinės duomenų bazės fragmentas, o dešinėje – modifikuotas. Šioje schemoje pateiktos trys lentelės – pradinės (A ir B) bei sujungta (AB).



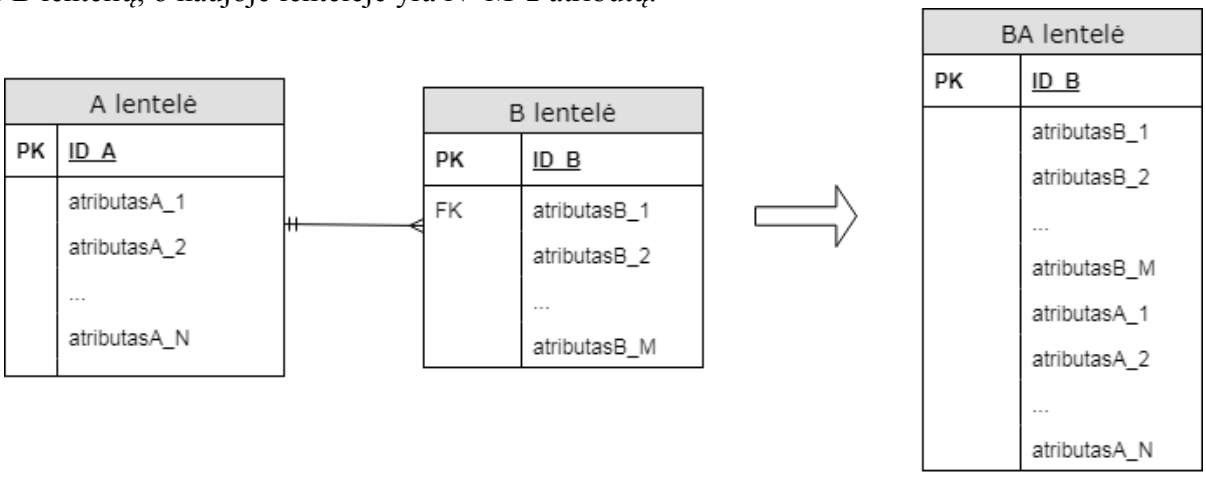
2.2 pav. Metodo-1 projektavimo schema

Kiekviena lentelė turi skirtingą kiekį atributų (A lentelė jų turi N vienetų, o B – M atributų). Po sujungimo nebelieka sąryšio, egzistavusio tarp A ir B lentelių, o naujoje lentelėje žymiai padidėja atributų skaičius. Kadangi sujungimas vykdomas A lentelės atžvilgiu, todėl A lentelė informacija yra perkeliama į naują lentelę (pirminiu raktu tampa *ID_A* atributas, o kiti A lentelės atributai perrašomi). B lentelės duomenys yra saugomi kaip pasikartojantis sąrašas naujoje lentelėje. Ši informacija yra kartojama L kartų (L – iš anksto numatytas skaičius, kuris nurodo, kiek pasikartojančių reikšmių saugoma lentelėje). Kadangi sąryšio nebelieka, todėl atributas, kuris buvo išorinis raktas B lentelėje, nebenaudojamas.

2.3.2. Metodo-2 projektavimas

Kitas metodas yra lentelių sujungimas į vieną. Jis yra panašus į pirmąjį – skiriasi tik tai, kurios lentelės atžvilgiu vykdomas sujungiamas. Jeigu abiem atvejais turimos dvi tokios pačios lentelės (A ir B), tuomet pirmojo metodo atveju pagrindiniai duomenys yra A lentelės informacija, o šiuo atveju – B lentelės.

2.3 pav. pateikta antrojo metodo projektavimo schema. Kaip ir pirmuoju atveju, schemoje pateiktos trys lentelės – pradinės (A ir B) bei sujungta (BA). Kiekviena lentelė turi skirtingą atributų kiekį (A lentelė jų turi N vienetų, o B – M atributų). Po sujungimo nebelieka sąryšio, egzistavusio tarp A ir B lentelių, o naujoje lentelėje yra N+M-1 atributų.



2.3 pav. Metodo-2 projektavimo schema

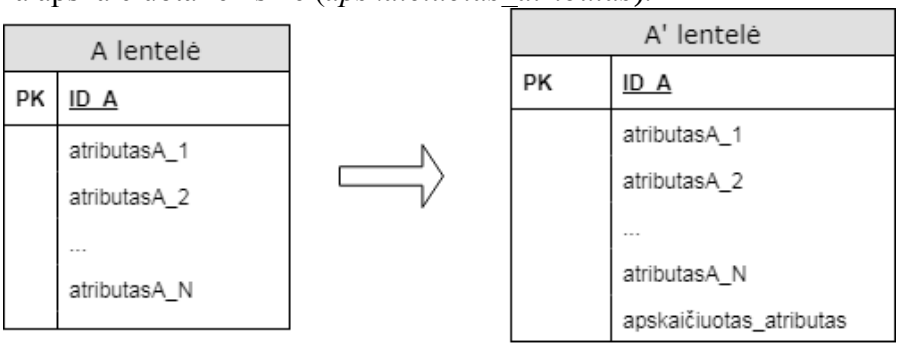
Gautoje lentelėje yra bent vienu atributu mažiau negu abiejų lentelių atributų suma. Taip yra todėl, kad pradiniu atveju duomenų bazėje tarp lentelių turėjo egzistuoti vienodi atributai, o sujungus pakanka turėti tik vieną tokį atributą. Šiuo atveju A lentelėje toks atributas yra *ID_A*, B lentelėje – *atributasB_1*, o naujoje lentelėje (BA) šis atributas yra *atributasB_1*. Toks sujungimas lemia, kad sumažėja pirminių raktų kiekis. Kiti atributai iš abiejų lentelių yra išsaugomi naujoje lentelėje.

Sujungti galima tokias lenteles, kurios tarpusavyje susietos (pirminio ir išorinio raktų pora ar vienodais atributais). Be to, geriausia sujungti tokias lenteles, kurios yra beveik neatsiejamos (dažnai naudojamos kartu). Kitu atveju duomenų gavimas gali trukti ilgiau, kadangi lentelėje saugoma daugiau atributų ir lentelės skenavimas sulėtėja.

2.3.3. Metodo-3 projektavimas

Trečiasis metodas yra skaičiuojamųjų laukų saugojimas lentelėje. Šiuo atveju iš tam tikrų reikšmių išskaičiuota vertė (pavyzdžiui, maksimumas, minimumas, suma ar vidurkis) yra išsaugoma prie kiekvieno įrašo. Tokiam agregatui yra išskiriama viena ar daugiau stulpelių lentelėje.

2.4 pav. pateikta šio metodo projektavimo schema. Kairėje paveikslo pusėje yra pateikiamas pradinės duomenų bazės fragmentas (A lentelė), o dešinėje – modifikuotas (A‘ lentelė). Iš pradinės lentelės visi atributai yra perrašomi. Lyginant su pradine lentele, pakeistoji padidėja, kadangi joje yra papildomai išsaugoma apskaičiuota reikšmė (*apskaičiuotas_atributas*).

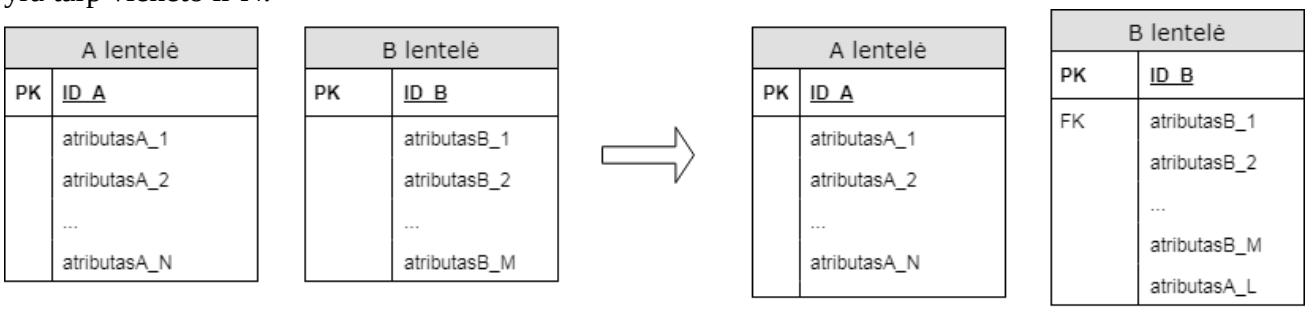


2.4 pav. Metodo-3 projektavimo schema

2.3.4. Metodo-4 projektavimas

Ketvirtojo metodo atveju lentelėje yra saugoma pasikartojanti informacija. Tai daroma tam, kad būtų išvengta dažnų sujungimų tarp lentelių, kai iš vienos lentelės naudojama tik maža dalis informacijos (pavyzdžiui, vienas stulpelis). Tokiu atveju nurodytas stulpelis yra pakartotinai išsaugomas ir kitoje lentelėje. Taip išvengiama sujungimų tarp lentelių, kai iš vienos lentelės gaunama tik maža dalis duomenų.

2.5 pav. pateikta šio metodo projektavimo schema. Kairėje pusėje pateiktos pradinės lentelės, o dešinėje – po metodo pritaikymo. Šiuo atveju lentelė A lieka nepakitusi, o B lentelė papildoma atributu *atributasA_L*. Nurodytas atributas yra A lentelės atributo kopija. Šiuo atveju L nurodo skaičių, kuris yra tarp vieneto ir N.



2.5 pav. Metodo-4 projektavimo schema

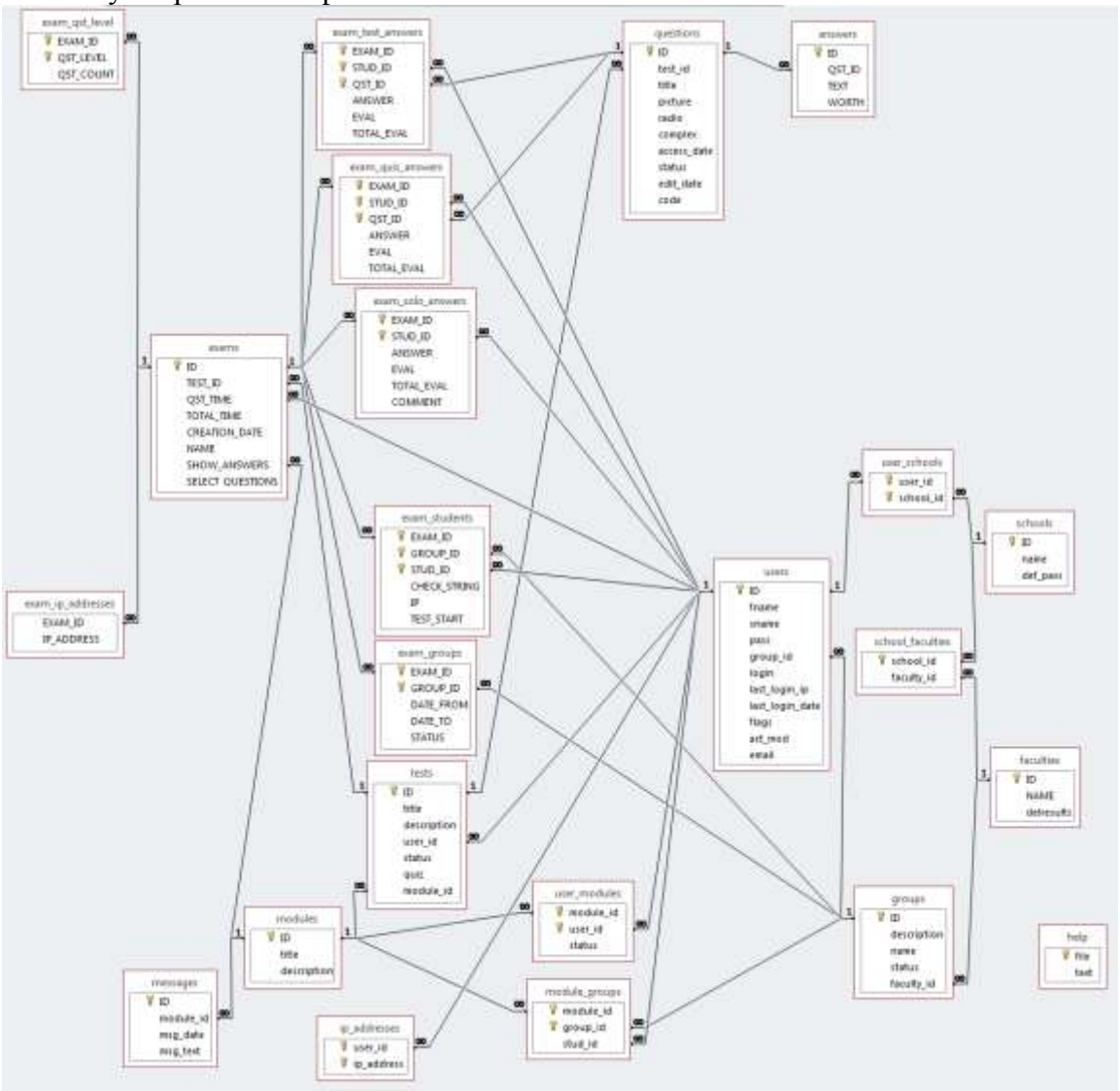
3. DUOMENŲ BAZĖS MODELIO OPTIMIZACIJOS METODŲ REALIZACIJA

Šiame skyriuje pateikiama pasirinktų optimizacijos metodų realizacija. Kaip minėta projektavimo skyriuje, metodai įvardinami kaip metodas-1 (sąrašų saugojimas lentelėje), metodas-2 (kelių lentelių sujungimas į vieną), metodas-3 (skaičiuojamų laukų išsaugojimas lentelėje) ir metodas-4 (pasikartojančios informacijos saugojimas lentelėje). Kiekvienas jų išskaidytas į atskirą poskyrį, kuriame atitinkamai yra apžvelgiama kiekvieno metodo realizacija. Poskyriuose parodoma, kokios lentelės ir duomenys buvo prieš optimizaciją ir kaip tai pasikeitė. Be to, yra pateikiami žingsniai, nurodantys, kas buvo daroma ir kas keičiama duomenų bazėje, realizuojant kiekvieną sprendimą.

Prieš pateikiant realizacijas, visų pirmą skyriuje iliustruojama duomenų bazės, su kuria dirbama, schema. Šis poskyris skirtas tam, kad būtų supažindinta su pradiniais duomenimis. Be to, šiame skyriuje taip pat trumpai aprašomos naudojamos technologijos.

3.1. Pradinė duomenų bazė

Šiame darbe dirbama su veikiančia sistema, kuri skirta testavimui (studentų žinių patikrinimui). Šios sistemos duomenų bazę sudaro 24 lentelės: *answers; exams; exam_groups; exam_ip_addresses; exam_qst_level; exam_quiz_answers; exam_solo_answers; exam_students; exam_test_answers; faculties; groups; help; ip_addresses; messages; modules; module_groups; questions; questions_no_calc; schools; school_faculties; tests; users; user_modules; user_schools*. Lentelės ir jų tarpusavio ryšiai pateikti 3.1 pav.



3.1 pav. Pradinė duomenų bazės schema

Iš visų lentelių, *answers*, *questions* ir *exam_test_answers* yra didžiausios bei vienos iš svarbiausių. Juose yra daugiausiai įrašų bei jų užimama vieta yra didžiausia. Šios lentelės jungiasi su daugeliu lentelių. Tačiau atsižvelgiant į sistemos paskirtį, *users*, *exams* ir *tests* lentelės yra svarbiausios – juose saugoma informacija apie pačius vartotojus bei egzaminus ir testus.

Lentelėje *questions* saugomi su klausimais susiję duomenys (žr. 3.1 lent.). Lentelėje *answers* saugomi klausimų atsakymai (žr. 3.2 lent.). Lentelėje *tests* saugomi su testais susiję duomenys (žr. 3.3 lent.). Lentelėje *exams* saugomi su egzaminu susiję duomenys (žr. 3.4 lent.). Lentelėje *exam_test_answers* saugoma studento egzamino informacija (žr. 3.5 lent.). Lentelėje *users* saugoma informacija apie vartotojus – tiek apie studentus, tiek apie dėstytojus (žr. 3.6 lent.).

3.1 lent. *Questions* lentelės struktūra

Laukas	id	test_id	title	picture	radio	complex	access_date	status	edit_date	code
Tipas	int(11)	int(11)	varchar (1200)	varchar (255)	tinyint (3)	tinyint (3)	timestamp	varchar (1)	timestamp	varchar (1200)
Papildoma informacija	Pirminis raktas	Indeksas				Indeksas		Indeksas		

3.2 lent. *Answers* lentelės struktūra

Laukas	id	qst_id	text	worth
Tipas	int(11)	int(11)	mediumtext	tinyint(3)
Papildoma informacija	Pirminis raktas	Indeksas		

3.3 lent. *Tests* lentelės struktūra

Laukas	id	title	description	user_id	status	quiz	module_id
Tipas	int(11)	varchar(255)	mediumtext	int(11)	varchar(1)	int(11)	int(11)
Papildoma informacija	Pirminis raktas			Indeksas	Indeksas	Indeksas	Indeksas

3.4 lent. *Exams* lentelės struktūra

Laukas	id	test _id	qst_time	total _time	creation _date	name	show_ answers	select_ questions
Tipas	int(11)	int(11)	int(11)	int(11)	timestamp	varchar(255)	int(1)	int(1)
Papildoma informacija	Pirminis raktas	Indeksas						

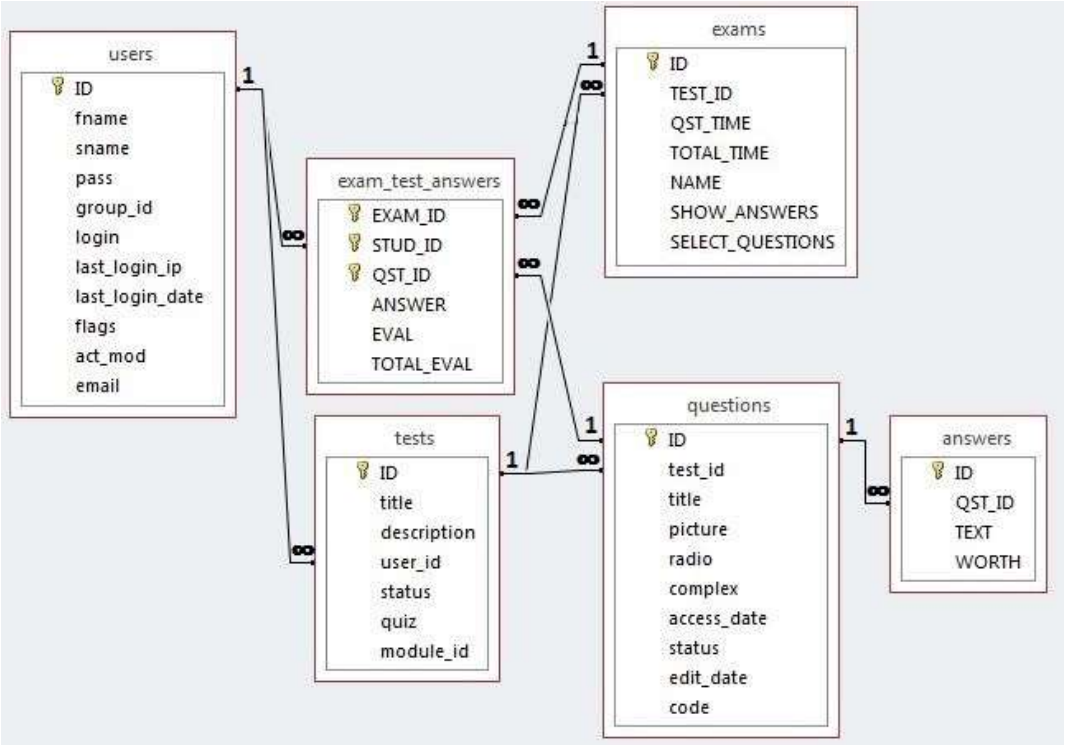
3.5 lent. *Exam_test_answers* lentelės struktūra

Laukas	exam_id	stud_id	qst_id	answer	eval	total_eval
Tipas	int(11)	int(11)	int(11)	int(11)	tinyint(3)	tinyint(3)
Papildoma informacija	Pirminis raktas	Pirminis raktas	Pirminis raktas, Indeksas			

3.6 lent. *Users* lentelės struktūra

Laukas	id	fname	sname	pass	group_id	login	last_ login_ip	last_login _date	flags	act_ mod	email
Tipas	int(10)	varchar (30)	varchar (30)	varchar (50)	int (11)	varchar (60)	varchar (15)	timestamp	varchar (10)	int (11)	varchar (60)
Papildoma informacija	Pirminis raktas				Indeksas	Indeksas					

Anksčiau išvardintose lentelėse yra pagrindiniai sistemos duomenys, todėl atliekant šį darbą yra ignoruojamos kitos lentelės. Principinė šių lentelių schema pateikta 3.2 pav. Kai kurie ryšiai yra eliminuoti. Tokia schema naudojama tam, kad būtų supaprastinta duomenų bazės schema ir jos suvokimas. Optimizacijos sprendimai realizuoti įsivaizduojant, kad duomenų bazę sudaro būtent šios lentelės ir kad duomenų bazė yra būtent tokia.



3.2 pav. Principinė pradinės duomenų bazės svarbiausių lentelių schema

3.2. Naudojamos technologijos

Realizacijai naudotas *WAMP (Windows Apache MySQL ir PHP)* programų paketas bei *PhpMyAdmin*, esantis jo viduje. *WAMP* programų paketas yra skirtas laikyti ir paleisti žiniatinklio (angl. *web*) serverį kompiuteriuose, kuriuose yra įrašyta *Windows* operacine sistema. Šis paketas sudarytas iš *Apache* žiniatinklio serverio, *MySQL* duomenų bazės valdymo sistemos bei *PHP*.

PhpMyAdmin yra įrankis naudojamas *MySQL* duomenų bazių valdymui. Kaip ir daugelis kitų duomenų bazių valdymo įrankių, jis suteikia galimybę manipuliuoti lentelėmis ir įrašais jose bei vykdyti *SQL* užklausas.

Darbui pasirinkta būtent *MySQL* duomenų bazė, kadangi ji yra viena iš populiariausių ir plačiai išplitusių atviro kodo duomenų bazių, kuriuos manipuliuoja duomenų bazes ir gali būti prieinamos internetinėse svetainėse. Daugelis interneto svetainių naudoja *MySQL* kaip loginę dalį (angl. *backend*) duomenų saugyklai.

MySQL yra suderinama su įvairiais varikliais ir sąsajomis ir yra viena iš labiausiai subrendusių duomenų bazių rinkoje. Jos populiarumas internetinėse programose yra glaudžiai susijęs su *PHP* programavimo kalbos populiarumu, kuri yra dažnai naudojama kartu su *MySQL* [30].

Šiame darbe *MySQL* naudojama duomenų bazei saugoti. Duomenų redagavimui skirtas *phpMyAdmin* įrankis. Naudojama *PHP* programavimo kalba.

3.3. Optimizacijos metodų realizacija

Norint išlaikyti sistemos duomenų konfidencialumą, tam tikros duomenų reikšmės nėra pateikiamos. Duomenys yra abstraktinami – pavyzdžiui, vietoje tikrojo klausimo rašoma *klausimas*. Esant ne vienai reikšmei, prie pavadinimo yra pridedamas skaičius (pavyzdžiui, *klausimas1*, *klausimas2*).

Kaip ir ankstesniame skyriuje, realizacija pateikiama poskyriais pagal optimizacijos metodus. Kiekviename poskyryje pateikiama duomenų struktūra prieš optimizaciją ir po jos. Be to, parodoma realizacijos eiga. Realizuojama *MySQL* duomenų bazėje rašant atitinkamas *SQL* užklausas. Skyriuje pateiktose lentelėse šviesiai žaliai nudažyti stulpeliai nurodo, kad atitinkamas stulpelis buvo pridėtas, o nudažytas tamsesnio atspalvio raudona spalva ir užbrūkšniuotas – kad pašalintas.

3.3.1. Metodo-1 realizacija

Pirmas duomenų bazės optimizacijos metodas yra kelių lentelių sujungimas į vieną. Sujungimas vykdomas saugant vienos lentelės duomenis kaip pasikartojantį sąrašą kitoje lentelėje. Šiuo atveju sujungimui pasirinktos *questions* ir *answers* lentelės. Šios lentelės dažniausiai yra ištraukiamos kartu – pavyzdžiui, kai norima peržiūrėti klausimą (tuomet rodomi ir atsakymai), kai norima surasti visus testo ar egzamino klausimus (atsakymai rodomi irgi), kai vartotojai nori pamatyti egzamino metu atsakytus klausimus.

Abi šios lentelės turi daug įrašų ir užima daug vietos, be to, traukiant duomenis iš duomenų bazės užklausose tenka naudoti sujungimus. Sujungus lenteles į vieną, sumažinama lentelių kiekis duomenų bazėje bei sumažinamas sujungimų kiekis užklausose. Sujungimų eliminavimas užklausose greitina užklausos vykdymo laiką.

Šio būdo esmė, kad dvi lentelės (*questions* ir *answer*) yra sujungiamos ir gaunama viena lentelė (*questionsWithAnswers*). *Answers* lentelės struktūra bei pavyzdiniai duomenys pateikti 3.7 lent., o *questions* – 3.8 lent.

3.7 lent. *Answers* lentelės pavyzdiniai duomenys prieš optimizaciją

id	qst_id	text	worth
1	6	Atsakymas6_1	3
2	6	Atsakymas6_2	0
3	9	Atsakymas9_1	10
4	11	Atsakymas11_1	0

3.8 lent. *Questions* lentelės pavyzdiniai duomenys prieš optimizaciją

id	test_id	title	picture	radio	complex	access_date	status	edit_date	code
6	1	Klausimas6	NULL	0	2	2012-10-22 11:01:52	A	2006-07-14 16:53:05	NULL
9	2	Klausimas9	NULL	1	0	2011-05-21 09:00:55	N	2006-07-14 16:53:05	NULL
11	1	Klausimas11	NULL	1	2	2010-10-20 14:35:51	N	2006-07-14 16:53:05	NULL

Jungiant lenteles buvo kuriamos tarpinės lentelės. Jos padėjo suvokti duomenis ir priimti atitinkamus sprendimus (pavyzdžiui, kiek sukurti stulpelių atsakymams). Lentelių sujungimo eiga:

1. *answers* lentelėje pridedamas naujas stulpelis, kuriame išsaugomas skaitiklis pagal klausimo identifikatorių (stulpelį *QST_ID*). Iš 3.7 lent. gauta lentelė su nauju stulpeliu pateikta 3.9 lent. Stulpelyje saugoma skaitiklio reikšmė parodo, su kiek atsakymų atitinkamas klausimas yra susietas. Jeigu skenuojant atsakymų lentelę, klausimo identifikatorius pasirodo pirmą kartą, skaitiklio reikšmė yra vienetas, jeigu antrą kartą – dvejetas, ir taip tiek kartų, kiek yra atsakymų pagal atitinkamą klausimo identifikatorių. Toks skaitiklis padeda nustatyti, kiek maksimaliai klausimas turi atsakymų. Su esama duomenų baze, maksimalus atsakymų kiekis yra aštuoni;

3.9 lent. *Answers* lentelė papildyta skaitikliu pagal klausimo identifikatorių

id	qst_id	text	worth	counter
1	6	Atsakymas6_1	3	1
2	6	Atsakymas6_2	0	2
3	9	Atsakymas9_1	10	1
4	11	Atsakymas11_1	0	1

2. pirmame žingsnyje nustatyta, kad 8 atsakymai yra maksimalus atsakymų kiekis vienam klausimui. Žinant tokią informaciją, *answers* lentelę galima projektuoti taip, kad vienoje eilutėje būtų visi atsakymai pagal klausimo identifikatorių (žr. 3.10 lent.). Taip gaunama nauja atsakymų lentelė, kurioje lyginant su originalia lentele (žr. 3.7 lent.) yra pridėti papildomi 16 laukų (po 8 atsakymui ir po 8 atsakymo vertei). Be to, lyginant su originalia lentelė, šioje lentelėje yra pašalinami stulpeliai *ID, TEXT* ir *WORTH*. Taip yra todėl, kad atsakymo identifikatorius nebereikalingas, o *TEXT* ir *WORTH* pasikeičia į *answer_n* ir *worth_n* laukus (šiuo atveju *n* yra skaičius, nurodantis atsakymo numerį atitinkamam klausimui). Tokiu būdu, naujoje lentelėje padidėja įrašų kiekis, o klausimo atsakymai yra pateikti vienoje eilutėje;

3.10 lent. *Answers* lentelės struktūros keitimas

id	qst_id	text	worth	answer_1	worth_1	answer_2	worth_2	answer_3	worth_3	answer_4	worth_4	answer_5	worth_5	answer_6	worth_6	answer_7	worth_7	answer_8	worth_8
	6			Atsakymas 6_1	3	Atsakymas 6_2	0	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0
	9			Atsakymas 9_1	10	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0
	11			Atsakymas 11_1	0	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0

3. antrame žingsnyje sukurta lentelė (3.10 lent.) su klausimų lentele (3.8 lent.) turi ryšį 1:1 (viena eilutė vienoje lentelėje atitinka tik vieną eilutę kitoje lentelėje). Sujungta lentelė pateikta 3.11 lent. Rezultatas yra klausimų su atsakymais lentelė, kurioje lyginant su originalia klausimų lentele (žr. 3.8 lent.) yra pridėti papildomi 16 laukų (po 8 atsakymui ir po 8 atsakymo vertei).

3.11 lent. *Questions* lentelė papildyta atsakymais

id	test_id	title	picture	radio	complex	access_date	status	edit_date	code	answer_1	worth_1	answer_2	worth_2	answer_3	worth_3	answer_4	worth_4	answer_5	worth_5	answer_6	worth_6	answer_7	worth_7	answer_8	worth_8
6	1	Klausimas6	NULL	0	2	2012-10-22 11:01:52	A	2006-07-14 16:53:05	NULL	Atsakymas6_1	3	Atsakymas6_2	0	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0
9	2	Klausimas9	NULL	1	0	2011-05-21 09:00:55	N	2006-07-14 16:53:05	NULL	Atsakymas9_1	10	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0
11	1	Klausimas11	NULL	1	2	2010-10-20 14:35:51	N	2006-07-14 16:53:05	NULL	Atsakymas11_1	0	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0	NULL	0

Atlikus anksčiau įvardintus žingsnius, iš dviejų lentelių gaunama viena lentelė. Lyginant šią lentelę su originalia klausimų lentele, pastaroji yra platesnė ir didesnė, nes turi daugiau stulpelių. Iš viso vienai klausimo eilutei prisidėjo 2*8 laukai – 8 *mediumtext* tipo laukai atsakymams ir 8 *tinyint(3)* tipo atsakymo vertei. Lentelei nustatomi *questions* lentelės indeksai – *testi_id, complex* ir *status.*

Rezultato esmė, kad vietoje dviejų siauresnių lentelių, gaunama viena platesnė. Tačiau tai reiškia, jog visi duomenys saugomi vienoje vietoje ir užklausose nebereikia naudoti sujungimų. Kadangi po sujungimo nebeegzistuoja atsakymų identifikatorius, kitos lentelės, kuriose buvo saugomas atsakymas, pasikeičia. *Exam_test_answers* lentelėje po pakeitimų saugomas nebe atsakymo identifikatorius, bet atsakymo numeris sąraše. Kitos lentelės lieka nepakitusios.

3.3.2. Metodo-2 realizacija

Sekantis optimizacijos metodas yra kelių lentelių sujungimas į vieną. Sujungimui pasirinktos *questions* ir *answers* lentelės. Technika analogiška pirmajai – skiriasi tik tai, kuri lentelė tampa pagrindine sujungimo metu. Pirmuoju atveju dominuojanti lentelė buvo *questions*, šiuo atveju – *answers*. Metodo-1 realizacija lemia, jog prarandamas lankstumas atsakymų atžvilgiu – naudojantis tik užklausomis negalima rasti specifinių (pavyzdžiui, teisingų) atsakymų. Šiuo atveju atsakymais galima manipuliuoti taip, kaip buvo daroma ir pradinėje duomenų bazėje.

Šio būdo esmė ta, kad dvi lentelės (*questions* ir *answers*) yra sujungiamos ir gaunama viena lentelė (*answersWithQuestions*). *Answers* lentelės struktūra bei pavyzdiniai duomenys pateikti 3.7 lent., o *questions* – 3.8 lent.

Lentelės sujungimas vykdomas perkeliant duomenis iš abiejų lentelių į naują. *Answers* lentelės duomenys perkeliami nieko nemodifikuojant. Atsižvelgiant į *QST_ID* (*answers* lentelė) ir *id* (*questions* lentelė), kiekvienas klausimas yra įrašomas į naują lentelę prie atitinkamo atsakymo (žr. 3.12 lent.). Kadangi klausimo identifikatorius yra saugoms abiejose pradinėse lentelėse, po sujungimo identifikatorius iš klausimų lentelės nebereikalingas. Tokiu būdu, į naują lentelę perkeliami visi klausimo duomenys, išskyrus identifikatorių (*id* stulpelį).

3.12 lent. *Answers* lentelė papildyta klausimais

id	qst_id	text	worth	test_id	title	picture	radio	complex	access_date	status	edit_date	code
1	6	Atsakymas6_1	3	1	Klausimas6	NULL	0	2	2012-10-22 11:01:52	A	2006-07-14 16:53:05	NULL
2	6	Atsakymas6_2	0	1	Klausimas6	NULL	0	2	2012-10-22 11:01:52	A	2006-07-14 16:53:05	NULL
3	9	Atsakymas9_1	10	2	Klausimas9	NULL	1	0	2011-05-21 09:00:55	N	2006-07-14 16:53:05	NULL
4	11	Atsakymas11_1	0	1	Klausimas11	NULL	1	2	2010-10-20 14:35:51	N	2006-07-14 16:53:05	NULL

Po įterpimo, gaunama nauja lentelė, kurioje lyginant su originalia atsakymų lentele (žr. 3.7 lent.) yra pridėti papildomi 9 laukai (visi susiję su klausimu – po vieną *int(11), varchar(1200), varchar(255), tinyint(3), tinyint(3), timestamp, varchar(1), timestamp* bei *varchar(1200)* tipo). Tokiu būdu, prie atsakymo yra saugomas ir klausimas. Klausimas kartojamas prie kiekvieno atitinkamo atsakymo, todėl įrašas dubliuojamas tiek kartų, kiek klausimas turi atsakymų. Toks kartojimas lemia, kad pasikeičia klausimo struktūra duomenų bazėje – klausimo identifikatorius nebėra vienintelis ir unikalus, o yra pasikartojantis. Todėl papildomai keičiasi tokios lentelės, kurios siejosi su *questions* lentele N:1 ryšiu. Šiuo atveju, tokia lentelė yra *exam_test_answers* lentelė. Po klausimų ir atsakymų sujungimo, šioje lentelėje nebereikia saugoti klausimo identifikatoriaus. Todėl *QST_ID* stulpelis yra išmetamas, o vietoje jo pirminį raktą sudaro atsakymo identifikatorius.

Atlikus anksčiau įvardintą realizacijos eigą, iš dviejų lentelių gaunama viena lentelė. Lyginant šią lentelę su originalia atsakymų lentele, pastaroji turi daugiau stulpelių (todėl ji yra platesnė ir didesnė). Iš viso vienai atsakymo eilutei prisidėjo klausimo 9 laukai. Lentelei priskirti abiejų pradinių lentelių indeksai – *qst_id* (iš *answers* lentelės), *test_id, complex, status* (iš *questions* lentelės). Tačiau turint tokią lentelę, nebereikalinga atskira klausimų lentelė, todėl pastarosios lentelės nebelieka. Taip pat pasikeičia lentelės, kurios turėjo N:1 sąryšį su klausimų lentele. Tokioje lentelėje sumažėja stulpelių bei įrašų kiekis.

Rezultato esmė, kad vietoje dviejų siauresnių lentelių, gaunama viena platesnė. Lyginant su metodo-1 lentelė, ši lentelė yra siauresnė. Kaip ir pirmuoju atveju, šio metodo realizacija lemia, jog visi duomenys saugomi vienoje vietoje ir užklausose sumažėja sujungimų kiekis. Tačiau metodas reikalauja pakeitimų ir kitoms lentelėms.

3.3.3. Metodo-3 realizacija

Trečiasis metodas yra skaičiuojamųjų laukų saugojimas lentelėje. Šiuo atveju saugoma klausimo atsakinėjimo statistika – kiek iš viso klausimas buvo atsakytas teisingai, dalinai teisingai ir neteisingai. Tokie skaičiai padeda analizuoti klausimus – pavyzdžiui, didelis neteisingo atsakinėjimo kiekis gali nurodyti dėstytojo padarytą klaidą kuriant klausimą ar atsakymus, didelis teisingo atsakinėjimo kiekis gali nurodyti pernelyg lengvą klausimą.

Šio metodo esmė, kad klausimo atsakinėjimo statistika yra saugoma klausimų lentelėje prie kiekvieno įrašo. Tokiu būdu, *questions* lentelė yra papildoma naujais trimis stulpeliais teisingai atsakytam, dalinai teisingai atsakytam bei neteisingai atsakytam kiekiui saugoti. Tokia statistika gaunama iš *exam_test_answers* lentelės, lyginant *eval* ir *total_eval* vertes. Pirmoji reikšmė nurodo,

kiek už nurodytą klausimą buvo gauta taškų, o kita reikšmė iliustruoja maksimalų galimą įvertinimą už klausimą. *Exam_test_answers* lentelės struktūra bei pavyzdiniai duomenys pateikti 3.13 lent.

3.13 lent. *Exam_test_answers* lentelės pavyzdiniai duomenys

exam_id	stud_id	qst_id	answer	eval	total_eval
Egzaminas1	Vartotojas1	Klausimas6	Atsakymas6_1	12	12
Egzaminas1	Vartotojas1	Klausimas9	Atsakymas9_3	0	12
Egzaminas1	Vartotojas3	Klausimas6	Atsakymas6_2	6	12
Egzaminas2	Vartotojas20	Klausimas11	Atsakymas11_1	10	10
Egzaminas2	Vartotojas25	Klausimas11	Atsakymas11_4	0	10

Realizacijos eiga susideda iš to, kad lentelėje pridedami trys stulpeliai – *answ_right, answ_partial* ir *answ_wrong.* Visi yra *int(11)* tipo. Reikšmės skaičiuojamos pagal *exam_test_answers* duomenis – jeigu *eval* reikšmė lygi *total_eval*, tuomet sumuojasi *answ_right* stulpelis. Jeigu *eval* reikšmė yra lygi nuliu, tuomet sumuojasi *answ_wrong* stulpelio reikšmės. Tarpiniuose atvejuose, kai *eval* reikšmė mažesnė už galimą maksimalią reikšmę, bet didesnė už nulį – pridedamos *answ_partial* reikšmės. Kiekiai skaičiuojami kiekvienam klausimui atskirai. Tokiu būdu gaunama *questionsWithCounts* lentelė (žr. 3.14 lent.)

3.14 lent. *Questions* lentelė papildyta atsakinėjimo statistika

id	test_id	title	picture	radio	complex	access_date	status	edit_date	code	answ_right	answ_partia_	answ_wrong
6	1	Klausimas6	NULL	0	2	2012-10-22 11:01:52	A	2006-07-14 16:53:05	NULL	1	1	0
9	2	Klausimas9	NULL	1	0	2011-05-21 09:00:55	N	2006-07-14 16:53:05	NULL	0	0	1
11	1	Klausimas11	NULL	1	2	2010-10-20 14:35:51	N	2006-07-14 16:53:05	NULL	1	0	1

Rezultato esmė, kad norint gauti klausimo atsakinėjimo statistiką, nebereikia naudoti skaičiavimų, o pakanka reikšmes skaityti iš lentelės. Tai pagreitina duomenų gavimą. Tačiau skaičiuojami laukai turi būti perskaičiuojami kiekvieną kartą, kai modifikuojami *exam_test_answers* lentelės duomenys. Tam gali būti naudojami trigeriai ar procedūros.

3.3.4. Metodo-4 realizacija

Ketvirtajame metode saugomi pasikartojantys laukai lentelėje. Šiuo atveju *exams* lentelėje saugomi *user_id* ir *module_id* laukai. Nurodyti laukai yra dublikatai, kadangi originaliai jie saugomi *tests* lentelėje. *Tests* ir *exams* lentelės susijusios 1:N sąryšiu. Pradinėje schemoje šie laukai su egzamino informacija gaunami sujungus *exams* ir *tests* lenteles. Tokiu atveju sujungimai reikalingi net tuomet, kai informacija iš kitos lentelės net neatvaizduojama – pavyzdžiui, bandant gauti tam tikro dėstytojo ar modulio egzaminus. Tokių laukų saugojimas *exams* lentelėje sumažina sujungimų kiekį užklausoje.

Šio metodo esmė, kad dėstytojo identifikatorius (*user_id*) bei modulio identifikatorius (*module_id*) yra saugomi *exams* lentelėje, nors ši informacija galėtų būti gauta sujungus *exams* ir *tests* lenteles. Pradinė *exams* lentelės struktūra bei pavyzdiniai duomenys pateikti 3.15 lent., *tests* lentelės struktūra bei pavyzdiniai duomenys pateikti 3.16 lent.

3.15 lent. *Exams* lentelės pavyzdiniai duomenys

id	test _id	qst_time	total _time	creation _date	name	show_ answers	select_ questions
Egzaminas1	Testas1	0	20	2017-02-21 11:56:23	Pavadinimas1	0	0
Egzaminas2	Testas1	0	15	2017-03-02 15:35:20	Pavadinimas2	0	0
Egzaminas6	Testas9	0	0	2017-04-27 10:31:03	Pavadinimas6	1	0

3.16 lent. *Tests* lentelės pavyzdiniai duomenys

id	title	description	user_id	status	quiz	module_id
Testas1	Pavadinimas1	Aprašas1	Vartotojas100	A	0	Modulis2
Testas9	Pavadinimas9	Aprašas9	Vartotojas124	A	0	Modulis5
Testas20	Pavadinimas20	Aprašas20	Vartotojas50	A	0	Modulis12

Realizacijos eiga susideda iš to, kad lentelėje pridedami du stulpeliai – *user_id* ir *module_id.* Nauji laukai yra pridedami kaip indeksai. Kadangi abu laukai yra identifikatoriai, todėl jie yra *int(11)* tipo. Reikšmės gaunamos atsižvelgiant į *tests* ir *exams lentelių* tarpusavio sąryšį. Tokiu būdu gaunama *examsRepeated* lentelė (žr. 3.17 lent.).

3.17 lent. *Exams* lentelė papildyta duomenų dublikatais

id	test_id	user_id	qst_time	total_time	creation_date	module_id	name	show_answers	select_questions
Egzaminas1	Testas1	Vartotojas100	0	20	2017-02-21 11:56:23	Modulis2	Pavadinimas1	0	0
Egzaminas2	Testas1	Vartotojas100	0	15	2017-03-02 15:35:20	Modulis2	Pavadinimas2	0	0
Egzaminas6	Testas9	Vartotojas124	0	0	2017-04-27 10:31:03	Modulis5	Pavadinimas6	1	0

Atlikus anksčiau įvardintą realizacijos eigą, egzaminų lentelė praplečiama. Kitos lentelės nekeičiamos. Dėstytojo ir modulio saugojimas egzaminų lentelėje lemia, kad užklausose nebereikalingas sujungimas su testų lentele, kai yra reikalinga informacija, susijusi su šiais dviem laukais.

4. DUOMENŲ BAZĖS OPTIMIZACIJOS METODŲ EKSPERIMENTAS

Šiame skyriuje pateikiamas eksperimentas bei jo eiga. Matavimai atliekami kiekvienam metodui – pasirenkami įvairūs scenarijai, kuriuose naudojamos skirtingos užklausos. Be to, skyriuje pateikiamos eksperimento metrikos – lentelių dydžiai, vykdymo laikai bei užklausos. Šie parametrai matuojami pradinei duomenų bazei ir duomenų bazei po optimizacijos. Gauti rezultatai apibendrinami kitame skyriuje.

4.1. Eksperimentų eiga

Eksperimentas pradedamas nuo to, jog kiekvienam metodui nusprendžiami eksperimentuojami scenarijai. Vienas scenarijus apima užklausas prieš optimizaciją ir po jos. Eksperimentas atliekamas naudojantis *WAMP* aplinka. Lokaliai kompiuteryje į .php failus *PHP* kalba surašomos užklausos, o paleidus naršyklę jos vykdomos. Naršyklės lange išvedami matavimų rezultatai, kurie yra surašomi prie atitinkamo metodo matavimų.

Matavimai atliekami po 100 kartų ir vėliau yra išmatuojamas užklausos vykdymo laiko vidurkis. Matavimai atliekami su trijų tipų duomenų imtimis. Pradinė duomenų bazė traktuojama kaip vidutinio dydžio imtis. Mažesnė duomenų imtis gaunama ištrynus tam tikrus duomenis, o didesnė – dubliuojant duomenis. Duomenų dydžiai:

- dešimties tūkstantųjų eilės (apie 10 000 įrašų konkrečiam metodui aktualiose lentelėse);
- šimto tūkstantųjų eilės (apie 100 000 įrašų konkrečiam metodui aktualiose lentelėse);
- milijono eilės (apie 1 000 000 įrašų konkrečiam metodui aktualiose lentelėse).

4.2. Metodo-1 eksperimentas

Pirmas metodas sujungia *questions* ir *answers* lenteles į vieną (*questionsWithAnswers*). Visų pirma matuojami lentelių dydžiai prieš ir po optimizacijos. Minėtų lentelių palyginimas pateiktas 4.1 lent. Šioje lentelėje atsižvelgiama tik į lentelių dydžius – tiek jų užimamą vietą, tiek elementų kiekį. Elementų kiekis apskaičiuojamas dauginant visų eilučių kiekį iš elementų kiekio vienoje eilutėje (neatsižvelgiama į tai ar egzistuoja reikšmė). Kiti parametrai apžvelgiami vėliau, kai analizuojamos konkrečios užklausos.

4.1 lent. Lentelių dydžių palyginimas

Parametrai \ Lentelės		Questions	Answers	QuestionsWithAnswers
Maža duomenų imtis				
Dydis	Iš viso	4,72 MB	7,44 MB	8,7 MB
	Duomenys	3,67 MB	5,14 MB	7,65 MB
	Indeksai	1,05 MB	2,30 MB	1,05 MB
Elementų kiekis (eilučių kiekis * elementų kiekis eilutėje)		28 644 * 10 = 286 440	112 944 * 4 = 451 776	28 644 * 26 = 744 744
Vidutinė duomenų imtis				
Dydis	Iš viso	17,72 MB	28,21 MB	32,51 MB
	Duomenys	13,74 MB	19,42 MB	28,53 MB
	Indeksai	3,98 MB	8,79 MB	3,98 MB
Elementų kiekis (eilučių kiekis * elementų kiekis eilutėje)		108 636 * 10 = 1086 360	427 320 * 4 = 1709 280	108 636 * 26 = 2824 536
Didelė duomenų imtis				
Dydis	Iš viso	300,00 MB	442,08 MB	510,03 MB
	Duomenys	213,91 MB	304,40 MB	447,32 MB
	Indeksai	86,09 MB	137,68 MB	62,71 MB
Elementų kiekis (eilučių kiekis * elementų kiekis eilutėje)		1704 936 * 10 = 17049 360	6700 740 * 4 = 26802 960	1704 936 * 26 = 44328 336

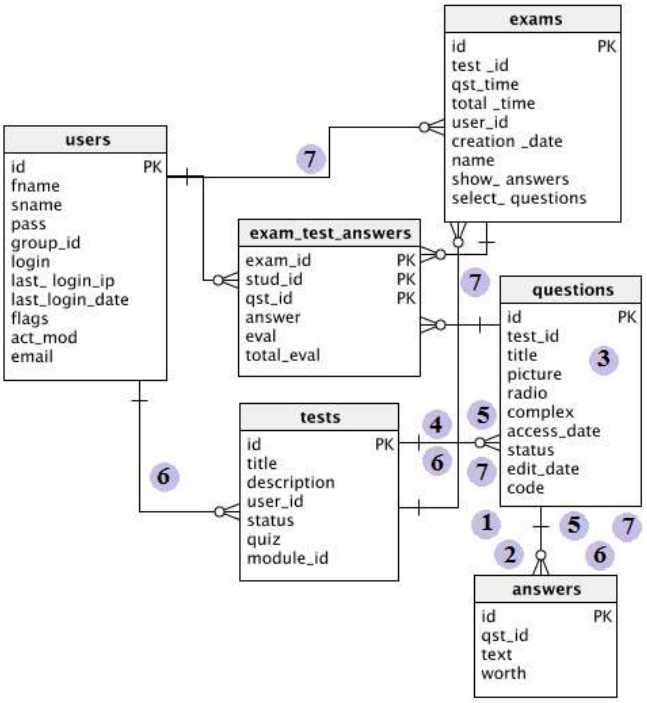
Kaip galima pastebėti iš 4.1 lent., mažos duomenų imties atveju pradinės lentelės (*questions* ir *answers*) bendrai užėmė 12,16 MB. Po optimizacijos lentelė užima 1,4 kartus mažiau vietos. Tikėtina,

kad šis dydis yra mažesnis negu pradinių lentelių suma kadangi iš *answers* lentelės nebenaudojami du stulpeliai – *ID* ir *QST_ID*. *QST_ID* yra indeksas, todėl po sujungimo indeksų dydis sumažėja 3,19 kartų lyginant su abiejų lentelių indeksų dydžių suma. Be to, pašalinus *ID*, 1,15 kartus sumažėja duomenų užimamas dydis.

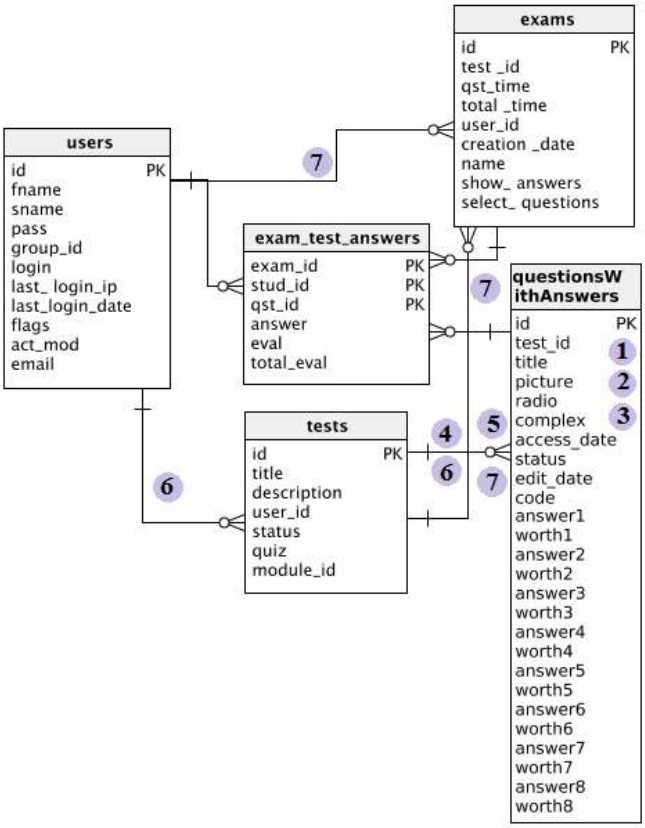
Su vidutine imtimi bendras lentelių dydis sumažėja 1,41 kartus, o su didele – sumažėja 1,45 kartus. Abiem atvejais duomenų dydis sumažėja 1,16 kartų. Indeksų dydis sumažėja 3,21 kartus esant vidutinei duomenų imčiai ir 3,57 esant didelei.

4.1 lent. taip pat galima pastebėti, jog visais duomenų dydžių atvejais įrašų kiekis lyginant su pradinių lentelių suma sumažėja apie 4,9 kartus. Lentelė praplatėja – elementų kiekis eilutėje padidėja 1,86 kartus. Bendrai elementų kiekis beveik nepasikeičia.

Norint apžvelgti kitus parametrus (pavyzdžiui, greitaveiką), nagrinėjamos konkrečios duomenų gavimo iš duomenų bazės operacijos. Tai daroma rašant konkrečias užklausas prieš optimizaciją ir po jos bei matuojant įvairias metrikas. Iš viso yra nagrinėjami 7 duomenų gavimo scenarijai. Scenarijai pradinei duomenų bazei pateikti 4.1 pav., o optimizuotai – 4.2 pav. Skaičiai nurodo scenarijaus numerį. Jeigu skaičius nurodytas prie lentelių sąryšio – užklausoje yra naudojamos abi lentelės. Jeigu numeris nurodytas lentelės viduje – užklausa grąžina tik tos lentelės duomenis.



4.1 pav. Metodo-1 analizuojami duomenų gavimo scenarijai prieš optimizaciją



4.2 pav. Metodo-1 analizuojami duomenų gavimo scenarijai po optimizacijos

4.2.1. Pirmas scenarijus

Pirmojo scenarijaus užklausos grąžina visus klausimus ir atsakymus. Po optimizacijos klausimai ir atsakymai yra saugomi vienoje lentelėje. Tokiu būdu užklausoje traukiami duomenis tik iš vienos lentelės ir užklausa supraprastėja.

Su visomis duomenų imtimis grąžinamų įrašų kiekis sumažėja apie 3,9 kartus. Taip yra todėl, jog pradiniu atveju klausimas yra kartojamas tiek kartų, kiek turi atsakymų. Sujungtoje lentelėje vienas klausimas nėra kartojamas. Nors įrašų kiekis šiuo atveju yra mažesnis, tačiau lyginant su pradine schema, šis rezultatas grąžina daug ilgesnę eilutę – elementų kiekis eilutėje padidėja 1,85 kartus.

Esant mažai duomenų imčiai, duomenų gavimo užklausa vykdoma 7,13 kartų (apie sekundę) greičiau, esant vidutiniai duomenų imčiai – 7,96 kartus (apie 4 sekundes) greičiau, didelės imties atveju – 6,55 kartus (apie 68 sekundes) greičiau. Rezultatai yra pateikti 4.2 lent.

4.2 lent. Pirmojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from questions q, answers a where q.id=a.qst_id	select * from questionswithanswers
Elementų kiekis eilutėje	10 + 4 = 14	26
Įrašų kiekis	112 944	28 644
Vykdymo laikas	1,1546 s	0,1620 s
Įrašų kiekis	427 320	108 636
Vykdymo laikas	4,6625 s	0,5856 s
Eilučių kiekis	6700 740	1704 936
Vykdymo laikas	80,6310 s	12,3027 s

4.2.2. Antras scenarijus

Sekančio scenarijaus užklausos grąžina klausimus ir atsakymus, kurie pateikti specifiniame teste. Kadangi po optimizacijos klausimai ir atsakymai saugomi vienoje lentelėje, užklausoje nebelieka sujungimų. Tokiu būdu užklausoje skenuojama viena, o ne dvi lentelės, todėl užklausa supaprastėja.

Su visomis duomenų imtimis grąžinamų įrašų kiekis sumažėja 5,33–5,42 kartus. Taip yra todėl, jog originalios schemos rezultate tas pats klausimas kartojamas prie tiek įrašų, kiek klausimas turi atsakymų. Tai lemia, kad egzistuoja daugiau negu vienas įrašas, kurio klausimo identifikatoriai sutampa (tačiau skiriasi atsakymų identifikatoriai). Po optimizacijos rezultate klausimai nesikartoja, tačiau vienas įrašas pasidaro platesnis – elementų kiekis eilutėje padidėja 1,85 kartus.

Užklausos vykdymo laikas su maža duomenų imtimi pagreitėja 3,35 kartus, su vidutine duomenų imtimi pagreitėja 4,39 kartus, o su didele – 4,07 kartus. Rezultatai yra pateikti 4.3 lent.

4.3 lent. Antrojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from questions q inner join answers a on a.qst_id = q.id where test_id = 1	select * from questionswithanswers where test_id = 1
Elementų kiekis eilutėje	10 + 4 = 14	26
Įrašų kiekis	288	54
Vykdymo laikas	0,0067 s	0,0020 s
Įrašų kiekis	1 020	189
Vykdymo laikas	0,0180 s	0,0041 s
Įrašų kiekis	15 660	2 889
Vykdymo laikas	0,2226 s	0,0547 s

4.2.3. Trečias scenarijus

Trečiame scenarijuje užklausa grąžina klausimus, kurie pateikti konkrečiame teste. Kadangi *questionsWithAnswers* lentelėje saugomi klausimai ir atsakymai, todėl norint gauti tik klausimus, užklausa turi būti atitinkamai keičiama. Tai lemia, kad gaunami ne visi lentelės duomenys, o tik jų dalis – todėl užklausoje turi būti išvardinti visi su klausimu susiję stulpeliai. Toks stulpelių išvardinimas prideda užklausai sudėtingumo.

Su visomis duomenų imtimis grąžinama tik klausimo informacija – todėl atvaizduojami duomenys yra vienodi. Dėl šios priežasties įrašų bei elementų kiekis prieš ir po optimizacijos lieka nepakitęs. Užklausos vykdymo laikas po optimizacijos mažos imties atveju sutrumpėja 1,28 kartus, su vidutine imtimi pagreitėja 1,03 kartus, o didelės imties atveju – pagreitėja 1,08 kartus. Rezultatai yra pateikti 4.4 lent.

4.4 lent. Trečiojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from questions where test_id = 1548	select id, test_id, title, picture, radio, complex, access_date, status, edit_date, code from questionswithanswers where test_id = 1548
Elementų kiekis eilutėje	10	10
Įrašų kiekis	426	426
Vykdymo laikas	0,0073 s	0,0057 s
Įrašų kiekis	1 446	1 446
Vykdymo laikas	0,0175 s	0,0170 s
Įrašų kiekis	21 846	21 846
Vykdymo laikas	0,3203 s	0,2947 s

Kitokie rezultatai gaunami, jeigu nekeičiama užklausos logika ir traukiami visi duomenys iš lentelių. Nors grąžinamų įrašų kiekis lieka nepakitęs, tačiau elementų kiekis eilutėje po optimizacijos padidėja 2,6 karto. Vadinasi, nurodyta užklausa grąžina tiek klausimo, tiek atsakymų informaciją (pradiniu atveju tik klausimo). Kadangi po metodo pritaikymo skenuojama didesnė lentelė ir grąžinama daugiau duomenų, todėl vykdymo laikas ilgėja. Mažos imties atveju užklausa vykdoma 1,68 kartus ilgiau, vidutinės imties atveju – 1,15 kartų lėčiau, o su didele imtimi – 1,36 kartus ilgiau. Rezultatai yra pateikti 4.5 lent.

4.5 lent. Trečiojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from questions where test_id = 1548	select * from questionswithanswers where test_id = 1548
Elementų kiekis eilutėje	10	26
Įrašų kiekis	426	426
Vykdymo laikas	0,0063 s	0,0106 s
Įrašų kiekis	1 446	1 446
Vykdymo laikas	0,0198 s	0,0229 s
Įrašų kiekis	21 846	21 846
Vykdymo laikas	0,2040 s	0,2780 s

4.2.4. Ketvirtas scenarijus

Šiame scenarijuje užklausa grąžina informaciją apie klausimus ir testą. Tačiau ši informacija yra specifinio testo ir vartotojo. Iš *questionsWithAnswers* lentelės norint gauti tik klausimo duomenis užklausoje turi būti nurodyti konkretūs su klausimu susiję stulpeliai. Norint gauti testo informaciją, užklausoje taip pat turi būti išvardinti testo stulpeliai. Pradiniu atveju konkrečių stulpelių minėti nereikia, todėl galima teigti, jog po optimizacijos užklausa pasidaro sudėtingesnė.

Įrašų bei elementų kiekis visų duomenų imčių atvejais yra vienodas. Su mažos imties duomenimis užklausa vykdoma 1,19 kartų greičiau, su vidutinio dydžio duomenimis – apie 1,31 kartus greičiau, o didelių duomenų atveju – 1,02 kartus. Rezultatai yra pateikti 4.6 lent.

4.6 lent. Ketvirtojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from questions q inner join tests t on t.id = q.test_id and t.user_id = 2 where test_id = 878	select qa.id, qa.test_id, qa.title, qa.picture, qa.radio, qa.complex, qa.access_date, qa.status, qa.edit_date, qa.code, t.id, t.title, t.description, t.user_id, t.status, t.quiz, t.module_id from questionswithanswers qa inner join tests t on t.id = qa.test_id and t.user_id = 2 where test_id = 878
Elementų kiekis eilutėje	10 + 7 = 17	10 + 7 = 17
Įrašų kiekis	131	131
Vykdymo laikas	0,0044 s	0,0037 s
Įrašų kiekis	395	395
Vykdymo laikas	0,0071 s	0,0054 s
Įrašų kiekis	5 675	5 675
Vykdymo laikas	0,1010 s	0,0992 s

Jeigu šiuo atveju užklausos logika nekeičiama (paliekamas visų duomenų išrinkimas iš lentelių), tuomet grąžinamų įrašų kiekis nepakinta, o eilutė tampa platesnė. Elementų kiekis eilutėje padidėja 1,94 kartus. Taip yra todėl, jog grąžinami ir atsakymai (kurių nebuvo pradiniu atveju).

Mažos duomenų imties atveju užklausa vykdoma 1,31 kartus lėčiau. Esant vidutiniam duomenų kiekiui užklausa vykdoma 1,15 kartus greičiau. Didelių duomenų atveju užklausa vykdoma 1,12 kartus lėčiau. Rezultatai yra pateikti 4.7 lent.

4.7 lent. Ketvirtojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from questions q inner join tests t on t.id = q.test_id and t.user_id = 2 where test_id = 878	select * from questionswithanswers qa inner join tests t on t.id = qa.test_id and t.user_id = 2 where test_id = 878
Elementų kiekis eilutėje	10 + 7 = 17	26 + 7 = 33
Įrašų kiekis	131	131
Vykdymo laikas	0,0029 s	0,0038 s
Įrašų kiekis	395	395
Vykdymo laikas	0,0067 s	0,0058 s
Įrašų kiekis	5 675	5 675
Vykdymo laikas	0,0939 s	0,1055 s

4.2.5. Penktas scenarijus

Penktasis scenarijus panašus į ketvirtąjį – tik į rezultatus yra pridedami ir atsakymai. Tai lemia, kad yra grąžinami klausimai, atsakymai bei testai. Tokia informacija yra grąžinama pagal konkretų vartotoją ir testą. Kadangi po optimizacijos užklausoje nebereikalingas vienas sujungimas ir skenuojamos viena lentele mažiau, todėl užklausa supaprastėja.

Originalios schemos atveju rezultate tas pats klausimas kartojamas ne vieną kartą. Po lentelių sujungimo, klausimas rezultate nebesikartoja ir įrašų kiekis visais duomenų imčių atvejais sumažėja apie 4,8 kartus. Elementų kiekis eilutėje padidėja 1,57 kartus.

Užklausos vykdymas su maža duomenų imtimi pagreitėja 4,76 kartus. Esant vidutinei duomenų imčiai užklausa vykdoma 5,56 kartus greičiau. Su dideliais duomenų kiekiais užklausa vykdoma 4,85 kartus greičiau. Rezultatai yra pateikti 4.8 lent.

4.8 lent. Penktojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from questions q inner join answers a on a.qst_id = q.id inner join tests t on t.id = q.test_id and t.user_id = 2 where test_id = 878	select * from questionswithanswers q inner join tests t on t.id = q.test_id and t.user_id = 2 where test_id = 878
Elementų kiekis eilutėje	10 + 4 + 7 = 21	26 + 7 = 33
Įrašų kiekis	622	131
Vykdymo laikas	0,0119 s	0,0025 s
Įrašų kiekis	1 906	395
Vykdymo laikas	0,0300 s	0,0054 s
Įrašų kiekis	27 586	5 675
Vykdymo laikas	0,3383 s	0,0698 s

4.2.6. Šeštas scenarijus

Šis scenarijus grąžina informaciją apie klausimus, atsakymus, testus bei vartotojus. Informacija grąžinama tik konkretaus testo. Scenarijus panašus į penktąjį, tik čia papildomai pridedama ir vartotojo informacija. Kaip ir anksčiau – užklausa dėl sumažėjusio sujungimų skaičiaus supaprastėja. Dėl to, kad klausimo ir jo atsakymų informacija saugoma vienoje eilutėje, grąžinamų įrašų kiekis pamažėja apie 4,8 kartus, o eilutė pailgėja 1,38 kartus.

Esant mažai duomenų imčiai užklausos vykdymas pagreitėja 4,27 kartus. Vidutinio duomenų dydžio atveju vykdymo laikas sutrumpėja 5,09 kartus, o didelio duomenų kiekio atveju – 3,8 kartus. Rezultatai yra pateikti 4.9 lent.

4.9 lent. Šeštojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from questions q inner join answers a on a.qst_id = q.id inner join tests t on t.id = q.test_id inner join users u on u.id = t.user_id where test_id = 878	select * from questionswithanswers q inner join tests t on t.id = q.test_id inner join users u on u.id = t.user_id where test_id = 878
Elementų kiekis eilutėje	10 + 4 + 7 + 11 = 32	26 + 7 + 11 = 44
Įrašų kiekis	622	131
Vykdymo laikas	0,0192 s	0,0045 s
Įrašų kiekis	1 906	395
Vykdymo laikas	0,0321 s	0,0063 s
Įrašų kiekis	27 586	5 675
Vykdymo laikas	0,3671 s	0,0965 s

4.2.7. Septintas scenarijus

Paskutinis pirmojo metodo scenarijus grąžina konkretaus egzamino informaciją. Rezultate pateikiami duomenys apie klausimus, atsakymus, testus, vartotojus bei egzaminą. Pašalinus sujungimą užklausoje, sumažėja skenuojamų lentelių kiekis bei užklausa supaprastėja. Sujungus klausimus ir atsakymus, grąžinamų įrašų kiekis pamažėja 4 kartus, tačiau eilutė pasidaro platesnė ir elementų kiekis eilutėje pailgėja 1,29 kartus.

Su maža duomenų imtimi užklausos vykdymo laikas sutrumpėja 4,37 kartus. Vidutinio duomenų dydžio atveju užklausa vykdoma 5,08 kartus greičiau. Esant dideliam duomenų kiekiui užklausa vykdoma 3,8 kartus greičiau. Rezultatai yra pateikti 4.10 lent.

4.10 lent. Septintojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from questions q inner join answers a o n a.qst_id = q.id inner join tests t on t.id= q.test _id inner join exams e on e.test_id = t.id inner j oin users u on u.id=e.user_id where e.id = 315	select * from questionswithanswers q inner join tests t on t.id = q.test_id inner join exams e on e.test_id = t.id inner join users u on u.id = e.user_id where e.id = 315
Elementų kiekis eilutėje	10 + 4 + 7 + 10 + 11 = 42	26 + 7 + 10 + 11 = 54
Įrašų kiekis	1 108	277
Vykdymo laikas	0,0245 s	0,0056 s
Įrašų kiekis	3 736	934
Vykdymo laikas	0,0737	0,0145
Įrašų kiekis	56 296	14 074
Vykdymo laikas	0,9438 s	0,2486 s

4.3. Metodo-2 eksperimentas

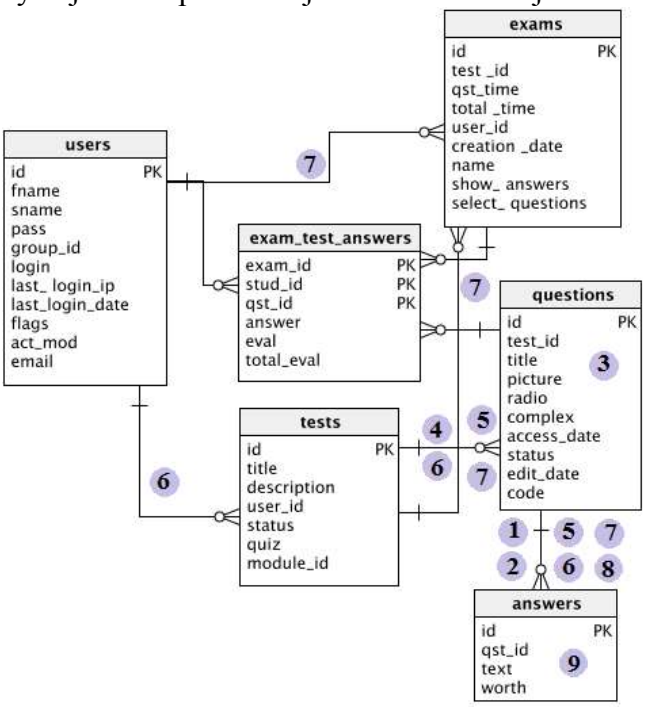
Kaip ir pirmasis metodas, metodas-2 sujungia *questions* ir *answers* lenteles į vieną (*answersWithQuestions*). Kaip ir pirmojo eksperimento atveju, poskyrio pradžioje pateikiamas lentelių dydžių palyginimas, o vėliau analizuojamos konkrečios užklausos. Minėtų lentelių užimamos vietos bei elementų kiekio palyginimas pateiktas 4.11 lent. Kaip galima pastebėti iš šios lentelės, sujungtos lentelės (*answersWithQuestions*) dydis su visomis duomenų imtimis yra apie 1,9 kartus didesnis negu pradinių lentelių dydžių suma. Tai galima būtų paaiškinti tuo, jog klausimo informacija yra kartojama – tas pats klausimas yra saugomas prie kiekvieno atitinkamo atsakymo. Po sujungimo duomenų dydis padidėja apie 2,1 kartą, o indeksų dydis padidėja 1,5 kartus.

4.11 lent. taip pat galima pastebėti, jog visais duomenų dydžių atvejais bendras elementų kiekis padidėja 1,99 kartus. Kadangi lentelė sujungiama atsakymo atžvilgiu, todėl *answersWithQuestions* lentelėje saugoma tiek eilučių, kiek jų yra *answers* lentelėje. Dėl panaikinto sąryšio nagrinėjama lentelė susiaurėja vienu elementu (lyginant su pradinių lentelių elementų suma) – elementų kiekis eilutėje sumažėja 1,08 kartus. Įrašų kiekis sumažėja 1,25 kartus.

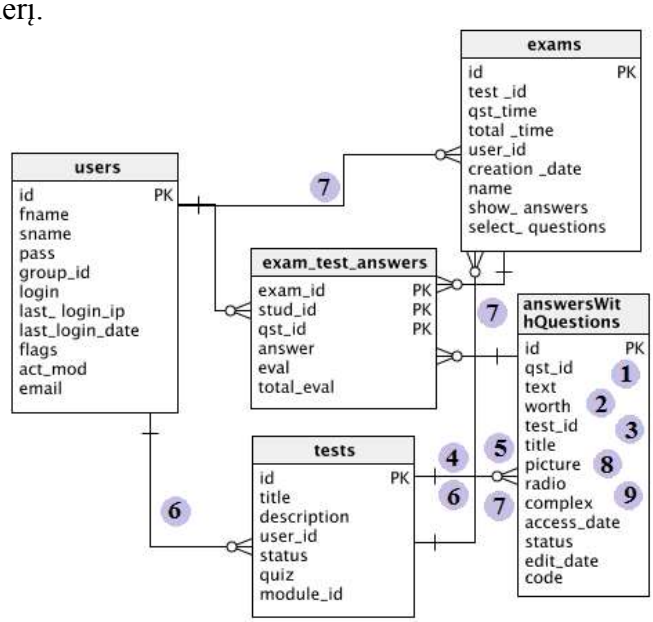
4.11 lent. Lentelių dydžių palyginimas

Parametrai \ Lentelės		Questions	Answers	AnswersWithQuestions
Maža duomenų imtis				
Dydis	Iš viso	4,72 MB	7,44 MB	24,01 MB
	Duomenys	3,67 MB	5,14 MB	18,66 MB
	Indeksai	1,05 MB	2,30 MB	5,35 MB
Elementų kiekis (eilučių kiekis * elementų kiekis eilutėje)		28 644 * 10 = 286 440	112 944 * 4 = 451 776	112 944 * 13 = 1468 272
Vidutinė duomenų imtis				
Dydis	Iš viso	17,72 MB	28,21 MB	89,34 MB
	Duomenys	13,74 MB	19,42 MB	69,21 MB
	Indeksai	3,98 MB	8,79 MB	20,13 MB
Elementų kiekis (eilučių kiekis * elementų kiekis eilutėje)		108 636 * 10 = 1086 360	427 320 * 4 = 1709 280	427 320 * 13 = 5555 160
Didelė duomenų imtis				
Dydis	Iš viso	300,00 MB	442,08 MB	1395,86 MB
	Duomenys	213,91 MB	304,40 MB	1080,97 MB
	Indeksai	86,09 MB	137,68 MB	314,89 MB
Elementų kiekis (eilučių kiekis * elementų kiekis eilutėje)		1704 936 * 10 = 17049 360	6700 740 * 4 = 26802 960	6700 740 * 13 = 87109 620

Norint apžvelgti kitus parametrus (pavyzdžiui, vykdymo laikus), nagrinėjamos konkrečios duomenų gavimo iš duomenų bazės operacijos. Kaip ir pirmojo metodo atveju, tai daroma rašant konkrečias užklausas prieš optimizaciją ir po jos bei matuojant įvairias metrikas. Iš viso yra nagrinėjami 9 duomenų gavimo scenarijai – pirmieji septyni scenarijai analogiški pirmojo metodo scenarijams. Scenarijai pradinei duomenų bazei pateikti 4.3 pav., o optimizuotai – 4.4 pav. Skaičių žymėjimas nepakinta – jie nurodo scenarijaus numerį.



4.3 pav. Metodo-2 nagrinėjami duomenų gavimo scenarijai prieš optimizaciją



4.4 pav. Metodo-2 nagrinėjami duomenų gavimo scenarijai po optimizacijos

4.3.1. Pirmas scenarijus

Pirmojo scenarijaus užklausos grąžina visus klausimus ir atsakymus. Prieš optimizaciją užklausoje gaunami visi duomenis iš *questions* ir *answers* lentelių. Po optimizacijos atsakymai ir klausimai yra saugomi vienoje lentelėje, todėl užklausoje traukiami duomenis tik iš vienos lentelės. Tai lemia, kad užklausa supraprastėja.

Po lentelių sujungimo nagrinėjama užklausa grąžina tiek pat įrašų kiek ir prieš optimizaciją. Kadangi nebenaudojamas išorinis raktas, todėl elementų kiekis yra mažesnis – lentelės plotis yra vienu elementu (1,07 kartus) siauresnis.

Esant mažai duomenų imčiai užklausos vykdymo laikas pagreitėja 2,81 kartus. Vidutinės duomenų imties atveju duomenų gavimo užklausa vykdoma 2,65 kartus trumpiau. Didelių duomenų atveju užklausos vykdymo laikas pagerėja 3,09 kartus. Rezultatai yra pateikti 4.12 lent.

4.12 lent. Pirmojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from answers a, questions q where q.id=a.qst_id	select * from answerswithquestions
Elementų kiekis eilutėje	4 + 10 = 14	13
Įrašų kiekis	112 944	112 944
Vykdymo laikas	1,3667 s	0,4858 s
Įrašų kiekis	427 320	427 320
Vykdymo laikas	4,1773 s	1,5773 s
Įrašų kiekis	6700 740	6700 740
Vykdymo laikas	85,3959 s	27,6206 s

4.3.2. Antras scenarijus

Sekančio scenarijaus užklausos grąžina klausimus ir atsakymus, kurie pateikti specifiniame teste. Šiuo atveju užklausa supaprastėja, kadangi po optimizacijos klausimai ir atsakymai saugomi vienoje lentelėje, todėl užklausoje nebelieka sujungimų ir skenuojama viena lentele mažiau. Grąžinamų įrašų kiekis po optimizacijos lieka nepakitęs. Tačiau sujungus lenteles, eilutė susiaurėja ir grąžinama vienu elementu (1,07 kartus) mažiau negu pradiniu atveju.

Po optimizacijos mažos duomenų imties atveju užklausa vykdoma 1,52 kartus greičiau. Esant vidutiniam duomenų dydžiui užklausa vykdoma 1,37 kartus trumpiau. Didelių duomenų atveju užklausos vykdymas pagreitėja 1,41 kartus. Rezultatai yra pateikti 4.13 lent.

4.13 lent. Antrojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from answers a inner join questions q on a.qst_id=q.id where test_id = 1	select * from answerswithquestions where test_id = 1
Elementų kiekis eilutėje	4 + 10 = 14	13
Įrašų kiekis	288	288
Vykdymo laikas	0,0064 s	0,0042 s
Įrašų kiekis	1 020	1 020
Vykdymo laikas	0,0153 s	0,0112 s
Įrašų kiekis	15 660	15 660
Vykdymo laikas	0,2238 s	0,1585 s

4.3.3. Trečias scenarijus

Trečiame scenarijuje užklausa grąžina klausimus, kurie pateikti konkrečiame teste. Šiuo atveju užklausos logika nekeičiama ir traukiami visi duomenys iš lentelių. Kadangi egzistuoja daugiau negu vienas įrašas, kurio klausimo identifikatoriai sutampa (tačiau skiriasi atsakymų identifikatoriai), įrašų kiekis po optimizacijos padidėja 4 kartus. Elementų kiekis eilutėje padidėja 1,3 kartus, kadangi nurodyta užklausa grąžina klausimo ir atsakymo informaciją (pradiniu atveju tik klausimo).

Po metodo pritaikymo skenuojama didesnė lentelė ir grąžinama daugiau duomenų. Mažos duomenų imties atveju užklausos vykdymo laikas pailgėja 2,21 kartus. Esant vidutinei duomenų imčiai užklausos vykdymo laikas yra 3,74 kartus ilgesnis. Didelių duomenų atveju – 3,62 kartus lėtesnis. Rezultatai yra pateikti 4.14 lent.

4.14 lent. Trečiojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from questions where test_id = 1548	select * from answerswithquestions where test_id = 1548
Elementų kiekis eilutėje	10	13
Įrašų kiekis	426	1704
Vykdymo laikas	0,0072 s	0,0159 s
Įrašų kiekis	1 446	5 784
Vykdymo laikas	0,0144 s	0,0539 s
Įrašų kiekis	21 846	87 384
Vykdymo laikas	0,2574 s	0,9319 s

Kadangi *answersWithQuestions* lentelėje saugomi atsakymai ir klausimai, todėl norint gauti tik klausimus, užklausa turi būti atitinkamai modifikuojama – turi būti išvardinti visi su klausimu susiję stulpeliai. Toks stulpelių išvardinimas prideda užklausai sudėtingumo.

Po lentelių sujungimo tas pats klausimas yra kartojamas. Tai lemia, kad grąžinamų įrašų kiekis padidėja 4 kartus. Norint grąžinti tik unikalius ir nesikartojančius klausimus užklausoje galima naudoti įvairias sąlygas (pavyzdžiui, *distinct* sąlygą). Tačiau tokie veiksmai sulėtina užklausos vykdymo laiką.

Esant mažai duomenų imčiai užklausos vykdymo laikas po optimizacijos pailgėja 2,65 kartus. Vidutinės duomenų imties atveju vykdymo laikas pailgėja 2,97 kartus. Didelių duomenų atveju laikas sulėtėja 3,44 kartus. Rezultatai yra pateikti 4.15 lent.

4.15 lent. Trečiojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from questions where test_id = 1548	select qst_id, test_id, title, picture, radio, complex, access_date, status, edit_date, code from answerswithquestions where test_id = 1548
Elementų kiekis eilutėje	10	10
Įrašų kiekis	426	1 704
Vykdymo laikas	0,0060 s	0,0159 s
Įrašų kiekis	1 446	5 784
Vykdymo laikas	0,0153 s	0,0454 s
Įrašų kiekis	21 846	87 384
Vykdymo laikas	0,2491 s	0,8574 s

4.3.4. Ketvirtas scenarijus

Šiame scenarijuje užklausa grąžina informaciją apie klausimus ir testą. Tačiau ši informacija yra specifinio testo ir vartotojo. Jeigu užklausos logika nekeičiama (paliekamas visų duomenų išrinkimas iš lentelių), tuomet po optimizacijos grąžinamų įrašų kiekis padidėja apie 4,8 kartus. Be to, eilutė yra platesnė, kadangi grąžinama ir atsakymo informacija. Elementų kiekis eilutėje padidėja 1,18 kartų.

Po sujungimo mažos imties duomenų atveju užklausos vykdymo laikas sulėtėja 2,1 kartus. Esant vidutiniai duomenų imčiai laikas pailgėja 2,68 kartus. Didelių duomenų atveju užklausa vykdoma 3,29 kartus ilgiau. Rezultatai yra pateikti 4.16 lent.

4.16 lent. Ketvirtojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from questions q inner join tests t on t.id = q.test_id and t.user_id = 2 where test_id = 878	select * from answerswithquestions aq inner join tests t on t.id = aq.test_id and t.user_id = 2 where test_id = 878
Elementų kiekis eilutėje	10 + 7 = 17	13 + 7 = 20
Įrašų kiekis	131	622
Vykdymo laikas	0,0038 s	0,0080 s
Įrašų kiekis	395	1 906
Vykdymo laikas	0,0072 s	0,0193 s
Įrašų kiekis	5 675	27 586
Vykdymo laikas	0,1181 s	0,3885 s

Iš *answersWithQuestions* lentelės norint gauti tik klausimo duomenis užklausoje turi būti nurodyti konkretūs su klausimu susiję stulpeliai. Norint gauti testo informaciją, užklausoje taip pat turi būti išvardinti testo stulpeliai. Pradiniu atveju konkrečių stulpelių minėti nereikia, todėl galima teigti, jog po optimizacijos užklausa pasidaro sudėtingesnė.

Grąžinamų įrašų kiekis su visomis duomenų imtimis padidėja apie 4,7–4,8 kartus, o elementų kiekis išlieka toks pat. Gali būti naudojamos įvairios sąlygos užklausoje, norint sumažinti grąžinamų eilučių iki tokio, koks buvo pradiniu atveju. Tačiau tokios sąlygos naudojimas pailgina vykdymo laiką.

Esant mažai duomenų imčiai užklausos vykdymo laikas sulėtėja 2,48 kartus. Vidutinės duomenų imties atveju užklausa vykdoma 3,15 kartus lėčiau. Didelių duomenų atveju po sujungimo užklausa vykdoma 2,95 kartus ilgiau. Rezultatai yra pateikti 4.17 lent.

4.17 lent. Ketvirtojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from questions q inner join tests t on t.id = q.test_id and t.user_id = 2 where test_id = 878	select aq.qst_id, aq.test_id, aq.title, aq.picture, aq.radio, aq.complex, aq.access_date, aq.status, aq.edit_date, aq.code, t.id, t.title, t.description, t.user_id, t.status, t.quiz, t.module_id from answerswithquestions aq inner join tests t on t.id = aq.test_id and t.user_id = 2 where test_id = 87
Elementų kiekis eilutėje	10 + 7 = 17	10 + 7 = 17
Įrašų kiekis	131	622
Vykdymo laikas	0,0031 s	0,0077 s
Įrašų kiekis	395	1 906
Vykdymo laikas	0,0071 s	0,0224 s
Įrašų kiekis	5 675	27 586
Vykdymo laikas	0,1123 s	0,3316 s

4.3.5. Penktas scenarijus

Penktasis scenarijus panašus į ketvirtąjį – tik į rezultatus yra pridedami ir atsakymai. Tokia informacija yra grąžinama pagal konkretų vartotoją ir testą. Po optimizacijos užklausoje nebereikalingas vienas sujungimas, todėl užklausa supaprastėja.

Po sujungimo įrašų kiekis lieka nepakitęs. Eilutė susiaurėja vienu elementu (1,05 kartus). Esant mažai duomenų imčiai užklausos vykdymo laikas pagreitėja 1,27 kartus. Su vidutine duomenų imtimi užklausos vykdymo laikas pagerėja 1,15 kartų. Didelių duomenų atveju – pagreitėja 1,39 kartus. Rezultatai yra pateikti 4.18 lent.

4.18 lent. Penktojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from answers a inner join questions q on a.qst_id = q.id inner join tests t on t.id =q.test_id and t.user_id = 2 where test_id = 878	select * from answerswithquestions a inner join tests t on t.id = a.test_id and t.user_id = 2 where test_id = 878
Elementų kiekis eilutėje	4 + 10 + 7 = 21	13 + 7 = 20
Įrašų kiekis	622	622
Vykdymo laikas	0,0113 s	0,0089 s
Įrašų kiekis	1 906	1 906
Vykdymo laikas	0,0241 s	0,0208 s
Įrašų kiekis	27 586	27 586
Vykdymo laikas	0,5532 s	0,3956 s

4.3.6. Šeštas scenarijus

Šis scenarijus grąžina informaciją apie klausimus, atsakymus, testus bei vartotojus. Informacija grąžinama tik konkretaus testo. Scenarijus panašus į penktąjį, tik čia papildomai pridedama ir vartotojo informacija. Kaip ir anksčiau – užklausa dėl sumažėjusio sujungimų skaičiaus supaprastėja. Grąžinamų įrašų kiekis nepakinta, o elementų kiekis eilutėje 1,03 kartus sumažėja.

Užklausos vykdymo laikas esant mažai duomenų imčiai pagreitėja 1,33 kartus. Vidutinės duomenų imties atveju laikas sutrumpėja 1,11 kartų. Didelių duomenų atveju užklausa vykdoma 1,25 kartus greičiau. Rezultatai yra pateikti 4.19 lent.

4.19 lent. Šeštojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from answers a inner join questions q on a.qst_id = q.id inner join tests t on t.id = q.test_id inner join users u on u.id = t.user_id where test_id = 878	select * from answerswithquestions a inner join tests t on t.id = a.test_id inner join users u on u.id = t.user_id where test_id = 878
Elementų kiekis eilutėje	4 + 10 + 7 + 11 = 32	13 + 7 + 11 = 31
Įrašų kiekis	622	622
Vykdymo laikas	0,0129 s	0,0097 s
Įrašų kiekis	1 906	1 906
Vykdymo laikas	0,0245 s	0,0221 s
Įrašų kiekis	27 586	27 586
Vykdymo laikas	0,6182 s	0,4945 s

4.3.7. Septintas scenarijus

Septintas antrojo metodo scenarijus grąžina konkretaus egzamino informaciją. Pašalinus sujungimą užklausoje, užklausa supaprastėja. Nors elementų kiekis eilutėje 1,02 kartus (vienu elementu) sumažėja, tačiau eilučių kiekis nepakinta.

Mažos duomenų imties atveju užklausos vykdymo laikas pagreitėja 1,55 kartus. Esant vidutinei duomenų imčiai užklausos vykdymo laikas sutrumpėja 1,23 kartus. Didelių duomenų atveju užklausa vykdoma 1,46 kartus greičiau. Rezultatai pateikti 4.20 lent.

4.20 lent. Septintojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from answers a inner join questions q on a.qst_id=q.id inner join tests t on t.id= q.test_id inner join exams e on e.test_id=t.id inner join users u on u.id=e.user_id where e.id = 315	select * from answerswithquestions a inner join tests t on t.id=a.test_id inner join exams e on e.test_id=t.id inner join users u on u.id=e.user_id where e.id=315
Elementų kiekis eilutėje	4 + 10 + 7 + 10 + 11 = 42	13 + 7 + 10 + 11 = 41
Įrašų kiekis	1 108	1 108
Vykdymo laikas	0,0243 s	0,0157 s
Įrašų kiekis	3 736	3 736
Vykdymo laikas	0,0604 s	0,0491 s
Įrašų kiekis	56 296	56 296
Vykdymo laikas	1,2756 s	0,8724 s

4.3.8. Aštuntas scenarijus

Aštuntasis scenarijus grąžina teisingus konkretaus klausimo atsakymus bei patį klausimą. Po optimizacijos užklausoje sumažėja sujungimų kiekis, todėl užklausa supaprastėja. Grąžinamų įrašų kiekis nepakinta, o elementų kiekis eilutėje po sujungimo sumažėja vienu elementu (1,08 kartus).

Užklausos vykdymo laikas mažos duomenų imties atveju pagreitėja 2 kartus. Esant vidutinei duomenų imčiai vykdoma 1,4 kartus greičiau. Didelių duomenų atveju užklausa vykdoma 1,22 kartus trumpiau. Rezultatai pateikti 4.21 lent.

4.21 lent. Aštuntojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from answers a inner join questions q on a.qst_id = q.id where a.worth > 0 and a.qst_id = 6	select * from answerswithquestions where worth > 0 and qst_id = 6
Elementų kiekis eilutėje	14	13
Eilučių kiekis	3	3
Vykdymo laikas	0,0006 s	0,0003 s
Eilučių kiekis	3	3
Vykdymo laikas	0,0007 s	0,0005 s
Įrašų kiekis	3	3
Vykdymo laikas	0,00011 s	0,0009 s

4.3.9. Devintas scenarijus

Paskutinis antrojo metodo scenarijus pateikia konkretaus klausimo visus atsakymus. Šiuo atveju užklausos logika nekeičiama ir traukiami visi duomenys iš lentelių. Kadangi *answersWithQuestions* lentelėje saugoma informacija apie atsakymus ir klausimus, todėl elementų kiekis eilutėje padidėja 3,25 kartus. Tačiau grąžinamų įrašų kiekis lieka nepakitęs.

Esant mažai duomenų imčiai užklausos vykdymo laikas suletėja 1,5 kartus. Užklausa vykdoma 1,2 kartus lėčiau esant vidutinei duomenų imčiai. Didelių duomenų atveju užklausos vykdymo laikas nepakinta. Rezultatai pateikti 4.22 lent.

4.22 lent. Devintojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from answers where qst_id=6	select * from answerswithquestions where qst_id=6
Elementų kiekis eilutėje	4	13
Įrašų kiekis	6	6
Vykdymo laikas	0,0004 s	0,0006 s
Įrašų kiekis	6	6
Vykdymo laikas	0,0005 s	0,0006 s
Įrašų kiekis	6	6
Vykdymo laikas	0,0007 s	0,0007 s

Iš *answersWithQuestions* lentelės norint gauti tik atsakymo informaciją, užklausoje turi būti nurodyti konkretūs su atsakymu susiję stulpeliai. Pradiniu atveju konkrečių stulpelių minėti nereikia, todėl galima teigti, jog po optimizacijos užklausa pasidaro sudėtingesnė. Grąžinamų įrašų kiekis bei elementų kiekis eilutėje nepakinta.

Užklausos vykdymo laikas po optimizacijos esant mažai duomenų imčiai nepakinta. Vidutinės duomenų imties atveju užklausa vykdoma 1,2 kartus greičiau. Didelių duomenų atveju – 1,5 kartus trumpiau. Rezultatai pateikti 4.23 lent.

4.23 lent. Devintojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
Sql užklausa	select * from answers where qst_id=6	select id, qst_id, text, worth from answerswithquestions where qst_id=6
Elementų kiekis eilutėje	4	4
Įrašų kiekis	6	6
Vykdymo laikas	0,0005 s	0,0005 s
Eilučių kiekis	6	6
Vykdymo laikas	0,0006 s	0,0005 s
Įrašų kiekis	6	6
Vykdymo laikas	0,0009 s	0,0006 s

4.4. Metodo-3 eksperimentas

Trečiasis metodas lentelėje saugo klausimų atsakinėjimo statistiką (*questionsWithCounts*). Klausimų lentelės palyginimas prieš ir po optimizacijos (skaičiuojamųjų laukų pridėjimo) pateiktas 4.24 lent. Pridėjus klausimų atsakinėjimo statistiką, bendras lentelės dydis padidėja. Esant mažai duomenų imčiai lentelės dydis padidėja 1,73 kartus, su vidutinio dydžio imtimi – 1,37 kartus, o esant dideliems duomenims padidėja 1,12 kartus.

Duomenų dydis padidėja 1,57 kartus esant mažai duomenų imčiai. Kai duomenų imtis yra vidutinė, tuomet duomenų dydis padidėja 1,27 kartus. Su didele duomenų imtimi duomenų dydis padidėja 1,2 kartus.

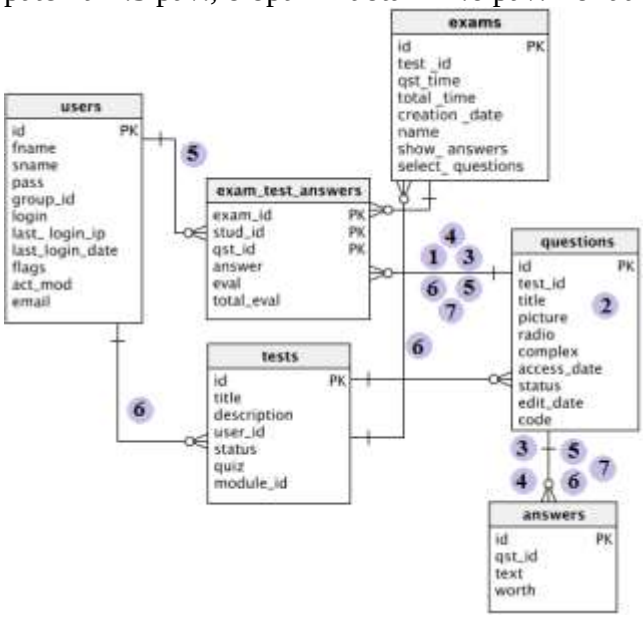
Su maža duomenų imtimi indeksų kiekis padidėja 2,3 kartus. Esant vidutinei duomenų imčiai indeksų dydis padidėja 1,27 kartus. Su dideliais duomenimis sumažėja 1,09 kartus.

Naujoji lentelė papildoma trimis laukais ir praplatėja, todėl ir elementų kiekis padidėja. Su visomis duomenų imtimis bendras elementų kieki padidėja 1,3 kartus.

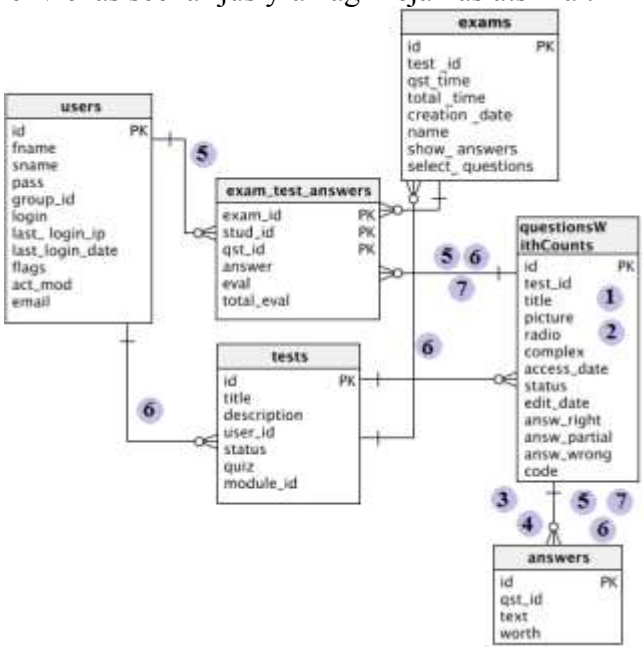
4.24 lent. Lentelių dydžių palyginimas

Parametrai \ Lentelės		Questions	QuestionsWithCounts
Maža duomenų imtis			
Dydis	Iš viso	4,72 MB	8,18 MB
	Duomenys	3,67 MB	5,77 MB
	Indeksai	1,05 MB	2,41 MB
Elementų kiekis (eilučių kiekis * elementų kiekis eilutėje)		28 644 * 10 = 286 440	28 644 * 13 = 372 372
Vidutinė duomenų imtis			
Dydis	Iš viso	17,72 MB	24,22 MB
	Duomenys	13,63 MB	17,36 MB
	Indeksai	4,09 MB	6,86 MB
Elementų kiekis (eilučių kiekis * elementų kiekis eilutėje)		108 636 * 10 = 1086 360	108 636 * 13 = 1412 268
Didelė duomenų imtis			
Dydis	Iš viso	299,79 MB	334,92 MB
	Duomenys	214,33 MB	256,69 MB
	Indeksai	85,46 MB	78,21 MB
Elementų kiekis (eilučių kiekis * elementų kiekis eilutėje)		1704 936 * 10 = 17049 360	1704 936 * 13 = 22164 168

Šiam metodui nagrinėjami 7 duomenų gavimo scenarijai. Scenarijai pradinei duomenų bazei pateikti 4.5 pav., o optimizuotai – 4.6 pav. Toliau kiekvienas scenarijus yra nagrinėjamas atskirai.



4.5 pav. Metodo-3 analizuojami duomenų gavimo scenarijai prieš optimizaciją



4.6 pav. Metodo-3 analizuojami duomenų gavimo scenarijai po optimizacijos

4.4.1. Pirmas scenarijus

Pirmojo scenarijaus užklausa grąžina specifinio testo visus klausimus ir jų atsakinėjimo statistiką. Panaikinus skaičiavimus užklausoje, po optimizacijos užklausa supaprastėja. Įrašų kiekis bei elementų kiekis eilutėje po optimizacijos nepasikeičia.

Mažos duomenų imties atveju užklausos vykdymo laikas pagreitėja 13,5 kartus. Su vidutine duomenų imtimi užklausa vykdoma 7,71 kartus trumpiau. Turint didelę duomenų imtį, užklausos vykdymo laikas 7,4 kartus pagreitėja. Rezultatai yra pateikti 4.25 lent.

4.25 lent. Pirmojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select q.id, q.test_id, q.title, q.picture, q.radio, q.complex, q.access_date, q.status, q.edit_date, q.code, count(case when eval=total_eval then 1 else null end), count(case when eval<total_eval and eval <> 0 then 1 else null end), count(case when eval=0 then 1 else null end) from questions q left outer join exam_test_answers e on e.qst_id = q.id where q.test_id = 1468 group by q.id	select * from questionswithcounts q where q.test_id = 1468
Elementų kiekis eilutėje	10 + 3 = 13	13
Įrašų kiekis	26	26
Vykdymo laikas	0,0054 s	0,0004 s
Įrašų kiekis	104	104
Vykdymo laikas	0,0108 s	0,0014 s
Įrašų kiekis	1664	1664
Vykdymo laikas	0,1228 s	0,0166 s

4.4.2. Antras scenarijus

Antrajame scenarijuje užklausoje grąžinama visų klausimų informacija. Po optimizacijos klausimų lentelė praplatėja 1,3 kartus (trimis elementais – stulpeliais teisingam, dalinai teisingam ir klaidingam atsakytam klausimų kiekiui saugoti). Eilučių kiekis lieka nepakitęs, o *SQL* užklausos sudėtingumas nepasikeičia.

Dėl padidėjusio lentelės dydžio, užklausa vykdomos ilgiau. Duomenų gavimo užklausa su maža duomenų imtimi vykdoma 1,15 kartų ilgiau. Esant vidutinei duomenų imčiai užklausos vykdymo laikas sulėtėja 1,15 kartų. Turint didelę duomenų imtį užklausa vykdoma daugiau nei vieną sekundę ilgiau – vykdymo laikas sulėtėja 1,22 kartus. Rezultatai yra pateikti 4.26 lent.

4.26 lent. Antrojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from questions	select * from questionswithcounts
Elementų kiekis eilutėje	10	13
Įrašų kiekis	28 644	28 644
Vykdymo laikas	0,0891 s	0,1025 s
Įrašų kiekis	108 636	108 636
Vykdymo laikas	0,3271 s	0,3752 s
Įrašų kiekis	1704 936	1704 936
Vykdymo laikas	5,1309 s	6,2474 s

Jeigu gaunami ne visi duomenys, o užklausoje yra konkretizuojami specifiniai lentelių stulpeliai, tuomet duomenų gavimo laikas nežymiai pailgėja. Esant mažai duomenų imčiai užklausa vykdoma 1,01 kartus ilgiau. Su vidutinio dydžio duomenimis užklausos vykdymo laikas pailgėja 1,15 kartų. Didelės duomenų imties atveju užklausa vykdoma 1,24 kartus lėčiau. Rezultatai yra pateikti 4.27 lent.

4.27 lent. Antrojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select q.id, q.title, q.picture, q.test_id from questions q	select q.id, q.title, q.picture, q.test_id from questionswithcounts q
Elementų kiekis eilutėje	4	4
Įrašų kiekis	28 644	28 644
Vykdymo laikas	0,1092 s	0,1098 s
Įrašų kiekis	108 636	108 636
Vykdymo laikas	0,3271 s	0,3752 s
Įrašų kiekis	1704 936	1704 936
Vykdymo laikas	2,957 s	3,6722 s

4.4.3. Trečias scenarijus

Trečiasis scenarijus grąžina specifinio testo klausimus, jų atsakinėjimo statistiką bei atsakymus. Po optimizacijos *SQL* užklausa supaprastėja, kadangi užklausoje nebereikia atlikti skaičiavimų, o pakanka nurodyti atitinkamą klausimų lentelės stulpelį. Eilučių kiekis bei elementų kiekis eilutėje nepakinta.

Esant mažai duomenų imčiai užklausos vykdymo laikas sutrumpėja 59,23 kartus. Kai duomenų kiekis yra vidutinio dydžio, užklausa vykdoma 30,2 kartus greičiau. Su didele duomenų imtimi užklausos vykdymo laikas sutrumpėja 1,9 sekundėmis (15,78 kartus). Rezultatai pateikti 4.28 lent.

4.28 lent. Trečiojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select q.id, q.test_id, q.title, q.picture, count(case when eval =total_eval then 1 else null end), count(case when eval< total_eval and eval <> 0 then 1 else null end), count(case when eval=0 then 1 else null end), a.text, a.worth from questions q left outer join answers a on q.id=a.qst_id left outer join exam_test_answers e on e.qst_id=q.id where q.test_id=1468 group by q.id, q.test_id, q.title, q.picture, a.text, a.worth	select q.id, q.test_id, q.title, q.picture, q.answ_right, q.answ_partial, q.answ_wrong, a.text, a.worth from questionswithcounts q left outer join answers a on q.id=a.qst_id where q.test_id = 1468
Elementų kiekis eilutėje	4 + 3 + 2 = 9	4 + 3 + 2 = 9
Įrašų kiekis	116	116
Vykdymo laikas	0,077 s	0,0013 s
Įrašų kiekis	464	464
Vykdymo laikas	0,1389 s	0,0046 s
Įrašų kiekis	7 424	7 424
Vykdymo laikas	2,0436 s	0,1295 s

4.4.4. Ketvirtas scenarijus

Ketvirtasis scenarijus grąžina specifinio testo klausimus, jų atsakinėjimo statistiką bei atsakymus. Tačiau priešingai nei trečiajame scenarijuje, šiuo atveju grąžinami tik tie klausimai, kurie bent vieną kartą buvo atsakyti tik neteisingai.

SQL užklausa po optimizacijos supaprastėja. Eilučių kiekis bei elementų kiekis eilutėje nepakinta. Duomenų gavimo užklausa esant mažai duomenų imčiai vykdoma 26,56 kartus greičiau. Su vidutinio dydžio duomenimis užklausos vykdymo laikas sutrumpėja 53,45 kartus. Turint didelę duomenų imtį, užklausos vykdymas sutrumpėja 29,06 kartus. Rezultatai yra pateikti 4.29 lent.

4.29 lent. Ketvirtojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select q.id, q.test_id, q.title, q.picture, q.radio, q.complex, q.access_date, q.status, q.edit_date, q.code, count(case when eval=total_eval then 1 else null end) as rigth, count(case when eval<total_eval and eval <> 0 then1 else null end) as partial, count(case when eval=0 then 1 else null end) as wrong, a.text, a.worth from questions q left outer join answers a on q.id = a.qst_id left outer join exam_test_answers e on e.qst_id =q.id where q.test_id = 462 group by q.id, a.text, a.worth having wrong > 0 and rigth = 0 and partial = 0	select q.id, q.test_id, q.title, q.picture, q.radio, q.complex, q.access_date, q.status, q.edit_date, q.code, q.answ_right, q.answ_partial, q.answ_wrong, a.text, a.worth from questionswithcounts q left outer join answers a on q.id = a.qst_id where q.test_id=462 and q.answ_wrong>0 and q.answ_right = 0 and q.answ_partial = 0
Elementų kiekis eilutėje	10 + 3 + 2 = 15	13 + 2 = 15
Įrašų kiekis	23	23
Vykdymo laikas	0,0239 s	0,0009 s
Įrašų kiekis	49	49
Vykdymo laikas	0,0855 s	0,0016 s
Įrašų kiekis	784	784
Vykdymo laikas	1,6330 s	0,0562 s

4.4.5. Penktas scenarijus

Penktasis scenarijus grąžina konkretaus studento atsakinėjimo statistiką. Šiuo atveju pateikiama atsakinėjimo statistika egzaminuose, parodant klausimus, jų atsakymus bei studento pasirinktus atsakymus. *SQL* užklausa po optimizacijos supaprastėja. Eilučių bei elementų kiekis eilutėje nepakinta, tačiau skiriasi grąžinami duomenys – prieš optimizaciją rodoma būtent to studento statistika, o po jos rodomas bendra klausimo atsakinėjimo statistika.

Duomenų gavimo užklausos vykdymo laikas esant mažai duomenų imčiai sutrumpėja 1,38 kartus. Su vidutinio dydžio duomenimis užklausa vykdoma 1,37 kartus greičiau. Didelės duomenų imties atveju užklausos vykdymo laikas sutrumpėja 1,66 kartus. Rezultatai yra pateikti 4.30 lent.

4.30 lent. Penktojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select u.fname, u.sname, e.exam_id, count(case when eval=total_eval then 1 else null end), count(case when eval<total_eval and eval <> 0 then 1 else null end), count(case when eval=0 then 1 else null end), q.id as question, q.title, e.answer, a.id, a.text, a.worth from users u left outer join exam_test_answers e on e.stud_id =u.id left outer join questions q on q.id=e.qst_id left outer join answers a on q.id=a.qst_id where u.id=15622 group by q.id, e.exam_id, e.answer, a.id	select u.fname, u.sname, e.exam_id, q.answ_right, q.answ_partial, q.answ_wrong, q.id, q.title, e.answer, a.id, a.text, a.worth from users u left outer join exam_test_answers e on e.stud_id = u.id left outer join questionswithcounts q on q.id = e.qst_id left outer join answers a on q.id=a.qst_id where u.id=15622 group by q.id, e.exam_id, e.answer, a.id
Elementų kiekis eilutėje	2 + 2 + 3 + 2 + 3 = 12	2 + 2 + 5 + 3 = 12
Įrašų kiekis	129	129
Vykdymo laikas	0,0898 s	0,0650 s
Įrašų kiekis	149	149
Vykdymo laikas	0,0911 s	0,0666 s
Įrašų kiekis	2 294	2 294
Vykdymo laikas	1,7216 s	1,0375 s

4.4.6. Šeštas scenarijus

Šeštasis scenarijus grąžina specifinio dėstytojo sukurtų klausimų atsakinėjimo statistiką. Rezultate pateikiamas dėstytojas, egzaminas, klausimas, atsakymai bei statistika. Po optimizacijos *SQL* užklausa supaprastėja, o eilučių bei elementų eilutėje kiekis nepakinta. Grąžinami rezultatai prieš ir po optimizacijos gali skirtis (statistika nesutapti), jei tas pats klausimas naudojamas keliose skirtinguose testuose (sukurtuose skirtingų dėstytojų).

Su maža duomenų imtimi užklausos vykdymo laikas pagreitėja 2,11 kartų. Esant vidutinio dydžio duomenims užklausa vykdoma 2,05 kartus trumpiau. Didelės duomenų imties atveju užklausos vykdymo laikas pagreitėja 1,91 kartą. Rezultatai yra pateikti 4.31 lent.

4.31 lent. Šeštojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select u.fname, u.sname, e.id, e.name, count(case when eval=total_eval then 1 else null end), count (case when eval<total_eval and eval <> 0 then 1 else null end), count(case when eval=0 then 1 else null end), q.id, q.title, a.text, a.worth from users u left outer join tests t on t.user_id= u.id left outer join exams e on t.id = e.test_id left outer join exam_test_answers eta on e.id = eta.exam_id left outer join questions q on q.id = eta.qst_id left outer join answers a on q.id = a.qst_id where u.id=83 group by q.id, a.text, a.worth, e.id	select u.fname, u.sname, e.id, e.name, q.answ_right, q.answ_partial, q.answ_wrong, q.id, q.title, a.text, a.worth from users u left outer join tests t on t.user_id = u.id left outer join exams e on t.id = e.test_idleft outer join exam_test_answers eta on e.id= eta.exam_id left outer join questionswithcounts q on q.id = eta.qst_id left outer join answers a on q.id = a.qst_id where u.id = 83 group by q.id, a.text, a.worth, e.id
Elementų kiekis eilutėje	2 + 2 + 3 + 2 + 2 = 11	2 + 2 + 5 + 2 = 11
Įrašų kiekis	304	304
Vykdymo laikas	0,0303 s	0,0143 s
Įrašų kiekis	316	316

Vykdymo laikas	0,0455 s	0,0222 s
Įrašų kiekis	5 041	5 041
Vykdymo laikas	0,5158 s	0,2702 s

4.4.7. Septintas scenarijus

Paskutinis trečiojo metodo scenarijus iliustruoja specifinio egzamino atsakinėjimo statistiką. Rezultate parodomas egzaminas, klausimas, atsakymas ir statistika. Po optimizacijos *SQL* užklausa supaprastėja. Eilučių bei elementų kiekis eilutėje nepakinta. Grąžinami rezultatai gali nesutapti, nes tas pats klausimas gali būti keliose egzaminuose. Tokiu atveju užklausos prieš optimizaciją rodytų būtent to egzamino statistiką, o po optimizacijos – bendrą statistiką (per visus egzaminus).

Esant mažai duomenų imčiai užklausos vykdymo laikas pagreitėja 3,05 kartus. Kai duomenys yra vidutinio dydžio, užklausa vykdoma 3 kartus trumpiau. Didelės duomenų imties atveju užklausos vykdymo laikas sutrumpėja 3,95 sekundėmis (2,98 kartus). Rezultatai yra pateikti 4.32 lent.

4.32 lent. Septintojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select eta.exam_id, q.id, q.title, a.text, a.worth, count(case when eval=total_eval then 1 else null end) as rigth, count(case when eval<total_eval and eval <> 0 then 1 else null end) as partial, count(case when eval=0 then 1 else null end) as wrong from exam_test_answers eta left outer join questions q on q.id = eta.qst_id left outer join answers a on q.id = a.qst_id where eta.exam_id = 22 group by q.id, a.id	select eta.exam_id, q.id, q.title, a.text, a.worth, q.answ_right, q.answ_partial, q.answ_wrong from exam_test_answers eta left outer join questionswithcounts q on q.id = eta.qst_id left outer join answers a on q.id =a.qst_id where eta.exam_id = 22 group by q.id, a.id
Elementų kiekis eilutėje	1 + 2 + 2 + 3 = 8	1 + 5 + 2 = 8
Įrašų kiekis	168	168
Vykdymo laikas	0,3817 s	0,1251 s
Įrašų kiekis	183	183
Vykdymo laikas	0,4215 s	0,1403 s
Įrašų kiekis	2 793	2 793
Vykdymo laikas	5,9534 s	1,9984 s

4.5. Metodo-4 eksperimentas

Ketvirtasis metodas lentelėje saugo pasikartojančią informaciją. Šiuo atveju *exams* lentelėje saugomi *user_id* ir *module_id* laukai, kurie originaliai saugomi *tests* lentelėje. Pradinėje schemoje šie laukai gaunami sujungus egzaminų ir testų lenteles. Atliekant matavimus, visų pirma matuojami lentelių dydžiai prieš ir po optimizacijos. Minėtų lentelių palyginimas pateiktas 4.33 lent. Šioje lentelėje atsižvelgiama tik į lentelių dydžius – tiek jų užimamą vietą, tiek elementų kiekį.

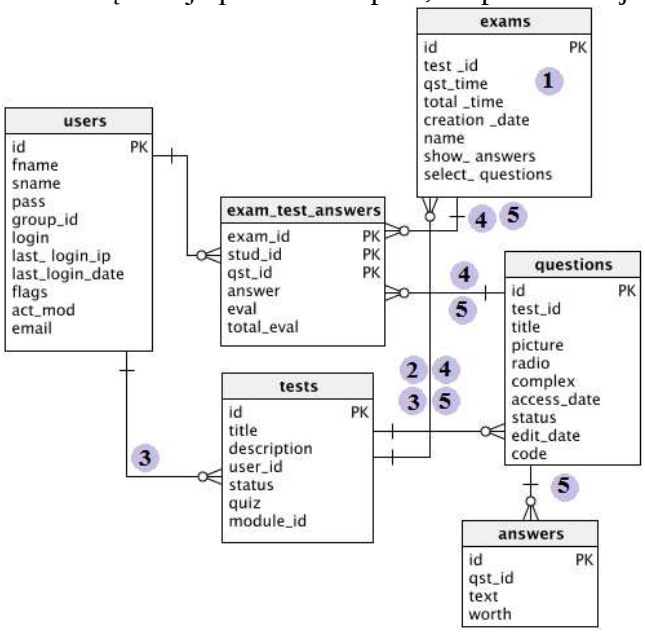
4.33 lent. Lentelių dydžių palyginimas

Parametrai \ Lentelės		Exams	ExamsRepeated
Maža duomenų imtis			
Dydis	Iš viso	0,28 MB	0,43 MB
	Duomenys	0,21 MB	0,23 MB
	Indeksai	0,07 MB	0,20 MB
Elementų kiekis (eilučių kiekis * elementų kiekis eilutėje)		2 721 * 8 = 21 768	2 721 * 10 = 27 210
Vidutinė duomenų imtis			
Dydis	Iš viso	5,24 MB	9,54 MB
	Duomenys	3,67 MB	4,77 MB
	Indeksai	1,57 MB	4,77 MB
Elementų kiekis (eilučių kiekis * elementų kiekis eilutėje)		43 521 * 8 = 348 168	43 521 * 10 = 435 210
Didelė duomenų imtis			
Dydis	Iš viso	33,66 MB	49,39 MB
	Duomenys	27,84 MB	30,99 MB
	Indeksai	5,82 MB	18,40 MB
Elementų kiekis (eilučių kiekis * elementų kiekis eilutėje)		348 161 * 8 = 2785 288	348 161 * 10 = 3481 610

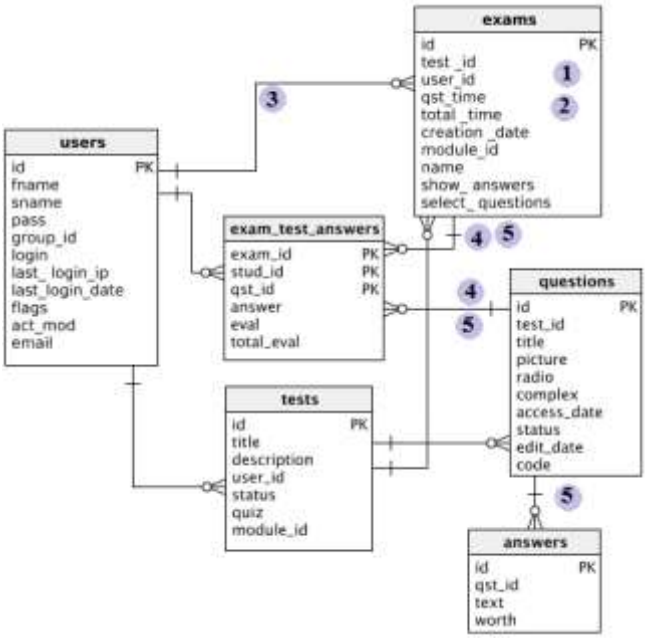
Kaip galima pastebėti iš 4.33 lent., visais duomenų imčių atvejais duomenų padaugėja. Tai galima būtų paaiškinti tuo, kad lentelėje atsiranda nauji laukai ir indeksai. Esant mažai duomenų imčiai, bendras lentelės dydis padidėja 1,54 kartus. Šiuo atveju indeksų dydis padidėja 2,86 kartus, o duomenų – 1,1 kartą. Su vidutinio dydžio duomenimis bendras lentelės dydis padidėja 1,82 kartus. Indeksų dydis padidėja 3,04 kartus, o duomenys – 1,3 kartus. Kai turima didelė duomenų imtis, lentelė padidėja 1,47 kartus. Indeksų dydis padidėja 3,16 kartų, o duomenys – 1,11 kartų.

4.33 lent. taip pat galima pastebėti, jog elementų kiekis padidėja. Nors eilučių kiekis nepakinta, bet lentelėje atsiranda du nauji stulpeliai. Tai lemia, kad bendras elementų kiekis visais duomenų dydžių atvejais padidėja 1,25 kartus.

Norint apžvelgti tokius parametrus kaip greitaveiką, nagrinėjamos konkrečios duomenų gavimo iš duomenų bazės operacijos. Tai daroma rašant konkrečias užklausas prieš optimizaciją ir po jos bei matuojant įvairias metrikas. Iš viso yra nagrinėjami 5 duomenų gavimo scenarijai. Scenarijai pradinėje duomenų bazėje pateikti 4.7 pav., o optimizuotoje – 4.8 pav. Skaičiai nurodo scenarijaus numerį.



4.7 pav. Metodo-4 analizuojami duomenų gavimo scenarijai prieš optimizaciją



4.8 pav. Metodo-4 analizuojami duomenų gavimo scenarijai po optimizacijos

4.5.1. Pirmas scenarijus

Ketvirtojo metodo pirmojo scenarijaus užklausos grąžina visų egzaminų pilną informaciją. *SQL* užklausos prieš ir po optimizacijos yra analogiškos. Eilučių kiekis nepakinta, o elementų kiekis eilutėje padidėja 1,25 kartus.

Esant mažai duomenų imčiai užklausos vykdymo laikas sulėtėja 1,05 kartus. Turint vidutinio dydžio duomenis, užklausa vykdoma 1,23 kartus greičiau. Kai turima didelė duomenų imtis, tuomet užklausos vykdymo laikas sutrumpėja 1,26 kartus. Rezultatai pateikti 4.34 lent.

4.34 lent. Pirmojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select * from exams e	select * from examsRepeated e
Elementų kiekis eilutėje	8	10
Įrašų kiekis	2 721	2 721
Vykdymo laikas	0,0118 s	0,0124 s
Įrašų kiekis	43 521	43 521
Vykdymo laikas	0,1124 s	0,0915 s
Įrašų kiekis	348 161	348 161
Vykdymo laikas	0,9667 s	0,7685 s

Keičiant užklausą ir išrenkant tam tikrus stulpelius iš klausimų lentelės, užklausos sudėtingumas lieka nepakitęs. Taip pat nepasikeičia ir eilučių bei elementų kiekis eilutėje. Duomenų gavimo užklausos vykdymo laikas pagreitėja 1,12 kartus esant mažai duomenų imčiai. Vidutinio dydžio duomenų atveju užklausa vykdoma 1,23 kartus trumpiau. Su didele duomenų imtimi užklausos vykdymo laikas pagreitėja 1,24 kartus. Rezultatai pateikti 4.35 lent.

4.35 lent. Pirmojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select e.id, e.test_id, e.name, e.creation_date from exams e	select e.id, e.test_id, e.name, e.creation_date from examsRepeated e
Elementų kiekis eilutėje	4	4
Įrašų kiekis	2 721	2 721
Vykdymo laikas	0,01116 s	0,0104 s
Įrašų kiekis	43 521	43 521
Vykdymo laikas	0,1150 s	0,0932 s
Įrašų kiekis	348 161	348 161
Vykdymo laikas	0,9781 s	0,7886 s

4.5.2. Antras scenarijus

Antrasis scenarijus grąžina specifinio dėstytojo egzaminus. Po pakeitimų užklausoje skenuojama mažiau lentelių, todėl užklausa supaprastėja. Elementų kiekis eilutėje bei įrašų kiekis nepakinta.

Esant mažai duomenų imčiai užklausos vykdymo laikas sutrumpėja 1,3 kartus. Vidutinio dydžio duomenų atveju užklausa vykdoma 2,03 kartus greičiau. Su didele duomenų imtimi, užklausos vykdomo laikas sutrumpėja 2,72 kartus. Rezultatai pateikti 4.36 lent.

4.36 lent. Antrojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select e.id, e.test_id, e.qst_time, e.total_time, e.creation_date, e.name, e.show_answers, e.select_questions from exams e left join tests t on t.id = e.test_id where t.user_id= 192	select e.id, e.test_id, e.qst_time, e.total_time, e.creation_date, e.name, e.show_answers, e.select_questions from examsRepeated e where e.user_id= 192
Elementų kiekis eilutėje	8	8
Įrašų kiekis	208	208
Vykdymo laikas	0,0034 s	0,0026 s
Įrašų kiekis	3 328	3 328
Vykdymo laikas	0,0349 s	0,0172 s
Įrašų kiekis	26 624	26 624
Vykdymo laikas	0,9779 s	0,3596 s

4.5.3. Trečias scenarijus

Trečiajame scenarijuje gaunamas tam tikras egzaminas ir su juo susijusi informacija, įskaitant dėstytoją ir modulį. Po optimizacijos užklausoje sumažėja skenuojamų lentelių kiekis, todėl *SQL* užklausa supaprastėja. Tuo tarpu elementų kiekis eilutėje bei įrašų kiekis lieka nepakitęs.

Esant mažai duomenų imčiai užklausos vykdymo laikas sutrumpėja 1,56 kartus. Turint vidutinio dydžio duomenis, užklausos vykdymas pagreitėja 3,54 kartus. Didelės duomenų imties atveju užklausos vykdymo laikas sutrumpėja 4,45 kartus. Rezultatai pateikti 4.37 lent.

4.37 lent. Trečiojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select e.id, e.test_id, u.fname, u.sname, e.qst_time, e.total_time, e.creation_date, t.module_id, e.name, e.show_answers, e.select_questions from exams e left join tests t on t.id=e.test_id left join users u on u.id t.user_id where e.id=2	select e.id, e.test_id, u.fname, u.sname, e.qst_time, e.total_time, e.creation_date, e.module_id, e.name, e.show_answers, e.select_questions from examsRepeated e left join users u on u.id = e.user_id where e.id= 2
Elementų kiekis eilutėje	11	11
Įrašų kiekis	2 721	2 721

Vykdymo laikas	0,0290 s	0,0186 s
Įrašų kiekis	43 521	43 521
Vykdymo laikas	0,6881 s	0,1944 s
Įrašų kiekis	348 161	348 161
Vykdymo laikas	6,9378 s	1,5575 s

4.5.4. Ketvirtas scenarijus

Ketvirtajame scenarijuje pateikiami modulio klausimai. Po optimizacijos užklausos vykdymo metu skenuojama viena lentele mažiau, todėl *SQL* užklausa supaprastėja. Grąžinamų duomenų kiekis lieka nepakitęs – nepasikeičia elementų kiekis eilutėje bei įrašų kiekis.

Mažos duomenų imties atveju užklausos vykdymo laikas pagreitėja 1,15 kartus. Kai duomenys yra vidutinio dydžio, tuomet užklausa vykdoma 1,21 kartus trumpiau. Turint didelę duomenų imtį, užklausos vykdymo laikas pagreitėja 1,86 kartus. Rezultatai pateikti 4.38 lent.

4.38 lent. Ketvirtojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select t.module_id, e.id, q.title from exams e left join tests t on t.id=e.test_id left join exam_test_answers eta on e.id=eta.exam_id left join questions q on eta.qst_id=q.id group by q.id, e.id	select e.module_id, e.id, q.title from examsRepeated e left join exam_test_answers eta on e.id=eta.exam_id left join questions q on eta.qst_id=q.id group by q.id, e.id
Elementų kiekis eilutėje	3	3
Įrašų kiekis	9 760	9 760
Vykdymo laikas	1,1239 s	0,9803 s
Įrašų kiekis	50 560	50 560
Vykdymo laikas	2,0520 s	1,6938 s
Įrašų kiekis	355 200	355 200
Vykdymo laikas	12,5219 s	6,7149 s

4.5.5. Penktas scenarijus

Paskutinysis ketvirtojo metodo scenarijus iliustruoja tam tikro egzamino klausimus bei atsakymus. *SQL* užklausoje skenuojama mažiau lentelių, todėl ji supaprastėja. Grąžinamų duomenų kiekis lieka nepakitęs.

Duomenų gavimo užklausos vykdymo laikas pagreitėja 1,17 kartus, kai analizuojama maža duomenų imtis. Turint vidutinio dydžio duomenis, užklausa vykdoma 1,28 kartus trumpiau. Su didele duomenų imtimi užklausos vykdymo laikas pagreitėja 1,31 kartą. Rezultatai pateikti 4.39 lent.

4.39 lent. Penktojo scenarijaus lentelių parametrų palyginimas

	Prieš optimizaciją	Po optimizacijos
SQL užklausa	select t.module_id, e.id, q.title, a.text, a.worth from exams e left join tests t on t.id = e.test_id left join exam_test_answers eta on e.id = eta.exam_id left join questions q on eta.qst_id = q.id left join answers a on q.id = a.qst_id where t.module_id = 12 and e.id = 30 group by q.id, a.id	select e.module_id, e.id, q.title, a.text, a.worth from examsRepeated e left join exam_test_answers eta on e.id = eta.exam_id left join questions q on eta.qst_id = q.id left join answers a on q.id = a.qst_id where e.module_id = 12 and e.id = 30 group by q.id, a.id
Elementų kiekis eilutėje	5	5
Įrašų kiekis	144	144
Vykdymo laikas	0,0145 s	0,0124 s
Įrašų kiekis	144	144
Vykdymo laikas	0,0197 s	0,0154 s
Įrašų kiekis	144	144
Vykdymo laikas	0,0214 s	0,0163 s

5. DUOMENŲ BAZĖS OPTIMIZACIJOS REZULTATAI

Ankstesniame eksperimentų skyriuje kiekvienam metodui buvo išmatuoti įvairūs duomenų bazės parametrai prieš ir po optimizacijos. Šiame skyriuje eksperimento matavimai yra apibendrinami. Visų pirma yra pateikiamas rezultatų įvertinimo planas. Vėliau kiekvienam metodui atskirai yra nurodomos bendro pobūdžio pastabos ir pastabos iš visų jo scenarijų. Galiausiai yra atliekamas rezultatų palyginimas tarp metodų.

5.1. Rezultatų įvertinimo planas

Kaip ir ankstesniuose skyriuose, optimizacijos rezultatai įvertinami kiekvienam optimizacijos metodui atskirai. Metodas-1 yra pasikartojančių stulpelių saugojimas lentelėje, metodas-2 yra dviejų lentelių sujungimas į vieną, metodas-3 yra skaičiuojamųjų laukų saugojimas ir metodas-4 yra pasikartojančios informacijos saugojimas lentelėje. Kiekvienai technikai yra įvertinami tokie parametrai kaip lentelių dydis, įrašų kiekis ir užklausų vykdymo laikas (greitaveika). Šie parametrai išmatuoti ankstesniame eksperimentų skyriuje. Šiame skyriuje gauti rezultatai įvertinami ir nustatoma, kaip tam tikri parametrai pagreitėjo / sulėtėjo ar sumažėjo / padidėjo. Taip pat metodams nustatomi privalumai bei trūkumai.

Išanalizavus kiekvieno metodo rezultatus, vykdomas metodų palyginimas. Metodai palyginami keliais aspektais – peržiūrimas paveiktų lentelių dydžių pokytis, atliekamas greičiau veikiančių užklausų parametrų palyginimas bei lėčiau veikiančių užklausų parametrų palyginimas. Galiausiai atliekamas metodų bruožų apibendrinimas.

5.2. Optimizacijos rezultatai

Lentelių dydis, greitaveika ir užklausų sudėtingumas analizuojamas kiekvienam optimizacijos metodui atskirai. Visai tai atliekama atskiruose poskyriuose (pagal metodus). Be to, nagrinėjami kiekvieno metodo privalumai ir trūkumai.

5.2.1. Metodo-1 rezultatai

Metodas turi tokius privalumus:

- dažnai kartu naudojami duomenys (klausimai ir atsakymai) yra saugomi vienoje lentelėje. Tai lemia, kad užklausose sumažėja arba visiškai eliminuojami sujungimai;
- sujungus lenteles sumažėja sąryšių kiekis. Tuo pačiu sumažėja pirminių ir išorinių raktų bei indeksų kiekis;
- turint mažesnį lentelių kiekį, lengviau jas valdyti.

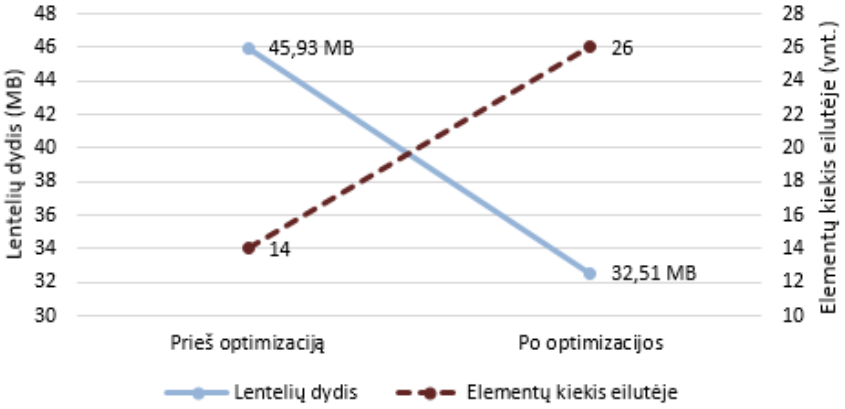
Metodo trūkumai:

- ribotas atsakymų kiekis lentelėje. Visų pirma dar projektuojant reikia nustatyti maksimalų atsakymų kiekį. Jeigu skaičius pasirenkamas per didelis, tuomet atsiranda tuščių reikšmių. Šiuo atveju, jeigu klausimas turėjo mažiau nei 8 atsakymus, naujoje lentelėje atsiranda tuščių reikšmių. Priešingai – jeigu pasirinktas skaičius yra per mažas, tuomet gali kilti problemų, kuomet reikia mažinti atsakymų kiekį vienam klausimui arba keisti duomenų bazės struktūrą (leisti suvesti daugiau atsakymų);
- po sujungimo lentelė tampa didesnė ir platesnė. Todėl ji užima daugiau vietos;
- prarandamas lankstumas atsakymų atžvilgiu – nebelieka galimybės sukurti tik atsakymus (be klausimo). Be to, naudojantis tik užklausomis negalima rasti tam tikro (pavyzdžiui, teisingo) atsakymo;
- tokie sąrašai prideda papildomo programavimo. Dėl šios priežasties modifikavimo operacijos tampa sudėtingesnės ir lėtesnės.

Sujungus *questions* ir *answers* lenteles, pasikeičia duomenų bazės dydis. Nauja *questionsWithAnswers* lentelė yra mažesnė negu originalių lentelių suma. Dydis sumažėja 1,4 kartus, kadangi panaikinamas sąryšis tarp lentelių, kurių metu nebenaudojamas išorinis raktas bei indeksas. Įrašų kiekis sumažėja 4,9 kartus ir po sujungimo jų yra tiek, kiek yra klausimų. Pradinių lentelių

elementų eilutėje suma sudarė 10 + 4 = 14, o po sujungimo elementų kiekis padidėjo 1,86 kartus – iki 26 elementų. Rezultate, įrašų kiekis sumažėja, bet lentelės eilutės platėja. Tai lemia, kad bendras elementų kiekis padidėja nežymiai – 1,01 kartus.

Lentelių dydžių palyginimo rezultatai pateikti 5.1 pav. Grafike rodomos dvi situacijos – prieš optimizaciją (lentelių sujungimą) ir po jos. Iliustruojamas tik vidutinės duomenų imties atvejis, kadangi su kitomis duomenų imtimis situacija yra analogiška. Melsva tiesia linija atvaizduojamas lentelių dydžių pokytis, o tamsiai raudona punktyrine – elementų kiekio eilutėje pokytis.



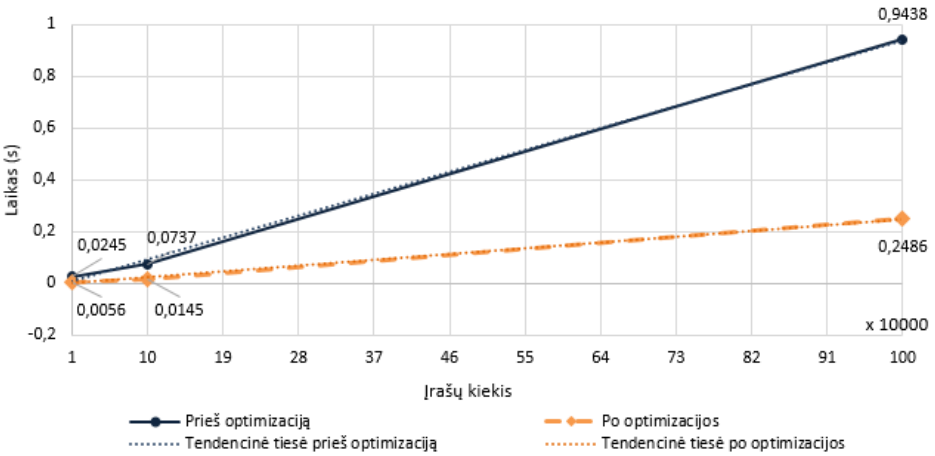
5.1 pav. Metodo-1 lentelių dydžių pasikeitimo grafikas

Užklausų, kuriose gaunami duomenys apie klausimus ir atsakymus, vykdymo laikai po lentelių sujungimo pagreitėja apie 3,35–7,96 kartus. Tokiose užklausose skenuojama viena lentele mažiau ir sumažėja sujungimų kiekis. Nors elementų kiekis eilutėje padidėja 1,28–1,85 kartus, tačiau užklausose grąžinama 3,9–5,4 kartus mažiau įrašų. Įrašų kiekis sumažėja, kadangi vienas klausimas atvaizduojamas tik vieną kartą. Originalioje schemoje užklausų rezultatuose tas pats klausimas kartojamas tiek kartų, kiek jis turi atsakymų. Po sujungimo vienas lentelės įrašas praplatėja ir jame saugoma informacija apie klausimą ir visus jo atsakymus, todėl rezultate klausimai nebedubliuojami.

Analizuojant užklausas taip pat galima pastebėti, jog kuo daugiau sujungimų juose, tuo lėčiau vykdoma užklausa. Didėjantį vykdymo laiką su didesniu sujungimų skaičiumi lemia tai, jog skenuojama daugiau lentelių ir pateikiama daugiau duomenų rezultate.

5.2 pav. pateiktas septinto scenarijaus užklausos vykdymo laikų pasikeitimo grafikas. Pasirinktas būtent toks scenarijus, kadangi jame skenuojama daugiausia lentelių. Grafike atvaizduojama informacija su visomis duomenų imtimis. Imtys nurodytos tik preliminariai – 10 000, 100 000 ir 1 000 000. Tamsiai mėlyna spalva atvaizduojami rezultatai prieš optimizaciją, o šviesiai rausva punktyrine – po. Smulkesne punktyrine linija nubrėžtos tendencinės tiesės. Tendencinės tiesės lygtis prieš optimizaciją yra y = 0,009441x - 0,001972, o po optimizacijos y = 0,002514x - 0,00345. Atsižvelgiant į tendencines tieses galima pastebėti, jog tarp vykdymo laikų ir duomenų kiekių yra gana tiesinė priklausomybė. Analizuojant rezultatus, galima pastebėti, jog po optimizacijos tendencinė tiesė tampa nuožulnesnė.

Šiuo atveju užklausos vykdymo laikas vidutiniškai sutrumpėja apie 4,3 kartus. Esant mažai duomenų imčiai užklausa vykdoma 4,37 kartus trumpiau, su vidutine imtimi – 5,08 kartus, o su didele imtimi – 3,8 kartus. Pagal tendencines kreives, turint 2 000 000 įrašų, prieš optimizaciją vykdymo laikas būtų lygus 1,9 sekundėms, o po – 0,5.

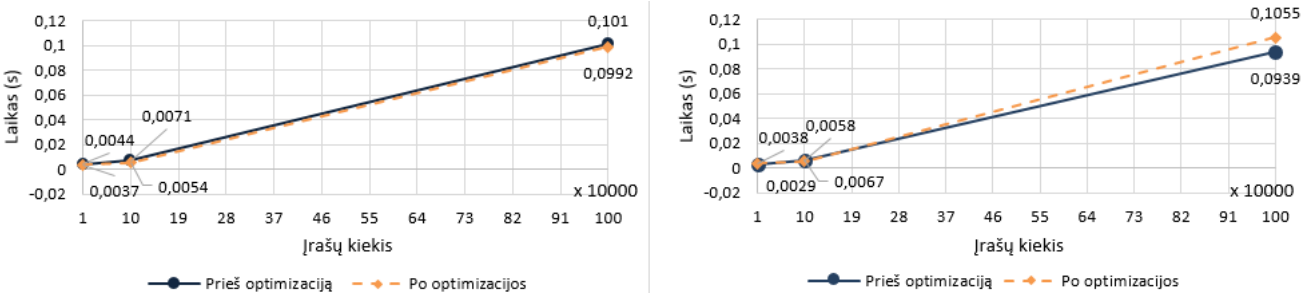


5.2 pav. Metodo-1 užklausos vykdymo laiko pasikeitimo grafikas

Gaunant tik klausimų informaciją, užklausų vykdymo laikai pagreitėja nežymiai (1,02–1,3 karto) arba sulėtėja 1,12–1,68 kartus. Be to užklausa tampa sudėtingesnė, jeigu norima gauti tik klausimus. Tokiu atveju užklausoje turi būti išvardinti visi gaunami lentelių stulpeliai. Užklausų vykdymo laikai, kai gaunami tik klausimai ir testai (ketvirtas scenarijus), pateikti 5.3 pav. Tamsiai mėlyna spalva atvaizduojama situacija prieš optimizaciją, o šviesiai rausva punktyrine – po. Šiuo atveju:

- *select <rows> from Q join T* reiškia, kad gaunami tik reikalingo stulpeliai (nurodomi *<rows>* dalyje) iš klausimų ir testų lentelių;
- *select * from Q join T* reiškia, kad traukiami visi duomenys iš klausimų ir testų lentelių.

Jeigu duomenys išrenkami ir grąžinama tik su klausimais ir testais susijusi informacija (žr. 5.3 pav. a)), tuomet užklausų vykdymo laikai nežymiai pagreitėja (apie 1,17 karto). Jeigu gaunami visi lentelės duomenys (žr. 5.3 pav. b)), tuomet užklausos po optimizacijos vykdomos ilgiau (apie 1,2 kartus). Pastaruoju atveju lentelės eilutė praplatėja, todėl bendrai grąžinama 1,94 kartus daugiau elementų eilutėje. Įrašų kiekis nepakinta. Tuo tarpu pirmuoju atveju grąžinamų įrašų kiekis bei elementų kiekis eilutėje po optimizacijos lieka nepakitęs.



a) *select <rows> from Q join T* b) *select * from Q join T*

5.3 pav. Metodo-1 užklausos vykdymo laiko pasikeitimo grafikas

5.2.2. Metodo-2 rezultatai

Metodas panašus į metodą-1, kadangi abiem atvejais sujungiamos *questions* ir *answers* lentelės. Šiuo atveju sujungimas vykdomas atsakymų atžvilgiu. Šio metodo privalumai analogiški metodo-1 privalumams:

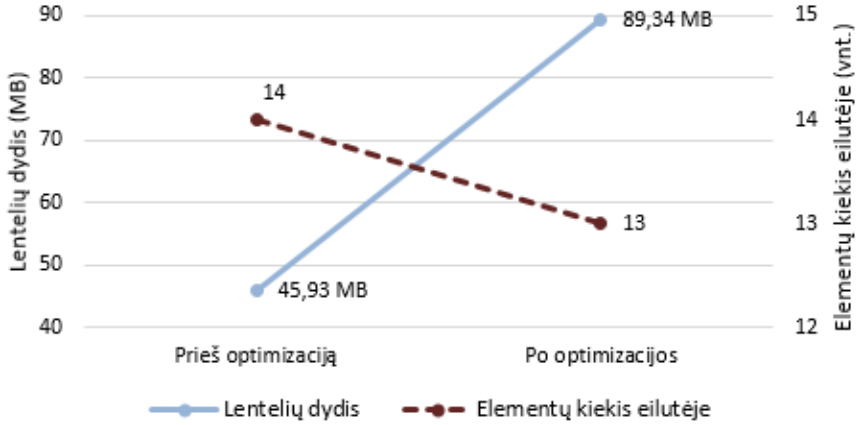
- atsakymai ir klausimai saugomi vienoje lentelėje, todėl užklausose nebereikalingas dviejų lentelių sujungimas;
- dėl panaikinto sąryšio, sumažėja raktų bei lentelių kiekis;
- lengviau valdyti lenteles, nes jų yra mažiau.

Metodo-2 trūkumai susiję su klausimų lankstumo stoka:

- neįmanoma sukurti tik klausimo, kadangi lentelės identifikatorius yra atsakymas;
- klausimo modifikacija sudėtingėja, nes vienas klausimas yra kartojamas prie daugelio įrašų. Visa tai gali privesti prie duomenų anomalijų;
- klausimų dublikatai smarkiai padidina lentelės užimamą vietą.

Metodo-2 atveju tas pats klausimas kartojasi ne viename įraše. Dėl šios priežasties, duomenų bazė po optimizacijos užima daugiau vietos. Nauja *answersWithQuestions* lentelė užima 1,9 kartus daugiau vietos negu pradinių lentelių bendra dydžių suma. Įrašų kiekis sumažėja 1,25 kartus ir po sujungimo jų yra tiek, kiek yra atsakymų. Pradinių lentelių elementų eilutėje suma sudarė 10 + 4 = 14, o po sujungimo elementų kiekis sumažėjo 1,08 kartus. Elementų kiekis eilutėje sumažėja dėl panaikinto sąryšio tarp lentelių, kurių metu nebenaudojamas vienas išorinis raktas. Pradiniu atveju daugiau įrašų yra atsakymų lentelėje, kuri yra siauresnė negu klausimų lentelė. Po sujungimo lentelė turi tiek pat įrašų, kiek ir atsakymų lentelė, tačiau lyginant su atsakymų lentele, elementų kiekis eilutėje didesnis. Tai lemia, kad bendras elementų kiekis padidėja 1,99 kartus.

Lentelių dydžių palyginimo rezultatai pateikti 5.4 pav. Grafike rodomos dvi situacijos – prieš optimizaciją (lentelių sujungimą) ir po jos. Iliustruojamas tik vidutinės duomenų imties atvejis, kadangi su kitomis duomenų imtimis situacija yra analogiška. Melsva tiesia linija atvaizduojamas lentelių dydžių pokytis, o tamsiai raudona punktyrine – elementų kiekio eilutėje pokytis.

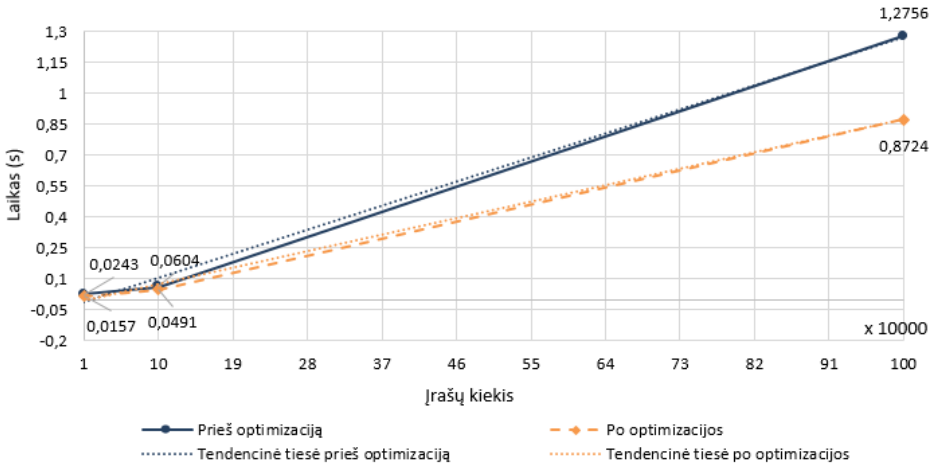


5.4 pav. Metodo-2 lentelių dydžių pasikeitimo grafikas

Užklausų, kuriose gaunami duomenys apie klausimus ir atsakymus, vykdymo laikai po lentelių sujungimo pagreitėja apie 1,16–3,1 kartus. Užklausose skenuojama viena lentele mažiau ir sumažėja sujungimų kiekis. Elementų kiekis eilutėje sumažėja apie 1,02 kartus (vienu elementu), o užklausose grąžinamas įrašų kiekis nepakinta. Analizuojant užklausas taip pat pastebėta, jog kuo daugiau sujungimų juose, tuo lėčiau vykdoma užklausa. Didėjantį vykdymo laiką su didesniu sujungimų skaičiumi lemia tai, jog skenuojama daugiau lentelių ir pateikiama daugiau duomenų rezultate.

5.5 pav. pateiktas septinto scenarijaus užklausos vykdymo laikų pasikeitimo grafikas. Pasirinktas būtent toks scenarijus, kadangi jame skenuojama daugiausia lentelių. Grafike atvaizduojama informacija su visomis duomenų imtimis. Imtys nurodytos tik preliminariai – 10 000, 100 000 ir 1 000 000. Tamsiai mėlyna spalva atvaizduojami rezultatai prieš optimizaciją, o šviesiai rusva punktyrine – po. Smulkesne punktyrine linija nubrėžtos tendencinės tiesės. Tendencinės tiesės lygtis prieš optimizaciją yra y = 0,01299x - 0,02717, o po optimizacijos y = 0,008854x - 0,01519. Atsižvelgiant į tendencines tieses galima pastebėti, jog tarp vykdymo laikų ir duomenų kiekių yra gana tiesinė priklausomybė. Analizuojant rezultatus, galima pastebėti, jog po optimizacijos tendencinė tiesė tampa nuožulnesnė.

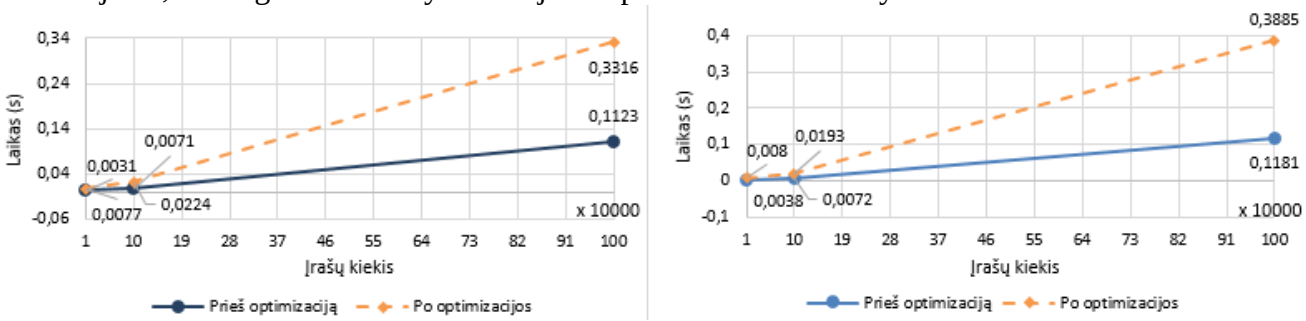
Šio scenarijaus atveju po optimizacijos užklausos vykdymo laikas vidutiniškai sutrumpėja apie 1,41 kartus. Su maža duomenų imtimi užklausa vykdoma 1,54 kartus greičiau, su vidutinio dydžio duomenimis – 1,23 kartus, o su didele imtimi – 1,46 kartus. Pagal tendencines kreives, turint 2 000 000 įrašų, prieš optimizaciją vykdymo laikas būtų lygus 2,6 sekundėms, o po – 1,75.



5.5 pav. Metodo-2 užklausų vykdymo laikų palyginimo grafikas

Gaunant tik klausimų informaciją, užklausų vykdymo laikai sulėtėja 2,1–3,6 kartus. Jeigu gaunama tik atsakymų informacija – užklausų vykdymo laikai sulėtėja apie 1,3 kartus, pagreitėja 0,87 kartus arba nepasikeičia. Pokyčiai priklauso nuo užklausos ir nuo duomenų imties dydžio.

Užklausos tampa sudėtingesnės, jeigu norima gauti tik klausimus ar tik atsakymus. Tokiu atveju užklausoje turi būti išvardinti visi dominantys lentelių stulpeliai. Užklausų vykdymo laikai, kai gaunami tik klausimai ir testai, pateikti 5.6 pav. Žymėjimas analogiškas metodui-1. Jeigu duomenys išrenkami ir grąžinama tik su klausimais ir testais susijusi informacija (žr. 5.6 pav. a)) ar gaunami visi lentelės duomenys (žr. 5.6 pav.b)), tuomet užklausos po optimizacijos vykdomos ilgiau (apie 2,7 kartus). Pirmuoju atveju grąžinama tiek pat elementų eilutėje kaip ir prieš optimizaciją. Kitu atveju grąžinama 1,18 kartų daugiau elementų eilutėje. Užklausos vykdymo laikas sulėtėja dėlto, jog po sujungimo grąžinama 4,82 kartus daugiau įrašų. Tokiame rezultate vieno klausimo informacija yra dubliuojama, kadangi klausimas yra kartojamas prie kiekvieno atsakymo.



a) *select <rows> from Q join T* b) *select * from Q join T*

5.6 pav. Metodo-2 užklausų vykdymo laikų palyginimo grafikas

5.2.3. Metodo-3 rezultatai

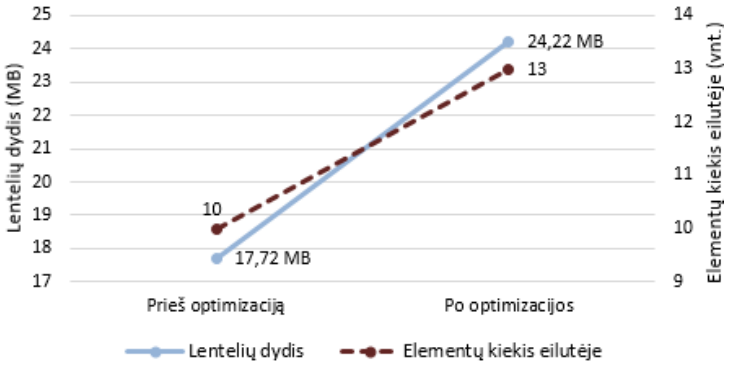
Metodo privalumai:

- skaičiuojamųjų laukų saugojimas lentelėje supaprastina *SQL* užklausas – juose nebereikia atlikti skaičiavimų ir naudoti tokius agregatus kaip *count()*. Vietoje to pakanka nurodyti lentelėje stulpelį, kuriame saugoma atitinkama reikšmė;
- kadangi skaičiavimai nebeatliekami duomenų gavimo metu, tuomet *SELECT* užklausos vykdomos greičiau.

Metodo trūkumai:

- norint, kad reikšmės būtų teisingos, jos turi būti reguliariai perskaičiuojamos. Kiekvieno reikšmingo įrašo modifikavimo atveju turi būti perskaičiuojami apskaičiuoti laukai. Tai lemia, kad pailgėja modifikavimo operacijų vykdymo laikai;
- išsaugoti apskaičiuoti laukai gali nepadengti visų reikiamų atvejų – pavyzdžiui, lentelėje saugoma bendra statistika, o nurodytu atveju reikalinga konkretaus vartotojo statistika;
- lentelė tampa didesnė ir platesnė.

Pritaikius trečiąjį optimizacijos metodą, lentelėje prisideda trys nauji stulpeliai – teisingai, dalinai teisingai ir neteisingai atsakytų klausimų kiekis. Elementų kiekis eilutėje bei lentelės dydis po optimizacijos padidėja apie 1,3 kartus. Priklausomybė tarp šių dviejų dydžių yra tiesioginė. Įrašų kiekis po optimizacijos nepakinta. Lentelės dydžių palyginimas prieš ir po optimizacijos pateiktas 5.7 pav. Kaip ir su kitais metodais, šiuo atveju iliustruojama tik vidutinės duomenų imties dydžiai.



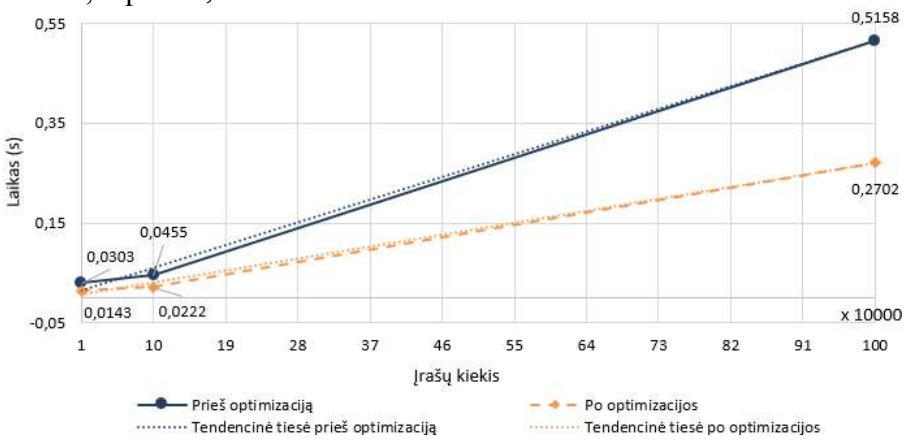
5.7 pav. Metodo-3 lentelių dydžių pasikeitimo grafikas

Užklausų, kuriose yra skaičiuojami laukai, vykdymo laikai po metodo pritaikymo pagreitėja apie 1,3–59,2 kartus. Tam tikros užklausos (pirmas, trečias ir ketvirtas scenarijus) po optimizacijos supaprastėjo tiek, kad užklausoje sumažėjo lentelių bei sujungimų kiekis. Tokių užklausų vykdymo laikai pagreitėjo daugiausiai – virš 7,4 kartų. Kitų užklausų, kuriose skenuojamų lentelių kiekis nepakinta, vykdymo laikai pagreitėja iki 3,05 kartų.

Po optimizacijos lentelė praplatėja, tačiau užklausoje nurodomi konkretūs laukai, todėl grąžinamų duomenų kiekis vienodas – nepakinta elementų kiekis eilutėje bei grąžinamų įrašų kiekis. Nors po optimizacijos skenuojama didesnė lentelė, tačiau agregatų pašalinimas turi ryškesnį efektą greitaveikai, todėl užklausos vykdomos greičiau. Be to, po optimizacijos užklausos, kuriose gaunama klausimų atsakinėjimo statistika, supaprastėja. Taip yra todėl, jog nebereikalingi įvairūs agregatai (tokie kaip *count()*), o pakanka tik nurodyti atitinkama klausimų lentelės stulpelį.

5.8 pav. pateiktas šeštojo scenarijaus užklausos vykdymo laikų pasikeitimo grafikas. Šiame scenarijuje skenuojama daugiausia lentelių (šešios). Grafike atvaizduojama informacija su visomis duomenų imtimis. Imtys nurodytos tik preliminariai – 10 000, 100 000 ir 1 000 000. Tamsiai mėlyna spalva atvaizduojami rezultatai prieš optimizaciją, o šviesiai rausva punktyrine – po. Smulkia punktyrine linija nubrėžtos tendencinės tiesės. Tendencinės tiesės lygtis prieš optimizaciją yra y = 0,005034x + 0,01093, o po optimizacijos y = 0,002654x + 0,004033. Atsižvelgiant į tendencines tieses galima pastebėti, jog tarp vykdymo laikų ir duomenų kiekių yra gana tiesinė priklausomybė. Analizuojant rezultatus, pastebėta, jog po optimizacijos tendencinė tiesė yra nuožulnesnė.

Šiuo atveju užklausos užklausos vykdymo laikas vidutiniškai sutrumpėja 2,02 kartus. Mažai duomenų imčiai užklausa pagreitėja 2,02 kartus, vidutinei – 2,05 kartus, o didelei duomenų imčiai – 1,91 kartą. Pagal tendencines kreives, turint 2 000 000 įrašų, prieš optimizaciją vykdymo laikas būtų lygus 1,02 sekundėms, o po – 0,52.

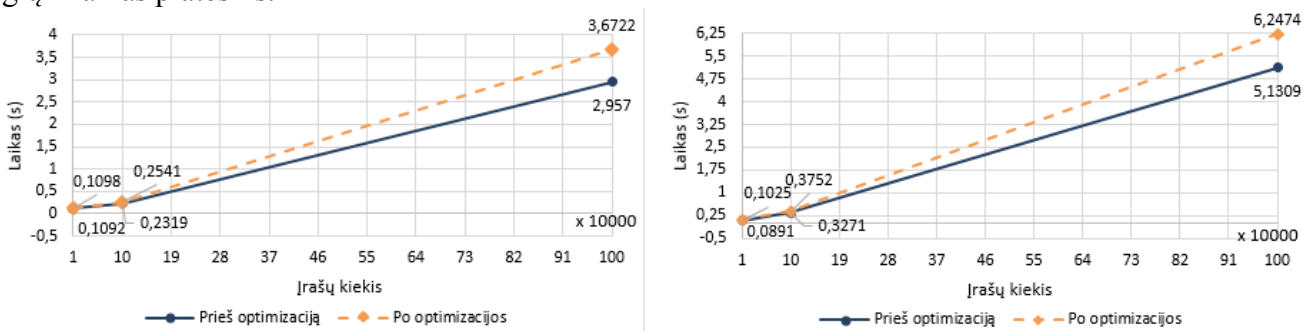


5.8 pav. Metodo-3 užklausos vykdymo laiko pasikeitimo grafikas

Užklausos, kuriose nenaudojami skaičiuojami laukai, nežymiai pailgėja. Taip yra todėl, jog klausimų lentelė praplatėja ir skenuojama didesnė lentelė. Tokiu atveju, gaunant tik klausimų informaciją, užklausų vykdymo laikai sulėtėja 1,01–1,24 kartus. Be to užklausos tampa sudėtingesnės, jeigu norima gauti tik klausimus. Tokiu atveju užklausoje turi būti išvardinti visi gaunami lentelių stulpeliai. Užklausų vykdymo laikai, kai gaunami tik klausimai, pateikti 5.9 pav. Tamsiai mėlyna spalva atvaizduojama situacija prieš optimizaciją, o šviesiai rausva punktyrine – po. Šiuo atveju:

- *select <rows> from Q* reiškia, kad gaunami tik reikalingo stulpeliai (nurodomi *<rows>* dalyje) iš klausimų lentelės;
- *select * from Q* reiškia, kad traukiami visi duomenys iš klausimų lentelės.

Jeigu duomenys išrenkami ir grąžinama tik tam tikra klausimų lentelės informacija (žr. 5.9 pav. a)), tuomet užklausų vykdymo laikai sulėtėja apie 1,1 kartą. Šiuo atveju grąžinama tiek pat įrašų bei elementų eilutėje. Jeigu gaunami visi lentelės duomenys (žr. 5.9 pav. b)), tuomet užklausos po optimizacijos vykdomos ilgiau (apie 1,16 karto). Nors įrašų grąžinama tiek pat, kaip ir prieš optimizaciją, tačiau elementų kiekis eilutėje padidėja 1,3 kartus. Tai lemia, kad vienas įrašas grąžinamas platesnis.



a) *select <rows> from Q* b) *select * from Q*

5.9 pav. Metodo-3 užklausos vykdymo laiko pasikeitimo grafikas

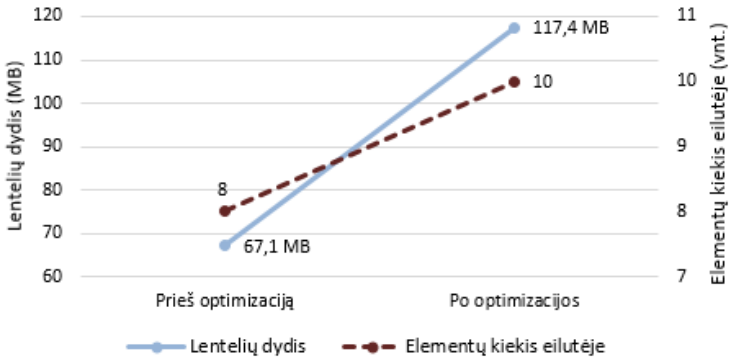
5.2.4. Metodo-4 rezultatai

Metodo privalumas yra toks, kad egzaminui aktuali informacija yra saugoma vienoje lentelėje. Vadinasi, užklausose sumažėja arba eliminuojami sujungimai. Metodo trūkumai:

- lentelė tampa platesnė ir didesnė;
- atsiranda duomenų dublikatai;
- daugiau indeksų, todėl duomenų bazė užima daugiau vietos;
- dėl pasikartojančių duomenų skirtingose lentelėse, duomenų modifikavimas tampa sudėtingesnis ir gali atsirasti duomenų anomalijos.

Išsaugojus vartotojo ir modulio identifikatorius egzaminų lentelėje, pasikeičia duomenų bazės dydis. Nauja lentelė yra 1,82 kartus didesnė negu originali lentelė, kadangi prisideda du nauji laukai. Negana to, kad padaugėja duomenų, tačiau šie laukai yra indeksai, todėl padidėja ir indeksų dydis. Įrašų kiekis lieka nepakitęs. Elementų kiekis eilutėje padidėja 1,25 kartus. Tai lemia, kad bendras elementų kiekis padidėja 1,25 kartus.

Lentelių dydžių palyginimo rezultatai pateikti 5.10 pav. Grafike rodomos dvi situacijos – prieš optimizaciją ir po jos. Iliustruojamas tik vidutinės duomenų imties atvejis, kadangi su kitomis duomenų imtimis situacija yra analogiška. Melsva tiesia linija atvaizduojamas lentelių dydžių pokytis, o tamsiai raudona punktyrine – elementų kiekio eilutėje pokytis.

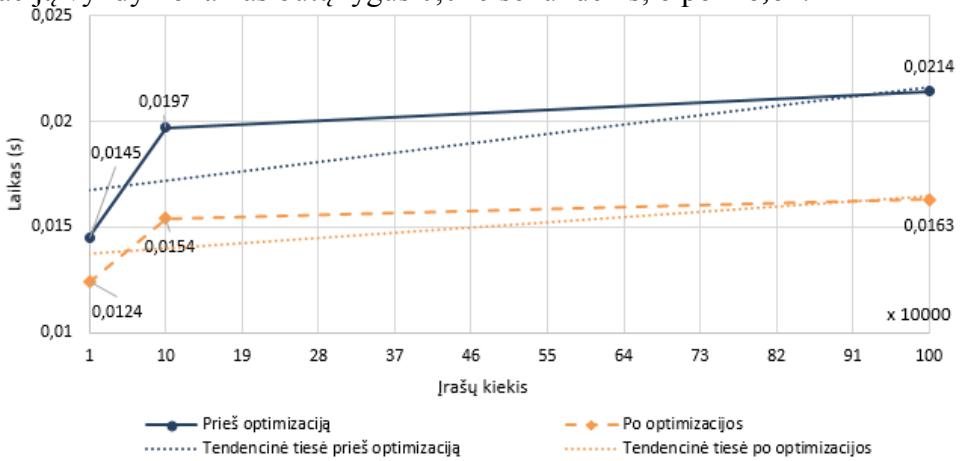


5.10 pav. Metodo-4 lentelių dydžių pasikeitimo grafikas

Užklausų, kuriose gaunami duomenys apie egzaminus, vykdymo laikai po metodo pritaikymo pagreitėja apie 1,14–4,45 kartus. Elementų kiekis eilutėje bei grąžinamų įrašų kiekis nepakinta. Originalioje schemoje norint gauti kartu su egzamino duomenimis ir dėstytojo ar modulio informaciją, reikia sujungti egzaminų ir testų lenteles. Po optimizacijos šie laukai saugomi egzaminų lentelėje, todėl nagrinėjamose užklausoje nebereikia skenuoti testų lentelės. Tai lemia, kad užklausose skenuojama viena lentele mažiau ir sumažėja sujungimų kiekis.

5.11 pav. pateiktas penktojo scenarijaus užklausos vykdymo laikų pasikeitimo grafikas. Pasirinktas būtent toks scenarijus, kadangi jame skenuojama daugiausia lentelių (pradžioje penkios, o po optimizacijos – keturios). Grafike atvaizduojama informacija su visomis duomenų imtimis. Imtys nurodytos tik preliminariai – 10 000, 100 000 ir 1 000 000 įrašų. Tamsiai mėlyna spalva atvaizduojami rezultatai prieš optimizaciją, o šviesiai rausva punktyrine – po. Smulkia punktyrine linija nubrėžtos tendencinės tiesės. Tendencinės tiesės lygtis prieš optimizaciją yra y = 0,0000491x + 0,01672, o po optimizacijos y = 0,00002748x + 0,01368. Atsižvelgiant į tendencines tieses galima pastebėti, jog tarp vykdymo laikų ir duomenų kiekių yra gana tiesinė priklausomybė.

Šiuo atveju užklausos vykdymo laikas vidutiniškai sutrumpėja apie 1,25 kartus. Esant mažai duomenų imčiai užklausos vykdymo laikas pagreitėja 1,17 kartų, esant vidutinei imčiai – 1,28 kartus ir su didele duomenų imtimi pagreitėja 1,31 kartus. Pagal tendencines kreives, turint 2 000 000 įrašų, prieš optimizaciją vykdymo laikas būtų lygus 0,026 sekundėms, o po – 0,02.



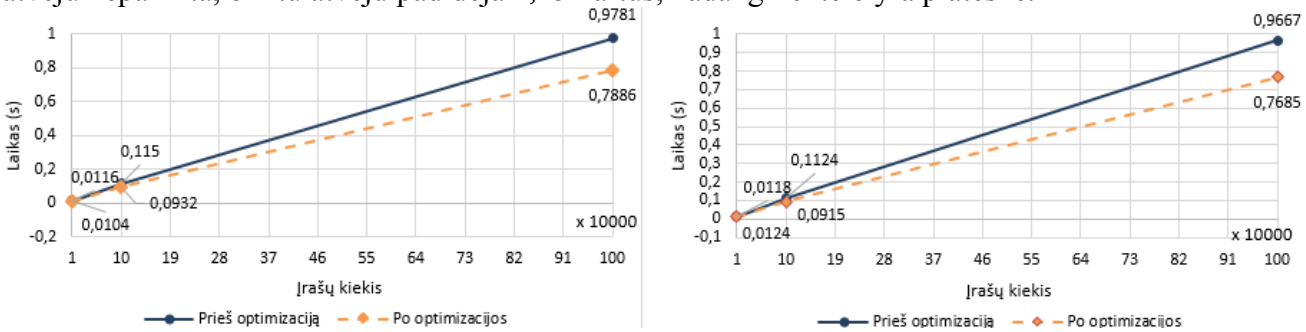
5.11 pav. Metodo-4 užklausos vykdymo laiko pasikeitimo grafikas

Jeigu gaunama tik egzaminų informacija, tuomet užklausų vykdymo laikai pagreitėja nežymiai (1,05–1,25 kartus) arba 1,05 kartus sulėtėja. Užklausų vykdymo laikai, kai gaunami tik egzaminai, pateikti 5.12 pav. Tamsiai mėlyna spalva atvaizduojama situacija prieš optimizaciją, o šviesiai rausva punktyrine – po. Šiuo atveju:

- *select <rows> from E* reiškia, kad gaunami tik reikalingo stulpeliai (nurodomi *<rows>* dalyje) iš egzaminų lentelės;
- *select * from E* reiškia, kad traukiami visi duomenys iš egzaminų lentelės.

Jeigu išrenkami konkretūs stulpeliai iš egzaminų lentelės (žr. 5.12 pav. a)), tuomet užklausų vykdymo laikai nežymiai pagreitėja (apie 1,19 karto). Jeigu gaunami visi lentelės duomenys (žr. 5.12

pav. b)), tuomet užklausa esant mažai duomenų imčiai vykdoma 1,05 kartus ilgiau, o kitais atvejais apie 1,3 kartus greičiau. Abiem atvejais grąžinama tiek pat įrašų. Elementų kiekis eilutėje pirmuoju atveju nepakinta, o kitu atveju padidėja 1,25 kartus, kadangi lentelė yra platesnė.



a) *select <rows> from E* b) *select * from E*

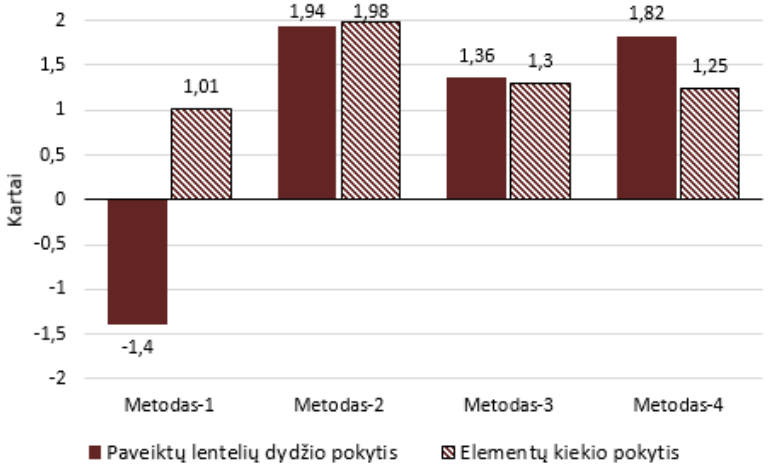
5.12 pav. Metodo-4 užklausos vykdymo laiko pasikeitimo grafikas

5.3. Optimizacijos metodų palyginimas

Pritaikius pirmąjį metodą, paveiktų lentelių dydis sumažėja 1,4 kartus. Šiuo atveju duomenų sumažėja 1,16 kartų, o indeksų dydis sumažėja 3,2 kartus. Taip yra todėl, kad panaikinami du stulpeliai, iš kurių vienas yra indeksinis. Antrojo metodo atveju paveiktų lentelių dydis padidėja 1,94 kartus. Šiuo atveju, dėl klausimų dubliavimo prie atsakymų, duomenų kiekis padidėja 2,08 kartus, o indeksų dydis padidėja 1,57 kartus. Pritaikius trečiąjį metodą, dėl pridėtų trijų naujų laukų, paveikta lentelė padidėja 1,36 kartus. Šiuo atveju duomenų padidėja 1,27 kartus, o indeksų dydis padidėja 1,67 kartus. Ketvirtojo metodo atveju paveikta lentelė dėl pridėtų naujų laukų padidėja 1,82 kartus. Kadangi nauji laukai yra indeksiniai, todėl indeksų dydis padidėja 3,03 kartus, o duomenys – 1,3 kartus.

Pirmojo metodo paveiktų lentelių įrašų kiekis sumažėja 4,9 kartus. Elementų kiekis eilutėje padidėjo 1,85 kartus. Tai lemia, kad bendras elementų kiekis padidėjo 1,01 kartą. Pritaikius antrąjį metodą, paveiktų lentelių įrašų kiekis sumažėja 1,25 kartus. Elementų kiekis eilutėje pamažėja 1,07 kartus, tačiau bendras elementų kiekis padidėja 1,98 kartus. Trečiojo metodo atveju paveiktų lentelių įrašų kiekis lieka nepakitęs. Tiek elementų kiekis eilutėje, tiek bendras elementų kiekis padidėja 1,3 kartus. Ketvirtojo metodo atveju paveiktų lentelių įrašų kiekis nepasikeičia. Elementų kiekis eilutėje padidėja 1,25 kartus. Bendras elementų kiekis pasikeičia atitinkamai – padidėja 1,25 kartus.

Parametrų palyginimai pateikti esant vidutinei duomenų imčiai. Su kitomis imtimis rezultatai yra panašūs. 5.13 pav. pateikti metodų paveiktų lentelių dydžių pokyčiai. Vientisos spalvos stulpeliai atvaizduoja lentelių dydžių pokytį. Įstrižomis juostelėmis nubraižyti stulpeliai nurodo bendro elementų kiekio paveiktose lentelėse pokytį. Šiuo atveju pokyčiai atvaizduojami kartais. Jeigu pokytis toks, jog atitinkamas parametras sumažėja, tuomet grafiko stulpelis turi neigiamas reikšmes. Priešingai – jeigu dydis padidėja, tuomet stulpelis yra teigiamas. Prie kiekvieno stulpelio nurodyta pokyčio (kartais) reikšmė. Reikšmė gaunama dalinant parametrus prieš optimizaciją ir po jos ar atvirkščiai.



5.13 pav. Metodų paveiktų lentelių dydžių pokyčiai

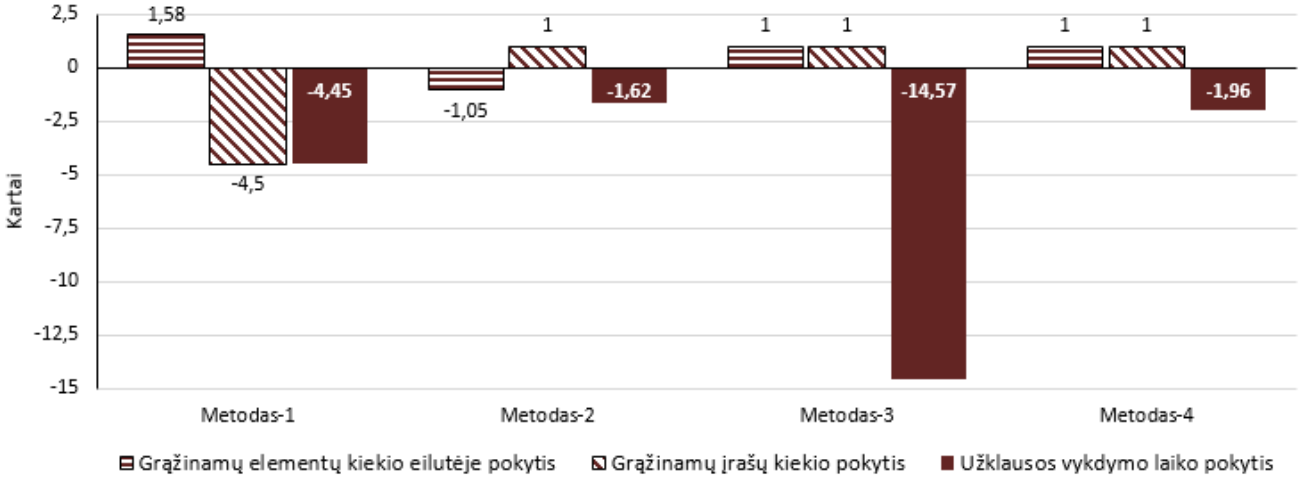
Pritaikius pirmąjį metodą, užklausose skenuojama viena lentele mažiau. Klausimų ir atsakymų informacija saugoma vienoje lentelėje, todėl užklausose sumažėja sujungimų kiekis. Pritaikius antrąjį metodą, užklausose taip pat skenuojama viena lentele mažiau. Kaip ir pirmuoju atveju, klausimų ir atsakymų informacija yra saugoma vienoje lentelėje, todėl užklausose sumažėja sujungimų kiekis. Trečiojo metodo atveju vienose užklausose skenuojama viena lentele mažiau (kadangi aktuali informacija saugoma klausimų lentelėje), o kitose skenuojamų lentelių kiekis nepasikeičia. Ketvirtojo metodo atveju užklausose skenuojama viena lentele mažiau. Taip yra todėl, jog aktuali informacija saugoma egzaminų lentelėje ir todėl nebereikalingas sujungimas su testų lentele.

Pirmojo metodo užklausose grąžinama 1,28–1,85 kartus daugiau elementų eilutėje. Taip yra todėl, jog lentelė prasiplečia ir joje saugoma tiek klausimo, tiek atsakymo informacija. Pritaikius antrąjį metodą, pašalinamas išorinio rakto stulpelis, siejantis klausimų ir atsakymų lenteles. Dėl šios priežasties lentelėje saugoma 1,02–1,07 kartus mažiau elementų eilutėje. Trečiojo bei ketvirtojo metodo atveju užklausose grąžinamų elementų eilutėje kiekis nepakinta.

Pirmojo metodo užklausų rezultatuose klausimai nebedubliuojami, o klausimas ir jo atsakymai saugomi viename įraše. Dėl šios priežasties grąžinamų įrašų kiekis sumažėja 3,9–5,4 kartus. Pritaikius antrąjį metodą, grąžinamų įrašų kiekis nepakinta, kadangi duomenys saugomi taip, kaip yra atvaizduojami originalios schemos užklausoje atlikus sujungimą tarp lentelių. Tai lemia, kad tas pats klausimas dubliuojamas prie kiekvieno atitinkamo atsakymo. Trečiojo ir ketvirtojo metodų užklausose grąžinamų įrašų kiekis išlieka nepakitęs.

Pirmojo metodo užklausų vykdymo laikai pagreitėja 3,35–5,55 kartus. Pritaikius antrąjį metodą, užklausų vykdymo laikai pagreitėja 1,11–3,09 kartus. Trečiojo metodo atveju užklausos, kuriose skenuojama viena lentele mažiau ir mažėja sujungimų skaičius, vykdomos 13,5–59,23 kartus greičiau. Jeigu užklausoje skenuojama tiek pat lentelių kaip ir originalios schemos atveju, tuomet užklausos vykdymo laikas sutrumpėja 1,36–3,05 kartus. Ketvirtojo metodo atveju užklausų vykdymo laikas pagreitėja 1,14–4,45 kartus.

Metodų palyginimo parametrai pateikti 5.14 pav. Šiuo atveju yra išskaičiuojami scenarijų parametrų vidurkiai. Vientisos spalvos stulpeliai atvaizduoja užklausų vykdymo laikų pokytį. Įstrižomis juostelėmis nubraižyti stulpeliai nurodo grąžinamų įrašų užklausose kiekio pokytį. Horizontaliomis linijomis nubraižyti stulpeliai iliustruoja užklausose grąžinamų elementų eilutėje kiekio pokytį. Šiuo atveju pokyčiai atvaizduojami kartais. Jeigu pokytis toks, jog atitinkamas parametras sumažėja ar sutrumpėja, tuomet grafiko stulpelis turi neigiamas reikšmes. Priešingai – jeigu dydis padidėja ar sulėtėja, tuomet stulpelis yra teigiamas. Jeigu reikšmė yra lygi vienetui, tuomet atitinkamo parametro dydis nepasikeičia. Prie kiekvieno stulpelio nurodyta pokyčio (kartais) reikšmė.



5.14 pav. Metodų užklausos parametrų pokyčiai

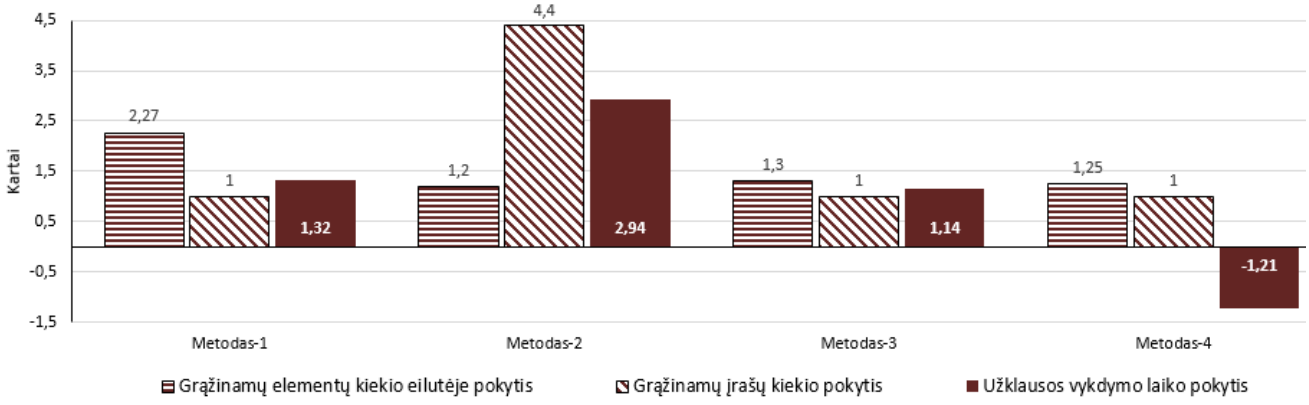
Anksčiau nagrinėjamose užklausos, kurios turėjo geresnius rezultatus ir trumpesnius užklausų vykdymo laikus. Šiuo atveju nagrinėjamos užklausos, kurių vykdymo laikai blogesni. Visų metodų tokiose užklausose skenuojamų lentelių kiekis nepasikeičia.

Pirmojo metodo atveju skenuojant visus lentelės duomenis, grąžinama 1,94–2,6 kartus daugiau elementų eilutėje, kadangi lentelė pasidaro platesnė. Jeigu duomenys išrenkami, tuomet grąžinamų elementų kiekis nepakinta. Antrojo metodo atveju gaunant visus klausimų lentelės duomenis, grąžinama 1,17–1,3 kartus daugiau elementų eilutėje. Jeigu gaunami visi atsakymų lentelės duomenys, tuomet grąžinama 3,25 kartus daugiau elementų eilutėje. Taip yra todėl, jog po optimizacijos lentelėje saugomi atsakymai ir klausimai. Užklausoje išrinkus stulpelius, elementų kiekis eilutėje nepakinta. Trečiojo metodo atveju elementų kiekis padidėja 1,3 kartus, jeigu gaunami visi lentelės duomenys. Taip yra todėl, jog lentelėje pridedami nauji laukai. Išrinkus stulpelius, grąžinamų elementų kiekis nepakinta. Gaunant visus duomenis ketvirtojo metodo užklausoje, grąžinamų elementų kiekis padidėja 1,25. Padidėjimas atsiranda dėl naujai pridėtų laukų. Išrinkus stulpelius, grąžinamų elementų kiekis lieka nepakitęs.

Pirmojo metodo atveju grąžinamų įrašų kiekis nepakinta. Taip yra todėl, jog klausimų lentelė tik prasiplėtė, o užklausose nagrinėjama būtent ši lentelė. Pritaikius antrąjį metodą, užklausoje grąžinama 4,37 kartus daugiau įrašų, jeigu analizuojama klausimų lentelė. Skenuojant atsakymų lentelę, grąžinamų įrašų kiekis nepakinta. Taip yra todėl, jog po metodo pritaikymo naujoje lentelėje prie kiekvieno atsakymo saugomas jo klausimas, todėl klausimo informacija yra kartojama. Trečiojo ir ketvirtojo metodo atveju užklausoje grąžinamų įrašų kiekis nepakinta, kadangi abiem atvejais paveiktos lentelės yra tik praplečiamos.

Pirmojo metodo atveju gaunant visus duomenis iš lentelės, vykdymo laikas sulėtėja 1,12–1,68 kartus. Jeigu išrenkami konkretūs stulpeliai, tuomet vykdoma 1,03–1,31 kartus greičiau. Pritaikius antrąjį metodą, užklausa vykdoma 2,1–3,7 kartus ilgiau (jeigu gaunami visi klausimų lentelės duomenys). Skenuojant visą atsakymų lentelę, užklausa vykdoma 1,2–1,5 kartus ilgiau. Jeigu nurodomi konkretūs klausimų lentelės stulpeliai, tuomet užklausos vykdymo laikas 2,48–3,44 kartus sulėtėja. Nurodžius atsakymų lentelės stulpelius, užklausa vykdoma 1,01–1,5 kartus greičiau. Trečiojo metodo atveju gaunant visus lentelės duomenis, užklausos vykdymo laikas sulėtėja 1,14–1,21 kartus. Jeigu išrenkami lentelės stulpeliai, tuomet užklausa vykdoma 1,01–1,24 kartus ilgiau. Ketvirtojo metodo atveju duomenys gaunami 1,15–1,25 kartus greičiau. Lėčiau gaunama tik užklausa, kai gaunami visi lentelės duomenys.

Metodų palyginimo parametrai pateikti 5.15 pav. Šiuo atveju yra išskaičiuojami scenarijų parametrų vidurkiai. Vientisos spalvos stulpeliai atvaizduojama užklausų vykdymo laikų pokytį. Įstrižomis juostelėmis nubraižyti stulpeliai nurodo grąžinamų įrašų užklausose kiekio pokytį. Horizontaliomis linijomis nubraižyti stulpeliai reiškia užklausose grąžinamų elementų eilutėje kiekio pokytį. Pokyčiai atvaizduojami kartais. Jeigu pokytis toks, jog atitinkamas parametras sumažėja ar pagreitėja, tuomet grafiko stulpelis turi neigiamas reikšmes. Priešingai – jeigu dydis padidėja ar pailgėja, tuomet stulpelis yra teigiamas. Jeigu reikšmė yra lygi vienetui, tuomet atitinkamo parametro dydis nepasikeičia. Prie kiekvieno stulpelio nurodyta pokyčio (kartais) reikšmė.



5.15 pav. Metodų užklausos parametrų pokyčiai

Metodo-1 (pasikartojančių stulpelių pridėjimo) bruožai:

- reikalauja daugiau projektavimo. Dar pradiniame etape reikia numatyti, kiek maksimaliai klausimas gali turėti atsakymų. Vadinasi, reikia žinoti ribotą lentelės stulpelių kiekį;
- reikalauja daugiau programavimo. Kadangi prarandamas lankstumas atsakymų atžvilgiu, užklausose nebeįmanoma gauti konkrečių atsakymų (pavyzdžiui, teisingų). Tam turi būti naudojamas programinis kodas, kuris nagrinėtų gautus įrašus;
- atsiranda tuščių reikšmių. Ne visi klausimai turi būtent 8 atsakymus, todėl nesant atsakymui, prie jo saugoma tuščia reikšmė. Pradinės schemos atveju tokios situacijos nebuvo;
- lentelė yra daug platesnė (lyginant su pradine klausimų lentele);
- klausimas ir atsakymas saugomas viename įraše;
- keičiasi lentelės, kuriose buvo saugoma atsakymo informacija;
- vienintelis metodas, kurio paveiktų lentelių dydis po metodo pritaikymo – sumažėjo.

Metodo-2 (lentelių sujungimo) bruožai:

- panašus į metodą-1. Kaip ir pirmuoju atveju, lentelėje saugomi klausimai ir atsakymai;
- pakeičia sistemos logiką. Lentelės identifikatorius yra atsakymas, todėl pagrindinis komponentas tampa atsakymas, o ne klausimas;
- keičiasi lentelės, kuriose buvo saugoma klausimo informacija;
- klausimas dubliuojamas prie atitinkamo atsakymo;
- metodas labiau pasiteisintų, jeigu būtų kartojama mažiau atributų (būtų naudojama siauresnė lentelė). Šiuo atveju klausimo lentelė turi 9 stulpelius, todėl visi 9 atributai yra kartojami ir lentelė žymiai padidėja;
- metodas, kurio paveiktų lentelių dydis labiausiai padidėjo (beveik dvigubai);
- metodas, kurio užklausos, gaunančios duomenis kartu, vykdymo laiko pokytis buvo mažiausias;
- metodas, kurio užklausos, gaunančios duomenis atskirai, vykdymo laikas labiausiai padidėjo;
- žymiai blogesni rezultatai, negu metodo-1. Pirmiems septyniems scenarijams metodo-1 užklausų vykdymo laikai greitesni 2–4 kartus. Grąžinamų eilučių kiekis metodo-1 atveju visada yra mažesnis 4–5 kartus negu metodo-2. Tačiau elementų kiekis eilutėje metodo-2 atveju yra mažesnis.

Metodo-3 (skaičiuojamųjų laukų saugojimo) bruožai:

- nepaveikiamos kitos lentelės;
- skaičiuojami laukai gali nepadengti visų panaudos atvejų. Tam tikrose situacijose gali reikėti konkretesnių matematinių išraiškų, o lentelėje saugoma bendra statistika;
- skaičiuojami laukai turi būti atnaujinami, kai keičiasi baziniai duomenys. Tai lemia, kad turi būti vykdomas stebėjimas, kada duomenys keičiasi ir vykdomos tam tikros procedūros. Be to, sudėtingėja modifikavimo operacijos. Galiausiai, skaičiavimai gali trukti ilgai;
- metodas, kurio užklausos, gaunančios skaičiuojamuosius duomenis, vykdymo laikas labiausiai sumažėjo.

Metodo-4 (pasikartojančių laukų saugojimas) bruožai:

- nepaveikiamos kitos lentelės;
- atsiranda duomenų dublikatai. Tie patys duomenys saugomi dviejose susijusiose lentelėse. Tai lemia, kad juos reikia modifikuoti abiejose lentelėse. Dėlto gali atsirasti nevientisi duomenys bei anomalijos;
- vienintelis metodas, kurio užklausos, gaunančios duomenis atskirai, vykdymo laikas sumažėjo.

6. IŠVADOS

1. Išanalizavus literatūros šaltinius pastebėta, kad duomenų bazės greitaveikos optimizaciją galima išskaidyti į tris pagrindines technikas: fizinę, modelio ir *SQL* kodo. Fizinė ir *SQL* kodo optimizacijos yra gana standartizuotos ir išanalizuotos, tuo tarpu modelio optimizacija nėra iki galo išnagrinėta. Denormalizacija yra viena iš galimų modelio optimizacijos technikų. Ji yra plačiai naudojama ir dažnai rekomenduojama kaip greitaveikos gerinimo strategija. Analizuojant galimus denormalizacijos metodus šaltiniuose, pastebėta, kad dažniausiai yra naudojamas lentelių sujungimas, lentelių (horizontalus ir vertikalus) skaidymas, pasikartojančios informacijos pridėjimas, skaičiuojamųjų laukų saugojimas ir pasikartojančių stulpelių saugojimas lentelėje. Atskirų denormalizacijos metodų įtaka nėra plačiai ištyrinėta, todėl buvo nuspręsta išsiaiškinti denormalizacijos įtaką duomenų bazės greitaveikai.

2. Tyrimo metu išsiaiškinta, kad po denormalizacijos dėl atsiradusių duomenų dublikatų, duomenų bazėje padaugėja duomenų. Dėl pasikartojančios informacijos, duomenys gali pasidaryti nevientisi ir atsirasti duomenų anomalijos. Dėl šios priežasties sudėtingėja ir ilgėja duomenų modifikavimo (įterpimo, šalinimo bei atnaujinimo) operacijos. Tačiau denormalizacija pagreitina duomenų gavimo laiką, kadangi trumpinami duomenų pasiekimo keliai ir užklausose atsisakoma tam tikrų operacijų (pavyzdžiui, skaičiavimo ar grupavimo), sumažinamas skenuojamų lentelių kiekis. Tai lemia, kad užklausose egzistuoja mažiau sujungimų, kurie turi neigiamą įtaką greitaveikai.

3. Darbe buvo atlikti keturi eksperimentai – pasikartojančių stulpelių saugojimas lentelėje, lentelių sujungimas, skaičiuojamųjų laukų bei pasikartojančios informacijos saugojimas lentelėje. Darbe suprojektuotos dvi duomenų bazės: originali ir po optimizacijos. Palyginimas atliktas nagrinėjant parametrus prieš ir po denormalizacijos. Eksperimentuota su skirtingomis duomenų imtimis – maža, vidutine ir didele. Vieno eksperimento lyginimas prieš ir po optimizacijos leido nustatyti konkretaus denormalizacijos metodo įtaką duomenų bazei, atrasti laiko ir duomenų kiekio priklausomybę bei išmatuoti įvairius parametrus – vykdymo laikus, grąžinamų duomenų dydį bei užklausų charakteristikas.

4. Pasikartojančių stulpelių saugojimo lentelėje metodas yra geriausias sprendimas duomenų bazės dydžio atžvilgiu. Šiuo atveju dydis beveik nepasikeitė (padidėjo 1,01 kartus) – lentelė praplatėjo, bet sumažėjo lentelių kiekis. Gaunant duomenis kartu, užklausos vykdymo laikas sutrumpėjo 4,45 kartus, o atskirai – 1,32 kartus padidėjo. Dėl atsakymų saugojimo specifikos, šis sprendimas reikalauja daugiausia projektavimo ir programavimo.

Lentelių sujungimo metodas pateikė prasčiausius rezultatus. Metodas sukūrė daug duomenų dublikatų – duomenų padaugėjo 1,98 kartus. Gaunant duomenis kartu, užklausos vykdymo laikas sutrumpėjo mažiausiai (1,62 kartus), o atskirai – padidėjo labiausiai (4,4 kartus). Metodas pakeičia sistemos logiką, nes svarbiausias objektas tampa atsakymas, o ne klausimas.

Taikant skaičiuojamųjų laukų saugojimo lentelėje metodą duomenų kiekis padidėjo 1,3 kartus. Šis metodas pateikė geriausią užklausos vykdymo laiką – vykdymas sutrumpėjo vidutiniškai 14,57 kartų. Užklausoje nenaudojant skaičiuojamų laukų, užklausos vykdymas sulėtėjo 1,14 kartų. Nors sprendimas geriausias skaitymo užklausos vykdymo atžvilgiu, tačiau jis reikalauja papildomų operacijų skaičiuojamiems duomenims atnaujinti.

Pritaikius pasikartojančios informacijos saugojimo lentelėje metodą, duomenų kiekis padidėjo 1,25 kartus. Gaunant duomenis kartu, užklausos vykdymo laikas sutrumpėjo 1,96 kartus, o atskirai – 1,21 kartus. Šis metodas neutraliausias – mažiausiai paveikiamos kitos operacijos ir nepaveikiamos kitos lentelės.

7. LITERATŪRA

1. *DB-engines. DBMS popularity broken down by database model* [interaktyvus]. [žiūrėta 2017-12-15]. Prieiga per: https://db-engines.com/en/ranking_categories.

2. POWELL, Gavin. *Oracle Performance Tuning for 10gR2*. Digital Press, Burlington, 2009. ISBN 9781555583453.

3. DAM, Sajal, ir Grant FRITCHEY. *SQL server 2008 query performance tuning distilled*. Apress, 2009. ISBN 9781430219026.

4. *ORACLE: A relational database overview* [interaktyvus]. [žiūrėta 2017-01-10]. Prieiga per: https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html.

5. KUTAS, Remigijus. *Duomenų bazės (DB). reliacinės DB* [interaktyvus]. 2003 [žiūrėta 2017-01-15]. Prieiga per: http://gama.vtu.lt/biblioteka/Information_Resources/3.2paskaita_reliac_DB.pps.

6. TEOREY, Toby J. ir kt. *Database design: know it all*. Amsterdam: Elsevier/Morgan Kaufmann, 2008. ISBN 9780123746306.

7. POWELL, Gavin. *Oracle High Performance Tuning For 9I And 10G*. Amsterdam: Elsevier/Digital Press, 2004. ISBN 1555583059.

8. *TECHOPEDIA: relational database (RDB)* [interaktyvus]. [žiūrėta 2017-01-12]. Prieiga per: https://www.techopedia.com/definition/1234/relational-database-rdb.

9. ENGLAND, Ken, ir Gavin POWELL. *Microsoft SQL Server 2005 Performance Optimization and Tuning Handbook*. Amsterdam: Elsevier/Digital Press, 2007. ISBN 9781555583194.

10. SANDERS, G. L. ir Seungkyoon SHIN. *Denormalization effects on performance of RDBMS*. Proceedings of the 34th Annual Hawaii International Conference, 2001.

11. STEPHENS, Ryan, Ron PLEW ir Arie JONES. *Sams teach yourself SQL in 24 hours*. USA: Sams Publishing, 2008. ISBN 9780672330186.

12. MCEWAN, Bennett ir Ray RANKINS. *Database design and performance* [interaktyvus]. etutorials.org, [žiūrėta 2017-10-03]. Prieiga per: http://etutorials.org/SQL/microsoft+sql+server+2000/Part+I+Welcome+to+Microsoft+SQL+Server/.

13. LEE, Heeseok. *Justifying database normalization: A cost/benefit model*. Information processing & management, 1995.

14. *Normalization of database* [interaktyvus]. [žiūrėta 2017-01-16]. Prieiga per: http://www.studytonight.com/dbms/database-normalization.php.

15. ROUSE, Margaret. *Database normalization* [interaktyvus]. 2016 [žiūrėta 2017-01-16]. Prieiga per: http://searchsqlserver.techtarget.com/definition/normalization.

16. *To be or not to be normal: That is the database question* [interaktyvus]. 2002 [žiūrėta 2017-08-31]. Prieiga per: https://www.techrepublic.com/article/to-be-or-not-to-be-normal-that-is-the-database-question/.

17. SCHWARTZ, Baron, Peter ZAITSEV ir Vadim TKACHENKO. *High performance MySQL: Optimization, backups, and replication*. O'Reilly Media, Inc, 2012. ISBN 9781449314286.

18. MITTRA, Sitansu S. *Database performance tuning and optimization: using Oracle*. Springer Science & Business Media, 2002. ISBN 0387953930.

19. MORTEZA, Zaker, Somnuk PHON-AMNUAISUK ir Su-Cheng HAW. *Optimizing the data warehouse design by hierarchical denormalizing*. World Scientific and Engineering Academy and Society, 2008. ISBN 9789604740284.

20. DRKUŠIĆ, Emil. *Denormalization: When, why, and how* [interaktyvus]. 2016 [žiūrėta 2017-09-03]. Prieiga per: http://www.vertabelo.com/blog/technical-articles/denormalization-when-why-and-how.

21. POWELL, Gavin. *Beginning database design*. Wiley Publishing, 2006. ISBN 0764574906.

22. GUNDERLOY, Mike, Joseph L. JORDEN ir David W. TSCHANZ. *Mastering Microsoft SQL Server 2005*. John Wiley & Sons, 2006.

23. PINTO, Yma. *A framework for systematic database denormalization*. Global Journal of Computer Science and Technology, 2009.

24. SHIN, Seung K. ir G. L. SANDERS. *Denormalization strategies for data retrieval from data warehouses*. Decision Support Systems, 2006.

25. HARRINGTON, Jan L. *Relational Database Design and Implementation : Clearly Explained*. Amsterdam: Morgan Kaufmann, 2009. ISBN 9780123747303.

26. BOCK, Douglas B. ir John F. SCHRAGE. *Denormalization guidelines for base and transaction tables*. ACM SIGCSE Bulletin, 2002.

27. MULLINS, Craig S. *Database Administration: The Complete Guide to Practices and Procedures*. Addison-Wesley Professional, 2002. ISBN 0201741296.

28. DZAKOVIC, Miroslav. *Using a non-traditional design to improve VLDB performance or RM-friendly 1NF denormalization* [interaktyvus]. 2002 [žiūrėta 2018-03-21]. Prieiga per: https://pdfs.semanticscholar.org/ebc4/40e878ff5b4c4205abac71497b6762a1380d.pdf.

29. SHASHA, Dennis, ir Philippe BONNET. *Database Tuning : Principles, Experiments, and Troubleshooting Techniques*. San Francisco, Californija: Morgan Kaufmann, 2003. ISBN 9781558607538.

30. *DBConvert help center: Popular database management systems overview* [interaktyvus]. [žiūrėta 2017-12-01]. Prieiga per: https://support.dbconvert.com/hc/en-us/articles/203189021-Popular-Database-Management-Systems-Overview.