VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Baigiamasis bakalauro darbas

# Duomenų bazių indeksavimo strategijų paieška
(A Search for Database Indexing Strategies)

Atliko: 4 kurso, 1 grupės studentas
Laisvydas Skurevičius (parašas)

Darbo vadovas:
Lekt. Irmantas Radavičius (parašas)

Recenzentas:
Lekt. Vytautas Jančauskas (parašas)

Vilnius
2015

# Turinys

# Įvadas

Technologijų tobulėjimą skatina noras viską atlikti greičiau ir paprasčiau. Paprastas ir greitas duomenų pasiekiamumas yra didelė siekiamybė. Tokio tikslo padeda siekti duomenų laikymas duomenų bazėse. Duomenų bazė yra metodiškai sutvarkytas duomenų rinkinys. Duomenys sutvarkyti laikantis duomenų bazės modelio. Vienas iš tokių modelių talpina duomenis į dvimates lenteles sudarytas iš eilučių, kuriose talpinami duomenų įrašai, ir stulpelių skaidančių duomenų įrašus į konkrečius duomenų laukus. Tokio modelio duomenų bazės yra vadinamos reliacinėmis. Šiame darbe nagrinėjamos būtent tokios duomenų bazės. Tokių duomenų bazių duomenys valdomi naudojant struktūrizuotą užklausų kalbą SQL (angl. structured query language), kurios užklausos yra įvykdomos programos vadinamos duomenų bazių valdymo sistema. SQL užklausos leidžia valdyti ir pasiekti norimus duomenis, tačiau kaip greitai tie duomenys pasiekiami jau priklauso nuo daugybės faktorių.

Vienas iš duomenų pasiekiamumo greitį duomenų bazėje įtakojančių faktorių yra indeksai. Indeksas yra duomenų struktūra sauganti nuorodas į duomenų bazės lentelės įrašus. Indeksas sutvarko duomenis tam tikra tvarka pagal indeksuojamą lentelės stulpelį ar stulpelius, tai leidžia kai kurioms užklausoms duomenis surasti greičiau. Kokia duomenų tvarka tvarkomi duomenys priklauso nuo to kokios duomenų struktūros yra indeksas. Indeksai įtakoja užklausos vykdymo planą, kurį parenka užklausos optimizatorius. Užklausos optimizatorius yra duomenų bazės valdymo sistemos funkcija ieškanti greičiausio užklausos vykdymo plano nusakančio kokiu būdu, kokia tvarka ir iš kokių struktūrų bus renkami duomenys.

Kaip ir daug kitų duomenų bazių optimizavimo sprendimų, indeksavimas kartu su privalumais turi ir trūkumus. Indeksai kuria duomenų kopijas kurios užima papildomą vietą ir turi būti atnaujinamos ir pertvarkomos, kai kinta duomenys indeksuotose lentelėse. Dėl šių priežasčių, nors ir kai kurias užklausas indeksai gali gerokai paspartinti, tačiau užklausoms keičiančioms duomenis užkraunamas papildomas indeksų atnaujinimo darbas, dėl ko jos sulėtėja.

Indeksai turi būti kuriami atsižvelgiant į duomenų bazės užklausų kvietimo tendencijas. Užklausų duomenų bazėse gali būti kviečiama labai daug ir įvairių. Viena problema yra surinkti tuos duomenis apie kviečiamas užklausas, kita problema yra parinkti joms tinkamus indeksus ir rasti balansą, žinant tai kad kai kurios užklausos pagreitės, o kai kurios sulėtės. Indeksų rinkinius vadinsime indeksavimo strategijomis. Tinkamų indeksų rinkinių paiešką vadinsime indeksavimo strategijų paieška. Atlikti visą šį procesą rankomis yra sudėtinga, tam gali reikėti atlikti duomenų bazės tendencijų tyrimus, bei daugybę bandymų ir matavimų. Net ir atrinkus indeksavimo

strategiją rankomis, kyla klausimas ar tai yra geriausias galimas variantas duotai duomenų bazei. Iš šių problemų naturaliai išplaukia poreikis automatizuoti šį procesą.

Indeksavimo proceso automatizavimas yra problema kurią būtent sprendžia šis darbas siekiant taupyti ne tik duomenų bazės vartotojų laiką greitinant SQL užklausų vykdymą, bet ir duomenų bazės administratorių bandančių užtikrinti geriausią indeksavimo strategiją. Problema jau buvo pradėta spręsti autoriaus kursinio projekto darbe. Buvo aprašytas indeksavimo strategijų paieškos modelis. Pradėtas kurti indeksavimo įrankis. Realizuotas užklausų analizatorius išrenkantis iš SELECT duomenų išrinkimo užklausos sąlygas ir parenkantis užklausai tinkamus indeksus. Bei pasiūlytas vienas algoritmas užklausos optimaliam indeksų rinkiniui parinkti. Tačiau nebuvo išspręstos tokios problemos kaip automatizuotas užklausų rinkinio sudarymas kartu su dažnio parametro išgavimu aprašytu indeksavimo strategijų paieškos modelyje. Indeksavimo strategijų paieškos modelis nebuvo pilnai realizuotas ir patikrintas, pateikti sprendimai tik SELECT užklausoms ir kai užklausų rinkinyje yra tik viena užklausa. Šis darbas tęsia kursinio projekto pradėtus darbus ir sprendžia likusias problemas indeksavimo proceso automatizavime.

**Darbo tikslas:** Automatizuoti duomenų bazių indeksavimo procesą.

Darbo uždaviniai:

1. Pasiūlyti ir realizuoti sprendimą indeksavimo strategijų paieškos įvesties – užklausų rinkinio, konstravimui automatizuoti.
2. Pritaikyti indeksavimo strategijų paiešką užklausų rinkiniams, tame tarpe ir duomenis keičiančioms užklausoms.
3. Pasiūlyti ir realizuoti sprendimą užklausoms vertinti.
4. Realizuoti ir palyginti skirtingus optimalios indeksavimo strategijos paieškos algoritmus.

Darbas susideda iš 6 skyrių. Pirmame skyriuje aprašomas indeksavimo strategijų paieškos modelis. Antrame skyriuje nagrinėjami užklausų rinkinio konstravimo etapai, su jais susijusios problemos ir jų sprendimai. Trečiame skyriuje trumpai aprašomas indeksų rinkinių konstravimo procesas į kurį įeina užklausų analizės ir užklausoms tinkamų indeksų parinkimo etapai. Ketvirtame skyriuje nagrinėjama indeksų ir užklausų vertinimo problematika, pateikiamas sprendimas užklausoms vertinti. Penktame skyriuje nagrinėjami įvairūs optimalios indeksavimo strategijų paieškos algoritmai, pateikiami ir analizuojami algoritmų testavimo rezultatai, algoritmai palyginami tarpusavyje. Paskutiniame šeštame darbo skyriuje aprašomas sukurtas indeksavimo įrankis įvardinant techninius sprendimus, bei įrankio taikymo galimybes.

# 1. Indeksavimo strategijų paieškos modelis

Indeksavimo strategiją arba planą galima apibrėžti kaip indeksų rinkinį konkrečiai duomenų bazei ir užklausų rinkiniui, indeksavimo strategijos paskirtis yra paspartinti užklausų iš duoto rinkinio vykdymą duotoje duomenų bazėje. Indeksavimo strategijų paieška siekia rasti tokius indeksų rinkinius su kuriais konkrečios užklausos būtų vykdomos greičiausiai duotoje duomenų bazėje. Tam, kad būtų galima ieškoti indeksavimo strategijų reikia apibrėžti paieškos modelį, kuriuo remsis paieška.

## 1.1. Užklausų rinkinys

Indeksavimo strategijų paieškos įvestis yra užklausų rinkinys $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$ (angl. query), kur $q_i$ ($i$ = 1, 2, ..., $|Q|$) yra SQL užklausa. Užklausos gali būti duomenų paieškos (SELECT), duomenų atnaujinimo (UPDATE), duomenų įterpimo (INSERT), arba duomenų ištrynimo (DELETE) užklausos [DD87].

Kiekviena užklausa $q_i$ turi savo vykdymo laiką $t(q_i)$, kuris yra kintantis ir priklauso nuo esamos indeksavimo strategijos. Indeksų paskirtis yra minimizuoti užklausos $q_i$ vykdymo laiką. Kadangi indeksai, kai kurias užklausas spartina, o kai kurias lėtina, ieškant indeksavimo strategijos reikia balansuoti indeksų kiekį pagal tai, kiek kokių užklausų kviečiama daugiausiai. Jei duomenys duomenų bazėje yra dažnai įterpiami ar keičiami, norint pagreitinti duomenų bazės darbą reikėtų riboti indeksų kiekį. Todėl strategijų paieškos modelyje kiekviena užklausa $q_i$ turi dažnumo koeficientą $f_i$ (angl. frequency) apibrėžiantį kaip dažnai atitinkama užklausa $q_i$ yra kviečiama.

Be to kaip dažnai kviečiama užklausa reikia atsižvelgti ir į tai koks užklausos prioritetas, tai yra kiek svarbu tai užklausai vykdymo laikas. Kadangi kai kurios užklausos gali būti vykdomos sistemos naudojančios duomenų bazę piko metu, kai duomenų baze naudojasi daug vartotojų ir svarbūs tampa duomenų bazės serverio resursai bei užklausos vykdymo laikas, o kai kurios užklausos gali būti vykdomos naktį, kai vartotojų naudojančių sistemą nedaug, arba išvis nėra, tokiu atveju užklausos vykdymo laikas gali būti nesvarbus. Dėl šios priežasties kiekviena užklausa $q_i$ turi prioritetą $p_i$.

Visos užklausos iš užklausų rinkinio $Q$ turi būti parašytos konkrečiai reliacinei duomenų bazei *DB*. *DB* indeksavimo strategijų paieškoje naudojama matuojant ir lyginant užklausų vykdymo laiką su įvairiais indeksais.

## 1.2. Indeksų rinkiniai

Užklausų analizatorius analizuodamas užklausą $q_i$ iš rinkinio $Q$ sukonstruoja užklausos sąlygų rinkinį $C_i = \{c_1, c_2, \ldots, c_{|C_i|}\}$ (angl. condition), kur $c_j$ ($j$ = 1, 2, ..., $|C_i|$) yra duomenų paieškos (SELECT), duomenų atnaujinimo (UPDATE), arba duomenų ištrynimo (DELETE) užklausos sąlygos. Duomenų įterpimo (INSERT) užklausos sąlygų neturi. Kiekviena sąlyga $c_j$ turi savo tipą *type*, operatorių *operator* ir stulpelių sąrašą $A_j = \{a_1, a_2, \ldots, a_{|A_j|}\}$ (angl. attribute), kur $a_k$ ($k$ = 1, 2, ..., $|A_j|$) yra stulpelis turintis lentelės pavadinimą *table_name* ir stulpelio pavadinimą *column_name*.

Iš sąlygų rinkinio $C_i$ išgaunamas indeksų rinkinys $X_i = \{x_1, x_2, \ldots, x_{|X_i|}\}$, kur $x_l$ ($l$ = 1, 2, …, $|X_i|$) yra indeksas, kuris turi savo tipą *type* ir visą kitą reikalingą informaciją tam kad galėtų būti sukurtas indeksas, kuri gaunama iš sąlygos stulpelių sąrašo $A_j$, tai lentelės pavadinimas ir stulpelių pavadinimų sąrašas. $X_i$ indeksų rinkinyje yra visi indeksai, kurie galėtų paspartinti užklausą $q_i$ duomenų bazėje *DB*. Optimalus indeksų rinkinys užklausai $q_i$ bus žymimas kaip $X_{i\ opt} = \{x_1, x_2, \ldots, x_{|X_{i\ opt}|}\}$. Indeksų rinkinys $X_i$ yra aibė ir jame vienodi indeksai negali kartotis.

## 1.3. Indeksavimo strategija

Galutinis indeksavimo strategijų paieškos rezultatas yra indeksavimo strategija $S = \{x_1, x_2, \ldots, x_{|S|}\}$. Indeksavimo strategija $S$ yra indeksų rinkinys skirtas užklausų rinkiniui $Q$ paspartinti duomenų bazėje *DB*. Indeksavimo strategijoje indeksai kaip ir anksčiau apsibrėžtuose rinkiniuose negali kartotis.

Indeksavimo strategijos gavimo būdas priklauso nuo pasirinkto indeksavimo strategijų paieškos algoritmo. Indeksavimo strategijų gali būti daug ir įvairių, tam kad rasti tinkamiausią esamai situacijai šis paieškos modelis leidžia įvertinti gautą indeksavimo strategiją, tam reikia sukurti visus indeksus iš $S$ duomenų bazėje *DB* ir apskaičiuoti indeksavimo strategijos įvertinimo dydį $e$ užklausoms iš užklausų rinkinio $Q$ taikant formulę: $e = \sum t_{(q_i)} * f_i * p_i$. Kuo įverčio reikšmė mažesnė tuo indeksavimo strategija yra geresnė.

# 2. Užklausų rinkinio konstravimas

Viena iš pirmųjų problemų kurias reikia spręsti indeksavimo strategijų paieškos procese yra užklausų rinkinio konstravimas. Užklausų rinkinys yra indeksavimo strategijų paieškos modelio įvestis ir be jo tolimesni paieškos žingsniai yra negalimi. Apskritai indeksuoti duomenų bazę nežinant kokios užklausos yra kviečiamos galima tiktai aklai spėliojant. Šiame skyriuje aprašomos užklausų rinkinio konstravimo procesas, jame kylančios problemos ir jų sprendimai.

## 2.1. Užklausų registravimas

Autoriaus kursiniame projekte užklausų rinkinio sudarymo problema buvo sprendžiama paprastai, darant prielaidą kad dažniausiai kviečiamos užklausos yra žinomos iš anksto ir yra surašomos ranka bei paduodamos indeksavimo strategijų paieškos procesui. Taip gali būti jei į duomenų bazę kreipiasi aplikacija su tam tikromis fiksuotomis iš anksto žinomomis užklausomis. Tačiau realybėje dažnai kokios užklausos yra kviečiamos nėra akivaizdu. Tam sužinoti reikalinga kaupti statistiką.

Pirmas žingsnis užklausų rinkinio konstravime yra duomenų bazėje kviečiamų užklausų išsaugojimas arba registravimas. Kiekviena kviečiama užklausa turi būti užfiksuojama ir įrašoma arba tam skirtame faile, arba tam skirtoje duomenų bazėje. Dažniausiai užklausų išsaugojimo galimybes siūlo pačios duomenų bazių valdymo sistemos. Per kiek laiko kaupiama užklausų statistika yra pakankama tiksliam užklausų rinkinio konstravimui priklauso nuo duomenų bazės vartotojų aktyvumo ir kviečiamų užklausų įvairovės.

## 2.2. Užklausų filtravimas

Sukaupus pakankamai didelę aibę užklausų reikia atlikti užklausų atranką, nes didelė dalis kviečiamų užklausų yra neprasmingos indeksavimo strategijų paieškoje. SQL kalboje yra užklausų tipų, kurių niekaip neįtakoja indeksai, kaip pavyzdžiui naujų lentelių kūrimo užklausos. Dažniausiai duomenų bazių valdymo sistemos įveda ir savo specifines užklausas nepatenkančias į SQL standartą kaip ir tam tikras komandas, kaip pavyzdžiui PostgreSQL meta komandos. Problema kad dažnai duomenų bazių valdymo sistemos siūlomos užklausų registravimo funkcijos jas visas fiksuoja. Indeksavimo strategijos paieškai įdomios tik SELECT, UPDATE, INSERT ir DELETE užklausos. Tad pirmas žingsnis yra iteruoti per visas užfiksuotas užklausas ir paprasčiausiai palikti tik tas užklausas kurios prasideda išvardintais raktažodžiais.

Atrinkus norimas užklausas pagal raktažodžius dar neišsprendžiamos visos problemos, kadangi užfiksuojamos gali būti ne tik vartotojų kviečiamos užklausos, bet ir pačios duomenų bazių valdymo sistemos kviečiamos sisteminės užklausos, kurios taip pat gali būti SELECT

užklausomis renkančiomis duomenis iš sisteminių lentelių saugančių įvairią informaciją apie duomenų bazę. Vienas šios problemos sprendimo būdas yra išsitraukti iš duomenų bazės informaciją apie sistemines lenteles, gauti jų pavadinimus ir patikrinti kievieną likusią užklausą ar ji dirba būtent su šiomis lentelėmis. Ir jei užklausose aptinkami šių lentelių pavadinimai, užklausos turi būti išmetamos iš užklausų rinkinio. Čia reikia neatmesti ir varianto kad pavadinimai gali būti tiesiog tekstinės reikšmės tarp kabučių, tokiu atveju užklausos turi būti paliktos. Kai kurios sisteminės užklausos nesikreipia į sistemines lenteles, jas taip pat galima atpažinti pagal tai kad jos kviečia sistemines funkcijas tokiu pačiu principu. Užklausų filtravimą galima pradėti jau ir užklausų registravimo metu, pavyzdžiui registruoti tik tas užklausas, kurias kviečia duomenų bazės vartotojai.

### 2.3. Užklausų grupavimas

Atmetus visas indeksavimo strategijai įtakos nedarančias užklausas lieka dar viena problema. Užklausų gali būti labai daug ir kiekvienos užklausos analizės ir matavimų kaina indeksavimo strategijų paieškoje yra didelė. Analizuoti visas užklausas iš eilės po vieną net nebūtų prasminga, kadangi dauguma jų atlieka tas pačias užduotis, turi tas pačias sąlygas ir skiriasi tik jų parametrų reikšmės. Problemos sprendimas turi būti užklausų grupavimas pagal tam tikrus užklausos požymius.

Visų pirma galima sugrupuoti visas sintaksiškai identiškas užklausas, tačiau to nepakanka. Žinant kad užklausų analizatoriaus algoritmui svarbiausi užklausų požymiai indeksų parinkime yra užklausų sąlygos, naturaliai išplaukia sprendimas grupuoti užklausas pagal jų sąlygas. Tą galima padaryti paprasčiausiai ignoruojant visų parametrų reikšmes, vietoj jų visose užklausose įstatant pasirinktą simbolių seką ir tada jau grupuoti visas identiškas užklausas pagal jų sintaksę. Ignoruoti parametrus gali būti netikslu tuo atveju, jei užklausos labai dažnai kviečiamos suo tomis pačiomis parametrų reikšmėmis. Kitaip tariant grupavimas galėtų vykti pagal bet kokius užklausų požymius, priklausomai nuo situacijos. Čia svarbiausia yra kuo labiau sumažinti užklausų skaičių neprarandant indeksavimui svarbių detalių ir tuo pačiu nepaveikiant visos indeksavimo strategijų paieškos tikslumo.

### 2.4. Užklausų dažnio ir prioriteto gavimas

Indeksavimo strategijų paieškos modelyje kiekviena užklausa $q_i$ turi savo dažnio parametrą $f_i$ nusakantį kaip dažnai kviečiama tokia užklausa. Šis parametras įtakoja indeksavimo strategijos įvertį ir tuo pačiu gali įtakoti galutinės indeksavimo strategijos parinkimą. Yra svarbu kad šis parametras būtų tikslus, objektyvus ir gaunamas automatizuotai. Jį užklausai galima nustatyti

visų pirma suskaičiavus kiek kartų ta užklausa buvo kviesta. Skaičiuoti reikia dar užklausų grupavimo metu. Tiek kiek užklausų yra vienoje grupėje tiek kartų kviesta buvo ta užklausa.

Dažnio parametrą palikti kaip skaičių nusakantį kiek kartų buvo kviesta užklausa yra netinkamas sprendimas, nes tokiu atveju šis parametras gali pasiekti neproporcingai dideles reikšmes, o jis naudojamas formulėje $e = \sum t_{(q_i)} * f_i * p_i$ skaičiuojančioje indeksavimo strategijos įvertį. Jei $f$ reikšmė tampa neproporcingai didele, išsikreipia kitų parametrų įtaka įverčiui ir tuo pačiu išsikreipia visa indeksavimo strategijų paieška, o gaunami rezultatai tampa nenuspėjami. Tai galima išspręsti susumavus kiek iš viso kartų buvo kviestos visos užklausos ir kiekvienai užklausai paskaičiavus kokią dalį tos sumos sudaro jų kvietimų skaičiai. Kitaip tariant taikyti tokią formulę $f_i = n_i / \sum n_j$ ($j = 1, 2, ..., |Q|$), kur $n_i$ šiuo atveju yra skaičius kiek kartų užklausa $q_i$ buvo kviesta. Pagal formulę gautas dažnio parametras įgauna realią reikšmę intervale nuo nulio iki vieneto ($0 < f_i \leq 1$).

Kitas užklausos $q_i$ parametras yra prioritetas $p_i$, jis taip pat įeina į indeksavimo strategijos įverčio skaičiavimą. Jo gavimas nėra automatizuojamas. Tai yra vienintelis subjektyvus kriterijus suteikiantis galimybę vertinti užklausas pagal jų svarbą. Tai gali būti bet koks teigiamas realus skaičius.

# 3. Indeksų rinkinio konstravimas

Turint užklausų rinkinį galimas sekantis indeksavimo strategijų paieškos etapas, tai yra užklausoms tinkamų indeksų rinkinių konstravimas. Šis etapas susideda iš užklausų analizės ir indeksų atrankos. Etapui reikalingas indeksavimo strategijų paieškos modelyje minėtas užklausų analizatorius, kuris kaip įvestį paimą vieną užklausą ir kaip išvestį grąžina indeksų, galinčių paspartinti duotą užklausą, rinkinį. Šiame skyriuje aprašomas indeksų rinkinio konstravimo procesas.

## 3.1. Užklausos analizė

Pirmas analizės žingsnis yra nustatyti kokia tai yra užklausa, tai nusako pirmasis užklausos žodis. Jei tai duomenų įterpimo INSERT užklausa tolimesni veiksmai nereikalingi, kadangi tokių užklausų jokie indeksai nespartina. [Win12] Tokias užklausas indeksai lėtina, jei indeksai sukuriami ant lentelės į kurią įterpiami duomenys. UPDATE užklausos turi duomenų paieškos sąlygas, tai reiškia kad jos gali būti spartinamos indeksų, tačiau jos keičia duomenis ir jei keičiami duomenys yra saugomi indekse, užklausa lėtėja. [Win12] DELETE užklausos taip pat turi paieškos sąlygas, kurios gali būti spartinamos indeksų, tačiau teoriškai bet koks indeksas ant lentelės iš kurios trinama turi ir lėtinti užklausą. [Win12]

Jei tai nėra INSERT užklausa, iš užklausos turi būti ištrauktos indeksavimui įtaką darančios sąlygos ir sukuriamas sąlygų rinkinys $C_i$. Kiek detalių įtraukti į sąlygas priklauso nuo užklausų analizatoriaus sudėtingumo. Bet kokiu atveju užklausų analizatorius turi atlikti gramatinį užklausos nagrinėjimą (angl. parsing). Vienas iš gramatinio nagrinėjimo būdų gali būti teksto skaidymas. [SW00] Užklausos tekstas turi būti skaidomas dalimis atskiriant savarankiškas ir neskaidomas dalis į žodžius. Skaidyti galima pagal tarpus, svarbiausia yra atskirti stulpelių pavadinimus kartu su lentelių pavadinimais, operatorius ir svarbiausius raktažodžius.

Toliau galima atlikti sąlygų rinkinio konstravimą iteruojant per atskirtus užklausos žodžius. Iteruojant turi būti ieškoma raktažodžių tokių kaip WHERE, JOIN, GROUP BY, ORDER BY. Šie raktažodžiai nusako sąlygos tipą. Suradus raktažodį toliau seka atitinkami veiksmai priklausomai nuo to koks raktažodis buvo rastas. Rezultate turi būti išrinkti sąlygos atributai, kurie buvo aprašyti indeksavimo strategijų paieškos modelyje, tai sąlygoje naudojamų stulpelių ir jų lentelių pavadinimai, operatoriai ir sąlygos tipas, kas turi būti vienas iš ieškomų raktažodžių. Pats gramatinio nagrinėjimo procesas detaliau buvo aprašytas autoriaus kursinio projekto darbe.

### 3.2. Indeksų parinkimas

Išrinktos užklausų sąlygos leidžia nesunkiai parinkti užklausai tinkamus indeksus. Indeksų tipų yra labai daug ir visi jie turi savo privalumų ir trūkumų. Nuo indekso savybių priklauso kokio tipo užklausai jis yra labiau tinkamas. Užklausų tipus galima išskirti pagal užklausos sąlygas. Plačiau tai aprašyta ir nagrinėjama autoriaus kursiniame darbe. Nors indeksų tipų egzistuoja daugybė, tačiau dauguma šiuolaikinių duomenų bazių valdymo sistemų naudoja tik B+ medžių ir maišos lentelių indeksus. Šie indeksai naudojami dėl keletos svarbių savybių.

B+ medžiai skaido duomenis į šakas, taip duomenų paieškoje ties kiekviena medžio viršūne gali atkristi didelė dalis įrašų. [Com79] Svarbi yra B+ medžių savybė rikiuoti duomenis didėjimo arba mažėjimo tvarka, nes šia savybe gali pasinaudoti tiek paprastos duomenų paieškos, tiek ir duomenų rikiavimo, lentelių jungimo užklausos ir lentelių grupavimo užklausos, kurių vykdymo algoritmai remiasi duomenų išrikiavimu. Tuo tarpu maišos lentelės pasižymi tuo kad duomenų įrašus dėlioja taikant maišos funkciją paskaičiuojančią adresą pagal duotą stulpelio reikšmę. Tokiu būdu išdėliotų duomenų paieškos sudėtingumas yra O(1), kai B+ medžio yra $O(\log_b n)$, kur b yra B+ medžio eilė. [GUW08] [Com79] Tačiau maišos lentelės indekso duomenų įrašai nėra išrikiuoti, dėl to šie indeksai gali spartinti tik užklausas kur ieškoma konkrečių reikšmių, kitaip tariant užklausose naudojančiose lygybės operatorius. Plačiau apie šiuos ir kitus indeksų tipus ir kokioms užklausoms jie labiausiai tinkami spartinti yra aprašyta autoriaus kursiniame darbe.

Renkantis indeksus galima išskirti keturis atvejus, kurie susiveda į du pagrindinius atvejus, esant ribotam indeksų pasirinkimui. Šie atvejai yra WHERE, JOIN, GROUP BY ir ORDER BY sąlygos užklausose. Nors WHERE ir JOIN sąlygos skirtingai įtakoja užklausos vykdymo planą ir indeksai čia panaudojami skirtingai, tačiau turint B+ medžio ir maišos lentelės indeksus šių situacijų indeksų parinkimas rinkiniui yra identiškas. Visada kuriami B+ medžio indeksai ant vieno ar dviejų sąlygos stulpelių, o jei sąlyga naudoja lygybės operatorių pridedami ir maišos lentelės indeksai. GROUP BY ir ORDER BY sąlygos taip pat nors indeksus naudoja skirtingai, tačiau susiveda į identišką indeksų rinkinio konstravimo atvejį. Visada kuriamas vienas B+ medžio indeksas ant visų sąlygos stulpelių iš karto. Šis procesas detaliau aprašomas autoriaus kursinio projekto darbe.

# 4. Užklausų ir indeksų vertinimo metodika

Indeksavimo strategijų paieškoje būtina turėti indeksų vertinimo metodiką, tam kad indeksus būtų galima palyginti tarpusavyje. Kadangi paieškos tikslas yra spartinti indeksais užklausas, tai indeksų vertinimas turi vykti per užklausų vydymo laiko vertinimą. Tai reiškia kad indeksams vertinti reikalinga užklausų vertinimo metodika, kuri ir aprašoma šiame skyriuje.

## 4.1. Parametrų generavimas

Kaip buvo aprašyta darbo 2.3 skyrelyje, užklausos turi būti grupuojamos pagal tam tikrus požymius, vienas iš sprendimų yra grupuoti užklausas ignoruojant parametrų reikšmes. Užklausų grupavimas sukelia problemų vertinant užklausą, kadangi užklausa tampa abstrakti ir jos realus vykdymas tampa neįmanomas, o vertinimas įmanoma taip pat tik abstraktus. Iš kitos pusės, jei užklausa turėtų konkrečias parametrų reikšmes, užklausos vertinimas būtų tikslus tik tai užklausai. Tai reiškia kad būtų aprėpta labai siaura užklausų imtis. Užklausų parametrų reikšmės įtakoja užklausos vykdymo laiką, o dažniausiai jos stipriai varijuoja. Vertinant užklausas reikia siekti vienu vertinimu aprėpti kuo didesnę užklausų imtį. Todėl užklausų grupavimas yra pirmasis užklausų vertinimo proceso žingsnis. Antrasis žingsnis yra parametrų generavimas, kurio rezultatas turi būti kiekvienai užklausai $q_i$ iš užklausų rinkinio $Q$ gauti generuotų parametrų užklausų rinkiniai $G_i = \{g_{i1}, g_{i2}, \ldots, g_{i|G_i|}\}$, kur kiekvienas $g_{im}$ ($m = 1, 2, ..., |G_i|$) yra užklausos $q_i$ varijacija su skirtingomis parametrų reikšmėmis.

Parametrų generavimo sudėtingumas skiriasi priklausomai nuo to kokia tai yra užklausa. Paprasčiausia yra generuoti parametrus SELECT ir DELETE užklausoms, nes generuoti duomenys ten neprivalo būti korektiški, kadangi abiejų tipų užklausose parametrai būna tik paieškos WHERE sąlygose. UPDATE užklausos tap pat turi parametrus WHERE sąlygose, tačiau turi ir parametrus įrašo reikšmėms keisti, tai reiškia kad generuoti duomenys negali pažeisti duomenų bazės lentelių apribojimų (angl. constraints), bei turi atitikti duomenų tipo formatą. Ta pat problema yra ir su INSERT užklausomis.

Parametrų reikšmes galima generuoti pseudo atsitiktinai, parenkant atsitiktines simbolių sekas laikantis tam tikro formato. Kad generuoti duomenys būtų korektiški, reikalinga išsitraukti iš duomenų bazės tikslius stulpelių, kuriems generuojami parametrai, duomenų tipus ir žinoti jų formatus. Paprasčiausių duomenų tipų, kaip tekstinių ir skaitinių duomenų generavimas nėra sudėtingas. Tačiau egzistuoja ir sudėtingesnių duomenų tipų, pavyzdžiui data, kuri turi daug skirtingų formatų. Tokiu atveju generuoti duomenis darosi sudėtinga, dėl to kad gali įvykti formato neatitikimai, užklausa neveiks ir tuo pačiu negalės būti įvertinta. Atsitiktinai generuojant duomenis problemų gali sukelti ir duomenų bazės lentelių apribojimai (angl. constraints).

Pavyzdžiui, gali būti išorinio rakto apribojimas, kuris reikalauja kad reikšmė sutaptų su kitos lentelės stulpelio reikšme. Taip pat visiškai atsitiktinės reikšmės dažniausiai nėra realūs duomenys ir nebus randami lentelėse vykdant SELECT, UPDATE ir DELETE užklausas. Norint užtikrinti tikslesnį užklausos vertinimą reikia simuliuoti kuo realesnę duomenų bazės vartojimo situaciją. Atsitiktinių neegzistuojančių duomenų paieška gali paveikti vertinamų užklausų vykdymo laiką ir neatspindėti realios situacijos.

Alternatyva visiškai atsitiktiniam duomenų generavimui yra realių lentelės duomenų panaudojimas parametrų reikšmėms. Parametrus galima generuoti pasitelkiant duomenų paiešką. [McM04] Sprendimas yra išsitraukti kiekvieno stulpelio, kuriam generuojami parametrai, realius duomenis iš duomenų bazės, atsitiktinai imti stulpelių reikšmes ir įstatyti į užklausos parametrų vietas. Tokiu būdu nereikia rūpintis dėl duomenų tipo formato, užtenka žinoti ar stulpelis yra tekstinio duomenų tipo, tokiu atveju parametro reikšmė turi būti tarp kabučių. Šis parametrų generavimo būdas išsprendžia neegzistuojančių duomenų problemą, todėl geriausiai tinka SELECT, UPDATE ir DELETE užklausoms. Tačiau šis būdas nesprendžia lentelės apribojimų problemos, todėl gali sukelti problemų generuojant INSERT užklausų parametrus.

Galimas ir kitas būdas, kuris išvengia duomenų generavimo ir jo sukeliamų problemų. Užklausų grupavimo metu reikia kaupti visas kviestas užklausas su dar nepanaikintais parametrais kiekvienai užklausų grupei. Tuomet užklausas vertinti galima testuojant realias vartotojų kviestas užklausas atsitiktinai imant jas iš grupavimo metu sukauptų užklausų rinkinių. Tokiu atveju ar kviečiamos užklausos bus su reikalavimus atitinkančiais parametrais priklausys jau nuo duomenų bazės vartotojų.

### 4.2. Vykdymo laiko matavimas

Užklausos įvertinimas indeksavimo strategijų paieškoje yra jos vykdymo laikas. Matuojant užklausos vykdymo laiką yra svarbu minimizuoti matavimo paklaidas, tam reikalingi daugkartiniai užklausos vykdymo bandymai. Autoriaus kursinio projekto darbe buvo nustatyta, kad mažiausia paklaida gaunama imant minimalų užfiksuotą užklausos vykdymo laiką per daug bandymų, šis būdas buvo lygintas su vidurkio skaičiavimu ir vidurkio skaičiavimu atmetus dalį didžiausių reikšmių. Tačiau testuojama buvo daug kartų kviečiant tą pačią užklausą ir nesikeičiant jos parametrams. Šiuo atveju reikia apskaičiuoti viso užklausų rinkinio $G_i$ vykdymo laiką. Keičiantis parametrų reikšmėms keičiasi ir užklausos vykdymo greitis, tai reiškia kad imti minimalią reikšmę visai užklausų grupei yra nekorektiška, nes tai atspindės tik vienos konkrečios užklausos vertinimą.

Sprendimas yra imti grupės užklausų vykdymo laiko vidurkį. [BHH78] Tačiau žinant kad vidurkio skaičiavimas išlaiko didesnes paklaidas nei minimalios reikšmės skaičiavimas, į vykdymo laiko skaičiavimą reikia apjungti abu būdus. Tai yra skaičiuoti visų generuotų užklausų vykdymo laiko matavimų minimumų vidurkį. Tarkime $T_m = \{t_{m1}, t_{m2}, \ldots, t_{m|T_m|}\}$ yra generuotos užklausos $g_{im}$ atlikti vykdymo laiko matavimai, kur $t_{mz}$ ($z = 1, 2, \ldots, |T_m|$) yra $z$-tąjį kartą atlikto matavimo gautas užklausos $g_{im}$ vykdymo laikas. Tuomet užklausos vykdymo laikas turi būti skaičiuojamas naudojant tokią formulę: $t(q_i) = (min(t_{11}, t_{12}, \ldots, t_{1|T_m|}) + min(t_{21}, t_{22}, \ldots, t_{2|T_m|}) + \ldots + min(t_{|G_i|1}, t_{|G_i|2}, \ldots, t_{|G_i||T_m|}))/|G_i|$. Čia *min* yra funkcija grąžinanti mažiausią reikšmę iš duotos aibės.

Matuojant užklausos vykdymo laiką svarbu užtikrinti, kad užklausos būtų testuojamos esant pradinei duotos duomenų bazės būsenai. INSERT, DELETE ir UPDATE užklausos keičia pradinius duomenis, kas įtakoja pačių užklausų vykdymo laiką. Tai galima išspręsti matavimus vykdant transakcijoje, po kiekvienos užklausos vykdant duomenų bazės būsenos atstatymą (angl. rollback). Tačiau net ir tai gali neužtikrinti pradinės duomenų bazės būsenos išlaikymo, kadangi kai kurios duomenų bazių valdymo sistemos vykdant UPDATE ir DELETE užklausas fiziškai palieka pakeistus įrašus lentelėje, nors jie ir nėra viešai matomi. Tokiu atveju reikia užtikrinti, kad duomenų bazė būtų išvaloma nuo užsilikusių įrašų po kiekvienos duomenis keičiančios užklausos.

Testuojant užklausas su generuotais parametrais gali įvykti lentelės apribojimų pažeidimai neleidžiantys įvykdyti užklausos ir tuo pačiu pamatuoti jos vykdymo laiką. Lentelių apribojimų problemą galima spręsti laikinai juos išjungiant tol kol vykdomas užklausų testavimas, tam reikia iš duomenų bazės išsitraukti kokie yra apribojimai ir juos išmesti. Įvykdžius matavimus, apribojimai turi būti vėl sukurti, nes indeksavimo procesas turi duomenų bazę palikti tokią kokia ji buvo prieš pradedant indeksavimo procesą.

# 5. Optimalios indeksavimo strategijos paieška

Indeksavimo strategijų vienam užklausų rinkiniui gali būti daug, tačiau tikslas yra rasti optimalią strategiją. Optimalia indeksavimo strategija duotam užklausų rinkiniui laikome tą strategiją, kurios įvertis *e* yra mažiausias įmanomas užklausų rinkiniui *Q* ir indeksų skaičius tam įverčiui pasiekti yra minimalus. Optimalios indeksavimo strategijos paieškos procesas pradedamas jau išanalizavus visas užklausas ir kiekvienai užklausai jau turint jai tinkamų indeksų rinkinius $X_i$. Šiame skyriuje pateikiami keturi paieškos algoritmai, jie realizuojami, ištestuojami ir palyginami.

## 5.1. Paieškos algoritmai

Optimalios indeksavimo strategijos paieškos algoritmai galimi įvairūs, vienintelis reikalavimas jiems yra tai, kad jų įvestis turi būti užklausų rinkinys *Q* ir užklausoms atitinkami indeksų rinkiniai $X_i$ bei duomenų bazė *DB*, o išvestis yra indeksavimo strategija *S*. Šių algoritmų tikslas yra minimizuoti įvertį *e*, kuris skirtas vertinti indeksavimo strategijoms, tačiau jį galima paskaičiuoti ir su tuščia indeksavimo strategija, tai yra galima apskaičiuoti pradinės duomenų bazės *DB* būsenos įvertį be indeksų, kas gali būti atskaitos taškas pagal kurį gali būti įvertinta strategijos nauda duomenų bazei. Visi toliau pateikiami algoritmai naudoja parametrą $\lambda$, tai realus skaičius tarp 0 ir 1 ($0 < \lambda < 1$), kuris nurodo kokia dalimi indeksas privalo pagerinti užklausos vykdymo laiką, arba indeksavimo strategiją, tam kad būtų įtrauktas į indeksavimo strategiją. Šis parametras padeda kontroliuoti paklaidas, bei leidžia reguliuoti kokiu tikslumu turi būti sudarinėjama indeksavimo strategija.

### 5.1.1. Atskirų užklausų indeksų atimties algoritmas

Atskirų užklausų indeksų atimties alogritmas (toliau bus vadinamas AUIA algoritmu) remiasi optimalaus indeksų rinkinio gavimo algoritmu aprašytu ir realizuotu autoriaus kursinio projekto darbe. Čia jis pritaikytas užklausų rinkiniui ir konstruoja indeksavimo strategiją. Algoritmo idėja yra rinkti indeksus atskirai kiekvienai užklausai nepriklausomai nuo kitų užklausų. Užklausai $q_i$ sukuriami visi $X_i$ indeksų rinkinio indeksai duomenų bazėje *DB* ir pamatuojamas užklausos vykdymo laikas $t(q_i)$, kuris tampa einamuoju laiku. Toliau indeksai po vieną išmetinėjami iš duomenų bazės, po kiekvieno išmetimo matuojant užklausos vykdymo laiką. Jei vykdymo laikas padidėja lyginant su einamuoju laiku, indeksas vėl sukuriamas duomenų bazėje, naujai gautas vykdymo laikas tampa einamuoju laiku ir indeksas dedamas į indeksavimo strategiją *S*, nes tai reiškia kad indeksas spartina užklausą.

Algoritmas daro prielaidą kad duomenų bazės užklausų optimizatorius parinks geriausią užklausos vykdymo planą parinkdamas tinkamiausius indeksus. Taip gali būti ne visada, todėl tai yra vienas algoritmo trūkumų. [Cha98] Taip pat šis algoritmas neatsižvelgia į indeksų neigiamą poveikį, vertina tik pagal tai kiek jie pagreitina konkrečias užklausas. Algoritmo sudėtingumas $O(|X|)$**,** čia $|X| = \sum |X_i|$ yra visų indeksų skaičius iš visų užklausų, nes kiekvienas indeksas bus vertinamas po vieną kartą su viena užklausa. Algoritmo pseudokodas pateiktas 1 paveikslėlyje.

```
create new indexing strategy S
for each query q ∈ Q do
    if  X_q is not empty then
        for each index x ∈ X_q do create index x on database DB
        calculate query q runtime t(q) on database DB
        current_runtime := t(q)
        for each index x ∈ X_q do
            drop index x from database DB
            calculate query q runtime t(q) on database DB
            new_runtime := t(q)
            if λ*new_runtime > current_runtime then
                add index x to S
                create index x on database DB
            else current_runtime := new_runtime
        for each index x ∈ S do drop index x from database DB
return S
```

**Pav. 1. Atskirų užklausų indeksų atimties algoritmo pseudokodas**

### 5.1.2. Indeksų reitingavimo ir pridėties algoritmas

Indeksų reitingavimo ir pridėties algoritmas (toliau bus vadinamas IRIP algoritmu) bando spręsti AUIA algoritmo problemas. Indeksai vertinami atsižvelgiant į visą užklausų rinkinį, taip įvertinant ir jų galimai neigiamą poveikį. Iš pradžių kiekvienas indeksas iš visų užklausų indeksų rinkinių įvertinamas su visomis užklausomis naudojant įvertį *e*. Tokiu būdu indeksai sureitinguojami ir išrikiuojami nuo geriausiai įvertinto iki blogiausiai įvertinto. Įvertis *e* paskaičiuojamas be indeksų. Tuomet indeksai nuo geriausiai įvertinto iki blogiausiai įvertinto yra kuriami po vieną ir skaičiuojamas indeksavimo strategijos įvertis *e* lyginamas su einamuoju įverčiu. Jei įvertis pagerėja, indeksas įtraukiamas į indeksavimo strategiją. Jei įvertis pablogėja ar lieka toks pats indeksas ištrinamas iš duomenų bazės.

Indeksų reitingavimas pagrįstas tuo kad užklausų optimizatorius ne visada parenka geriausią vykdymo planą, tuo pačiu ne visada parenkamas ir geriausiai tinkantis indeksas. Tuo tikslu indeksų reitingavimo sistema padeda parinkti geriausią indeksą nepasitikint užklausų optimizatoriaus sprendimu. Vienas iš algoritmo trūkumų yra, kad jo principas pridėti indeksus po vieną gali nepatikrinti atvejų kai tam tikros indeksų kombinacijos veikia greičiau nei bet kuris

vienas indeksas iš tos kombinacijos atskirai. Algoritmo sudėtingumas yra O(|*X*||*Q*|), čia |*X*| visų indeksų skaičius iš visų užklausų, o |*Q*| yra visų užklausų skaičius, nes kiekvienas indeksas bus vertinamas su kiekviena užklausa, tai bus padaryta du kartus, kadangi indeksai iš pradžių reitinguojami. Taip pat atliekamas indeksų rikiavimas, galima laikyti kad tai yra greitojo rikiavimo algoritmas, kurio sudėtingumas O(|*X*|log(|*X*|)), tuomet jis neįtakoja bendro algoritmo sudėtingumo, kitokiais atvejais rikiavimas gali ir padidinti algoritmo sudėtingumą. Algoritmo pseudokodas pateiktas 2 paveikslėlyje.

```
create new indexing strategy S
for each query q ∈ Q do
        for each index x ∈ X_q do
                add  index x to S
                calculate evaluation e with all queries from Q and index x
                set index x evaluation as evaluation e
sort S by index evaluation
calculate evaluation e with all queries from Q in current DB
current_evaluation := e
for each index x ∈ S do
        create index x on database DB
        calculate current evaluation e with all queries from Q in current DB
        new_evaluation := e
        if new_evaluation < λ*current_evaluation then
                current_evaluation := new_evaluation
                else drop index x from database DB and remove index x from S
for each index x ∈ S do drop index x from database DB
return S
```

**Pav. 2. Indeksų reitingavimo ir pridėties algoritmo pseudokodas**

### 5.1.3. Užklausų rinkinio indeksų atimties algoritmas

Užklausų rinkinio indeksų atimties (toliau bus vadinimas URIA algoritmu) algoritmas apjungia kai kurias abiejų ankstesnių algoritmų idėjas. Duomenų bazėje sukuriami visi visų užklausų indeksai ir pamatuojamas indeksavimo strategijos įvertis, kuris tampa einamuoju. Indeksai ištrinami iš duomenų bazės po vieną, po kiekvieno ištrinimo skaičiuojant įvertį *e* ir lyginant jį su einamuoju įverčiu. Jei įvertis pablogėja, indeksas vėl sukuriamas duomenų bazėje ir patenka į indeksavimo strategiją. Algoritmo sudėtingumas O(|*X*||*Q*|), čia |*X*| visų indeksų skaičius iš visų užklausų|, o |*Q*| yra visų užklausų skaičius, nes kiekvienas indeksas bus vertinamas su kiekviena užklausa. Algoritmo pseudokodas pateiktas 3 paveikslėlyje.

```
create new indexing strategy S
for each query q ∈ Q do
        for each index x ∈ X_q do
                add  index x to S
                create index x on database DB
calculate evaluation e with all queries from Q
current_evaluation := e
for each index x ∈ S do
        drop index x from database DB
        calculate evaluation e with all queries from Q
        new_evaluation := e
        if λ*new_evaluation > current_evaluation then
                create index x on database DB
                else    current_evaluation := new_evaluation
                        remove index x from S
for each index x ∈ S do drop index x from database DB
return S
```

**Pav. 3. Užklausų rinkinio indeksų atimties algoritmo pseudokodas**

URIA algoritmas įvertina ir galimas indeksų kombinacijas taikydamas indeksų atimties principą, atsižvelgia į indeksų negiamą poveikį vertindamas indeksus pagal indeksavimo strategijos įvertį, tačiau remiasi užklausų optimizatoriaus užklausos vykdymo planų parinkimo sprendimais.

### 5.1.4. Dviejų etapų indeksų atimties algoritmas

Dviejų etapų indeksų atimties (toliau bus vadinamas DEIA algoritmu) algoritmas yra AUIA ir URIA algoritmų junginys. Jis apjungia abiejų algoritmų gerąsias savybes, bei išsprendžia AUIA algoritmo problemą, kad indeksai parenkami neatsižvelgiant į visas užklausas ir indeksavimo strategijos įvertį, taip neįvertinant indeksų neigiamo poveikio. Ši problema sprendžiama AUIA algoritmo rezultatui, tai yra gautai indeksavimo strategijai $S$ taikant URIA algoritmą. Tai yra po vieną išmetant indeksus, kurie nepagerina indeksavimo strategijos įverčio. Algoritmo sudėtingumas blogiausiu atveju $O(|X||Q|)$, o geriausiu atveju $O(|X|)$, kur $|X|$ yra visų indeksų skaičius iš visų užklausų, o $|Q|$ visų užklausų skaičius. Algoritmo pseudokodas nepateikiamas, nes iš esmės tai yra AUIA ir URIA algoritmų junginys. Skirtumas yra toks, kad nereikia ištrinti indeksavimo strategijoje esančių indeksų iš duomenų bazės po AUIA algoritmo vykdymo ir nereikia jų sukurti URIA algoritmo pradžioje, tai yra atsiranda tęstinumas tarp dviejų algoritmų.

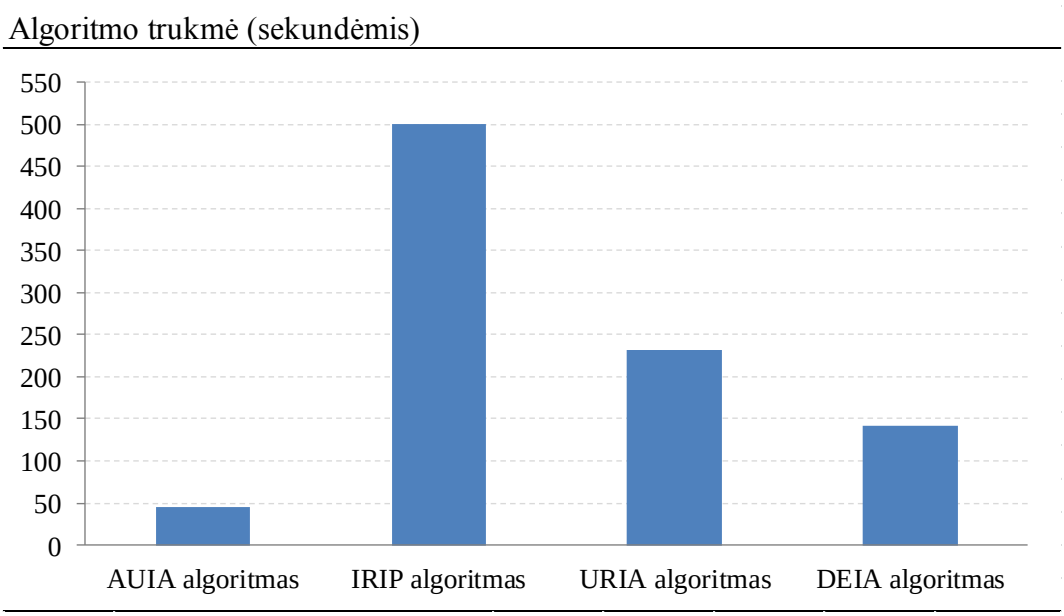
## 5.2. Paieškos algoritmų testavimas ir palyginimas

Įsitikinti algoritmų veiksmingumu ir juos palyginti tarpusavyje galima tik juos išbandžius praktiškai. Algoritmai testuojami su maža pavyzdine pasaulio valstybių, miestų ir kalbų

duomenų baze, sudarytą iš 3 lentelių. Ją sudaro 4079 miestai, 239 valstybės ir 984 kalbos. Užklausų rinkinys, su kuriuo buvo testuojami algoritmai, yra pavaizduotas 4 paveikslėlyje. Ne visos užklausos prasmingos, atsitiktinai parinktos testavimo tikslams. Pirmas skaičius po kabliataškio prie kiekvienos užklausos nurodo kiek kartų užklausa buvo kviesta, antras skaičius nurodo užklausos prioriteto parametrą. Algoritmai buvo testuojami, kai $\lambda = 0.95$.

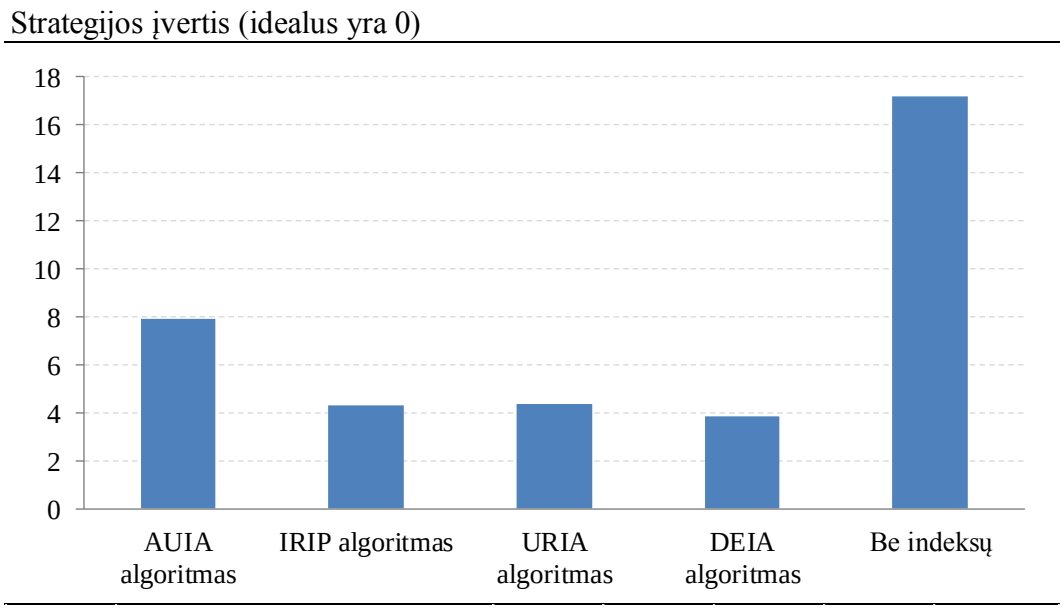
1. SELECT avg(population), name FROM city WHERE city.population>{} or city.name={} or city.district={} GROUP BY city.population, city.name ORDER BY city.population;50;1
2. UPDATE city set name={} WHERE city.population>{} AND city.population<{};250;5
3. SELECT language FROM countrylanguage WHERE countrylanguage.countrycode={};200;1
4. INSERT into countrylanguage (countrycode, language, isofficial, percentage) values ({}, {}, {}, {});200;1
5. SELECT city.name, country.name, countrylanguage.language FROM city full join country on city.population=country.population left join countrylanguage on country.lifeexpectancy<countrylanguage.percentage;100;1
6. DELETE FROM countrylanguage WHERE countrylanguage.language={};90;1
7. SELECT * FROM countrylanguage WHERE countrylanguage.countrycode={} ORDER BY countrylanguage.countrycode desc, countrylanguage.language desc;300;1
8. SELECT * FROM countrylanguage ORDER BY countrylanguage.countrycode, countrylanguage.language;450;1
9. DELETE FROM city WHERE city.population<{};250;5
10. SELECT * FROM countrylanguage WHERE countrylanguage.language={} ORDER BY countrylanguage.countrycode asc, countrylanguage.language asc;400;1
11. SELECT * FROM city WHERE city.district={};40;1
12. SELECT city.name, country.name, countrylanguage.language FROM city left join country on city.population=country.population left join countrylanguage on country.lifeexpectancy<countrylanguage.percentage;20;1
13. INSERT into city (id, name, countrycode, district, population) values ({}, {}, {}, {}, {});500;5
14. SELECT * FROM city LEFT JOIN country ON city.population>country.population;50;1
15. INSERT into country (code, name, continent, region, surfacearea, indepyear, population, lifeexpectancy, gnp, gnpold, localname, governmentform, headofstate, capital, code2) values ({}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {});200;1
16. SELECT name FROM country WHERE country.indepyear={} AND country.governmentform={};40;1
17. SELECT gnp FROM country WHERE country.surfacearea<{} AND country.continent={};50;1
18. DELETE FROM countrylanguage WHERE countrylanguage.countrycode={};100;1

**Pav. 4. Testuojamas užklausų rinkinys**



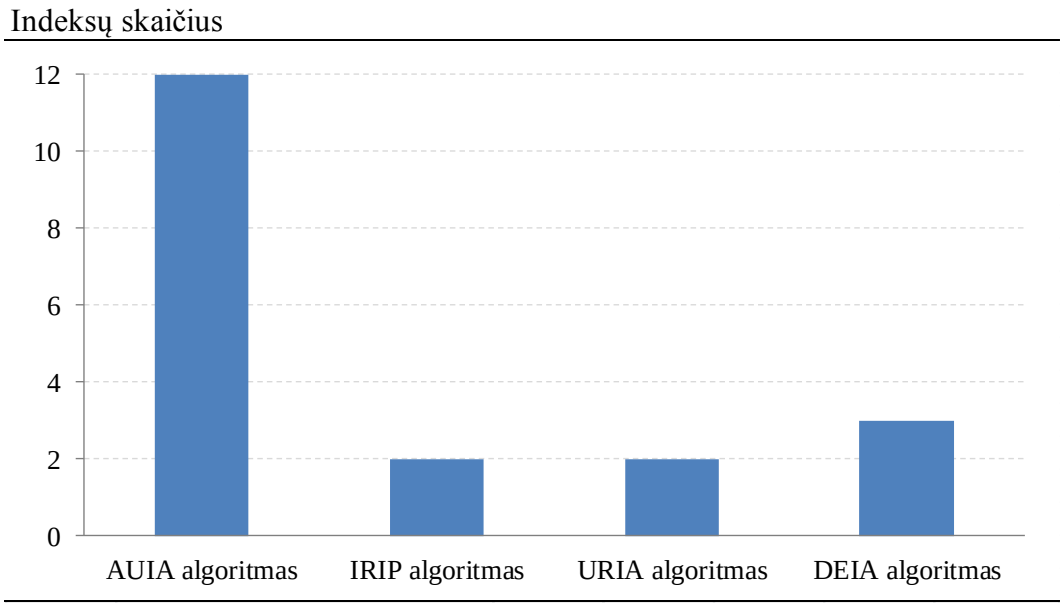

**Pav. 5. Algoritmų vykdymo trukmės grafikas**

Iš grafiko 5 paveikslėlyje matyti, kad AUIA algoritmas veikia greičiausiai. Taip yra todėl, kad AUIA algoritmas vertina indeksus tik su jiem aktualiom užklausom, kai tuo tarpu visi kiti algoritmai indeksus vertina su visom užklausom, galimai ir tom kurių tie indeksai niekaip nepaveikia. Tai reiškia kad kitiems algoritmams kiekvieno indekso įvertinimui, šiuo atveju reikia matuoti 18 užklausų vykdymo laiką. Tuo tarpu pirmam algoritmui reikia matuoti tik 1, nors kartais tas pats indeksas gali būti vertinamas ir kelis kartus, jei jis tinkamas daugiau nei 1 užklausai. IRIP algoritmas veikia gerokai lėčiau už kitus, nes vertina indeksus kitokia tvarka, kuriant ir vertinant indeksus po vieną, o ne iškart sukuriant visus ir atmetant po vieną. Kadangi indeksų paskirtis spartinti užklausas, tai dažniausiai sukūrus visus indeksus užklausų vykdymo laikas išmatuojamas greičiau. Kitaip gali atsitikti jei yra labai daug duomenis keičiančių užklausų, kurias indeksai lėtina. Tuomet IRIP algoritmas veiktų greičiau.




**Pav. 6. Algoritmų gautų strategijų įverčių grafikas**

Iš 6 paveikslėlio grafiko matosi, kad visi algoritmai atlieka savo užduotį ir sumažina indeksavimo strategijos įvertį *e* lyginant su strategija be indeksų. Prasčiausiai tą padaro AUIA algoritmas, dėl to kad renka indeksus spartinančius konkrečias užklausas, tačiau neatsižvelgiant į tai kaip tie indeksai paveikia kitas užklausas. AUIA algoritmas parenka indeksavimo strategiją neatsižvelgiant į indeksavimo strategijos įvertį. Šis trūkumas nėra toks akivaizdus kol nėra daug duomenis keičiančių užklausų. Kiti algoritmai pasiekia geresnius rezultatus, kadangi remiasi visais faktoriais įtakojančiais strategijos įvertį. Geriausią rezultatą pasiekia DEIA algoritmas, kuris išsprendžia AUIA algoritmo strategijos įverčio nepaisymo problemą atmesdamas indeksus didinančius įvertį.




**Pav. 7. Algoritmų gautų strategijų indeksų skaičiaus grafikas**

Užklausų analizatorius 4 paveikslėlio užklausų rinkiniui surado 23 šias užklausas galimai spartinančius unikalius indeksus, jie atvaizduoti 8 paveikslėlyje, jų kūrimo užklausų formatu. Visi testuoti algoritmai didžiosios dalies indeksų neįtraukė į indeksavimo strategiją. 7 paveikslėlio grafike matyti, kad daugiausia indeksų įtraukė AUIA algoritmas, dėl to kad jis įtraukia visus indeksus, kurie spartina bent vieną užklausą, kai visi kiti algoritmai įtraukia tik indeksus, kurie mažina indeksavimo strategijos įvertį. Visi algoritmai įtraukė 7 ir 10 numeriu pažymėtus indeksus. DEIA algoritmas įtraukė 1 indeksu daugiau nei IRIP bei URIA algoritmai, įtraukė ir 6 numeriu pažymėtą indeksą. Įdomu tai, kad kai kurie indeksai atrinkti AUIA algoritmo yra skirti DELETE užklausų spartinimui.

```
1. CREATE INDEX countrylanguage_countrycode_hash ON countrylanguage USING hash (countrycode)
2. CREATE INDEX countrylanguage_language_hash ON countrylanguage USING hash (language)
3. CREATE INDEX country_lifeexpectancy_btree ON country USING btree (lifeexpectancy)
4. CREATE INDEX country_continent_hash ON country USING hash (continent)
5. CREATE INDEX country_continent_btree ON country USING btree (continent)
6. CREATE INDEX countrylanguage_countrycode_language_btree ON countrylanguage USING btree
   (countrycode, language)
7. CREATE INDEX countrylanguage_percentage_btree ON countrylanguage USING btree (percentage)
8. CREATE INDEX countrylanguage_countrycode_btree ON countrylanguage USING btree (countrycode)
9. CREATE INDEX country_governmentform_btree ON country USING btree (governmentform)
10. CREATE INDEX country_population_btree ON country USING btree (population)
11. CREATE INDEX country_indepyear_btree ON country USING btree (indepyear)
12. CREATE INDEX city_population_btree ON city USING btree (population)
13. CREATE INDEX countrylanguage_language_btree ON countrylanguage USING btree (language)
14. CREATE INDEX city_population_name_btree ON city USING btree (population, name)
15. CREATE INDEX city_district_btree ON city USING btree (district)
16. CREATE INDEX city_name_hash ON city USING hash (name)
17. CREATE INDEX city_district_hash ON city USING hash (district)
18. CREATE INDEX country_surfacearea_btree ON country USING btree (surfacearea)
19. CREATE INDEX city_name_btree ON city USING btree (name)
20. CREATE INDEX country_governmentform_hash ON country USING hash (governmentform)
21. CREATE INDEX country_population_hash ON country USING hash (population)
22. CREATE INDEX country_indepyear_hash ON country USING hash (indepyear)
23. CREATE INDEX city_population_hash ON city USING hash (population)
```

**Pav. 8. Užklausų analizatoriaus atrinkti indeksai**

Apibendrinus gautus rezultatus, galima teigti kad DEIA algoritmas pasirodė geriausiai, jis parinko indeksavimo strategiją su geriausiu įverčiu, nors pasiekiamas rezultatas nedaug skiriasi nuo IRIP ir URIA algoritmų, tačiau tas rezultatas paseikiamas per trumpesnį laiką. AUIA algoritmas veikė greičiausiai, tačiau pasiekė prasčiausią indeksavimo strategijos įverčio rezultatą. IRIP algoritmas pasiekė gerus rezultatus, tačiau buvo pats lėčiausias. Matyti, kad indeksų reitingavimas neatnešė naudos. URIA algoritmas taip pat pasiekė gerus rezultatus ir tai padarė per konkurencingą laiką greičiausiems algoritmams.

Galima pastebėti, kad DEIA algoritmas iš pradžių išskaido paieškos užduotį į mažesnes užduotis ir tuomet šių užduočių sprendinius panaudoja palengvinant pagrindinės užduoties išsprendimą. Iš gautų rezultatų matyti, kad toks principas pasiteisino. Taip yra todėl, kad indeksų vertinimas atskiroms užklausoms kainuoja mažiau nei indeksų vertinimas su visu užklausų rinkiniu. Tuo pačiu, indeksų vertinimo metu kiekvienai užklausai atskirai, atkrenta didelė dalis indeksų, dėl to antrame etape, kai indeksai jau vertinami su visomis užklausomis, prireikia daug mažiau vertinimų.

Pirmoje lentelėje pateikiami algoritmų sudėtingumai ir pastebėjimai išryškinantys algoritmų skirtumus vykdymo žingsių atžvilgiu ir tuo pačiu aiškinančius kodėl algoritmų vykdymo laikai skirtingi, nors sudėtingumai asimptotiškai yra vienodi. Sudėtingumai pateikiami blogiausiais atvejais. Čia akcentuojami užklausų vykdymo laiko skaičiavimo žingsniai, kurie yra brangiausi

laiko atžvilgiu. Verta paminėti kad visi algoritmai taip pat atlieka tokias operacijas kaip indeksų sukūrimas ir ištrynimas duomenų bazėje, visi algoritmai jų atlieka ne daugiau nei po $|X|$ kartų.

**1 lentelė. Pastebėjimai dėl algoritmų vykdymo žingsnių**

| Algoritmas | Sudėtingumas | Pastebėjimai |
|---|---|---|
| AUIA | O(\|X\|) | Kiekvienas indeksas vertinamas po vieną kartą ir kiekvienai užklausai (jei turi indeksų) papildomai paskačiuojamas vykdymo laikas po vieną kartą, todėl iš viso atliekama ne daugiau nei $\|X\|+\|Q\|$ užklausų vykdymo laiko skaičiavimo žingsnių. |
| IRIP | O(\|X\|\|Q\|) | Algoritme visi indeksai vertinami po 2 kartus, dėl papildomo indeksų reitingavimo. Taip pat vieną kart įvykdomas strategijos vertinimas be indeksų su visom užklausom. Todėl užklausų vykdymo laikas skaičiuojamas iš viso ne daugiau nei $2*\|X\|*\|Q\|+\|Q\|$ kartų. Taip pat vykdomas indeksų rikiavimas, kurio žingsnių skaičius priklauso nuo rikiavimo algoritmo, prisideda $sort(\|X\|)$ žingsnių, kur *sort* yra funkcija grąžinanti rikiavimo žingsnių skaičių, šių žingsnių kaina nelygi užklausų laiko skaičiavimo žingsnių kainai. |
| URIA | O(\|X\|\|Q\|) | Visi indeksai vertinami po tiek kartų kiek yra užklausų ir taip pat atliekamas vienas pradinis indeksavimo strategijų vertinimas su visomis užklausomis, todėl iš viso atliekama ne daugiau nei $\|X\|*\|Q\|+\|Q\|$ užklausų vykdymo laiko skaičiavimo žingsnių. |
| DEIA | O(\|X\|\|Q\|) | Čia susideda AUIA ir URIA algoritmų žingsniai. Iš viso atliekama ne daugiau nei $\|X\|+ 2*\|Q\| + \|X\|*\|Q\|$ užklausų vykdymo laiko skaičiavimo žingsnių. Tačiau realiai atliekama gerokai mažiau, nes po pirmų $\|X\|+\|Q\|$ žingsnių sumažėja indeksų skaičius. |

# 6. Indeksavimo įrankis

Darbe pateiktų idėjų ir sprendimų visuma pritaikyta praktiškai, to rezultatas yra gautas indeksavimo įrankis išpildantis aprašytą indeksavimo strategijų paieškos modelį, bei padedantis automatizuoti duomenų bazių indeksavimo procesą. Realizuotas užklausų rinkinio konstravimas, užklausų parametrų generavimas, užklausų analizatorius parenkantis užklausoms tinkamus indeksus, bei 4 darbe aprašyti optimalios indeksavimo strategijos paieškos algoritmai. Šis įrankis ir buvo panaudotas testavimuose lyginant optimalios indeksavimo strategijos paieškos algoritmus. Šiame skyriuje aprašomi priimti techniniai sprendimai kuriant įrankį ir įrankio praktinio pritaikymo galimybės.

## 6.1. Techniniai sprendimai

Įrankis suprogramuotas Java programavimo kalba ir pritaikytas PostgreSQL duomenų bazių valdymo sistemai. PostgreSQL pasirinkta dėl to, kad tai yra viena populiariausių nemokamų atviro kodo reliacinių DBVS. Ir taip pat ji turi daugiau skirtingų indeksų tipų lyginant su kitomis panašiomis DBVS, dėl to turi daugiau potencialo indeksavimo strategijų paieškoje. Plačiau apie šios ir kelių kitų DBVS indeksavimo galimybes yra rašoma autoriaus kursiniame darbe.

Indeksavimo įrankio pradinė įvestis yra PostgreSQL sukauptas duomenų bazės kviestų užklausų failas ir prisijungimo duomenys prie analizuojamos duomenų bazės. Visų kviečiamų užklausų saugojimas faile yra nustatomas PostgreSQL serverio konfigūraciniame faile. Užklausų grupavimui ir užklausų kiekio skaičiavimui naudotas išorinis įrankis Enterprise Postgres Query Analyser[1]. Šis įrankis paima kaip įvestį PostgreSQL kaupiamą užklausų failą ir išveda jo analizės rezultatus html formatu. Analizės rezultate yra pateikiamos dažniausiai kviečiamos užklausos su skaičiais kiek kartų jos buvo kviestos. Užklausos sugrupuojamos ignoruojant parametrų reikšmes, vietoj jų įstatant „{}“.

Realizuotas indeksavimo įrankis iš pradžių kviečia išorinį įrankį ir tada išsitraukia iš gauto analizės rezultato dažniausiai kviečiamas užklausas su skaičiais kiek kartų užklausos buvo kviestos. Tada atliekamas užklausų filtravimas, iš pradžių atrenkamos tik SELECT, INSERT, DELETE ir UPDATE užklausos. Toliau iš analizuojamos duomenų bazės ištraukiami visų sisteminių lentelių pavadinimai. Visoms likusioms užklausoms patikrinama ar jos dirba su šiomis lentelėmis. Užklausos turinčios šiuos pavadinimus savo tekste yra išmetamos. Tuomet likusios užklausos patalpinamos į tekstinį failą kartu su skaičiais kiek kartų buvo kviestos ir prioritetais, kurių reikšmės pagal nutylėjimą yra vienetai. Užklausos ir jų parametrai faile atskirti

[1] http://epqa.sourceforge.net/

kabliataškiais. Indeksavimo įrankis suteikia galimybę vartotojui, jei jis nori, koreguoti užklausas ir jų parametrus kartu su prioritetais.

Vartotojui leidus pradedama užklausų analizė. Užklausos yra nagrinėjamos gramatiškai. Kiekviena užklausa išskaidoma dalimis į atskirus žodžius, skaidoma yra pagal tarpus ir kitus skyrybos ženklus. Skyrybos ženklai ignoruojami jei yra tarp kabučių, tai yra viskas kas yra tarp kabučių yra laikoma vienu žodžiu. Išskaidžius užklausas, einama per žodžius ir išrenkamos užklausų sąlygos kaip aprašyta 3.1. skyriuje. Sąlygose taip pat užfiksuojamos parametrų pozicijos užklausų tekste. Iš gautų sąlygų rinkinių konstruojami indeksų rinkiniai kiekvienai užklausai kaip aprašyta 3.2. skyriuje.

Baigus užklausų analizę pradedamas užklausų parametrų generavimo procesas. Tai atliekama einant per gautas užklausų sąlygas. Jei sąlygoje yra išsaugota parametro pozicija, iš sąlygoje esančio stulpelio ištraukiami duomenys naudojant SELECT užklausą. Tuomet atsitiktinai sugeneruojamas sveikas skaičius (ne didesnis nei įrašų kiekis lentelėje), kuris nurodo kelintas įrašas iš lentelės įstatomas kaip parametro reikšmė. Tokiu būdu kiekvienai užklausai sugeneruojama po 10 užklausų su skirtingais parametrais.

Prieš paduodant užklausų rinkinį optimalios indeksavimo strategijos paieškos algoritmui, visos užklausos yra išbandomos duomenų bazėje. Jei vykdant užklausą įvyksta klaida, užklausa yra išmetama iš užklausų rinkinio. Tokiu būdu išfiltruojamos neveikiančios užklausos su klaidomis. Tuomet apskaičiuojami visų likusių užklausų dažnio parametrai kaip parašyta 2.4. skyriuje. Galiausiai užklausos su indeksų rinkiniais paduodamos optimalios indeksavimo strategijos paieškos algoritmams aprašytiems 5 skyriuje.

Užklausų vykdymo laikas skaičiuojamas naudojant PostgreSQL komandą EXPLAIN ANALYZE, grąžinančią informaciją apie duotos užklausos vykdymą, kurioje yra užfiksuota kiek laiko vykdyta užklausa. Vykdymo laikas ištraukiamas iš XML (angl. Extensible Markup Language) formatu gauto komandos rezultato. Jei užklausa yra be parametrų, jos laikas yra matuojamas 10 kartų ir imamas mažiausias gautas rezultatas. Jei užklausa turi parametrus, vykdymo laikas skaičiuojamas pagal formulę aprašytą 4.2. skyriuje. Užklausos vykdomos transakcijos viduje, po kiekvienos užklausos atstatant duomenų bazės būseną į pradinę. Jei vykdoma UPDATE užklausa, po jos kviečiama VACUUM komanda, kadangi PostgreSQL atnaujinant įrašus fiziškai išlieka ir seni įrašai, kurie įtakoja tolimesnius užklausos vykdymo laiko matavimus, o ši komanda juos išvalo. Prieš vykdant kiekvieną užklausą lentelių apribojimų patikrinimai yra atidedami į transakcijos pabaigą, naudojant komandą „SET CONSTRAINTS ALL DEFERRED“.

## 6.2. Įrankio galimybės

Indeksavimo įrankis nors ir veikia su bet kokiu paduotu PostgreSQL serverio užregistruotų užklausų failu, tačiau turi apribojimų užklausoms. SELECT užklausose visi sąlygose naudojami stulpeliai privalo būti rašomi kartu su lentelės pavadinimais atskirti tašku, tai yra formatu <*table_name*>.<*column_name*>. Taip pat nepalaikomos užklausos naudojančios įvairias funkcijas ir sudėtingas išraiškas. Užklausos pažeidžiančios kai kuriuos lentelių apribojimus taip pat gali neveikti, nors lentelių apribojimų tikrinimas atidedamas į transakcijų pabaigą, kai kurie apribojimai yra neatidėliojami (NOT DEFERRABLE) ir tikrinami iškart. Nepalaikomos užklausos tiesiog bus ignoruojamos ir neįtraukiamos į indeksavimo strategijų paiešką. Įrankis testuotas tik su PostgreSQL 9.3 versija, tačiau turėtų veikti ir su kitomis versijomis.

Įrankis indeksavimo strategijas pateikia indeksų kūrimo užklausų pavidalu, o pačią duomenų bazę palieka tokią kokia ji buvo duota, tai yra indeksų nesukuria. Indeksavimo strategijų paieškos metu vykdomi tarpiniai išvedimai, kurie gali padėti nustatyti vertingiausius indeksus. Įrankis taip pat paskaičiuoja įverčius su ir be indeksavimo strategijos, taip leidžiant įvertinti gautos strategijos naudą.

# Rezultatai ir išvados

**Rezultatai:**

1. Pasiūlytas ir realizuotas sprendimas indeksavimo strategijos įvesties – užklausų rinkinio, konstravimui automatizuoti.
2. Indeksavimo strategijų paieška pritaikyta užklausų rinkiniams, tame tarpe ir duomenis keičiančioms užklausoms.
3. Pasiūlytas ir realizuotas sprendimas užklausoms vertinti.
4. Pasiūlyti, realizuoti ir palyginti 4 optimalios indeksavimo strategijos paieškos algoritmai.
5. Sukurtas indeksavimo įrankis įgyvendinantis indeksavimo strategijų paieškos modelį praktikoje.
6. Automatizuotas duomenų bazių indeksavimo procesas.

**Išvados:**

1. Generuojant užklausų parametrų reikšmes optimalu turėti kelis generavimo metodus ir juos pasirinkti dinamiškai priklausomai nuo to kokiai užklausai generuojamos reikšmės, kadangi skiriasi SELECT, INSERT, DELETE, UPDATE užklausų parametrų reikšmių reikalavimų specifika. SELECT ir DELETE užklausų parametrų reišmės neturi tokių griežtų reikalavimų kaip INSERT, bei UPDATE parametrai, todėl tokioms užklausoms neverta naudoti sudėtingesnių ir brangesnių metodų.
2. Vykdant algoritmų testavimus geriausius optimalios indeksavimo strategijos rezultatus pasiekė algoritmas skaidantis paieškos problemą į mažesnes atskiras dalis ir pritaikydamas šių dalių sprendimus sprendžiant pagrindinę problemą. Kitaip tariant optimalios indeksavimo strategijos paiešką efektyviausia pradėti vykdant indeksų vertinimą atskiroms užklausoms, o po to pereinant prie indeksų vertinimo bendrame užklausų rinkinio kontekste.
3. Kiekviena indekso vertinimo operacija yra brangi, nes turi apskaičiuoti užklausų vykdymo laikus. Iš algoritmų testavimo matosi, kad dažnai skaičiuojamas vykdymo laikas užklausų, kurioms vertinamas indeksas nedaro įtakos, todėl optimalios indeksavimo strategijos paieškos algoritmus galima būtų optimizuoti vykdant indeksų vertinimus tik su užklausomis, kurioms indeksai daro įtaką. Tam tik būtų reikalinga papildoma užklausų analizė algoritmų viduje, nustatanti ar užklausa kuo nors susijusi su vertinamu indeksu.
4. Įvertinus gautus testavimo rezultatus matyti kad algoritmai nereitinguojantys indeksų pasiekia panašius rezultatus, kaip ir indeksus reitinguojantis algoritmas ir padaro tai

gerokai greičiau, todėl galima daryti išvadą, kad indeksų reitingavimo procesas optimalios indeksavimo strategijos paieškoje yra brangi operacija ir neatneša pakankamai naudos, kuri kompensuotų jos kainą.

5. Vykdant indeksavimo strategijų paieškos algoritmų testavimus pastebėta, kad DELETE užklausos gali būti paspartintos indeksų, jei duomenų lentelėje nėra daug, tai reiškia kad duomenų paieškos greičio padidinimas gali kompensuoti įrašo iš indekso ištrynimo kainą.
6. Sukurtą indeksavimo įrankį galima taikyti indeksavimo proceso automatizacijai, arba naudoti jo gautus rezultatus kaip rekomendacijas vykdant indeksavimą rankomis. Įrankio galimybės turi apribojimus, tačiau jie netrukdo įrankiui gauti rezultatus.

## Summary

Indexing is one of the methods for database optimization. It helps achieve more effective data retrieval and faster query execution. This thesis investigates the problems of searching of indexing strategies in relational databases; it provides steps of this process with different solutions. An indexing strategy is any set of indexes whose main purpose is to speed up a set of queries in some database. The difficult part of the search is to find the optimal indexing strategy for a query set. The goal of this thesis is to automatize relational database indexing. In this paper a model for searching of database indexing strategies is defined which describes the main components and their place in the search process.

The problems that are addressed in the process of indexing strategy search and automatized indexing include the automatic construction of a query set on which the final indexing strategy is based; the analysis of queries, including the parsing of the query and the selection of indexes based on the query conditions; the methods for evaluating queries and indexes which are used in the search for indexing strategies; and finally, the search for the optimal indexing strategy.

The results of this paper are the different solutions to the mentioned problems. Some of the solutions are applied in practice which is resulted in an indexing tool which can be used for relational database indexing automatization, or just for helping select the right indexes for a set of queries.

# Literatūros sąrašas

[BHH78] G. E. Box, W. G. Hunter, J. S. Hunter. Statistics for experimenters. 1978, 629 pages.

[Cha98] S. Chaudhuri. An overview of query optimization in relational systems. In Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, 1998, pp. 34-43.

[Com79] D. Comer. Ubiquitous B-tree. ACM Computing Surveys (CSUR), 1979, pp. 121-137.

[DD87] C. J. Date, H. Darwen. A Guide to the SQL Standard (Vol. 3). New York: Addison-Wesley. 1987, 513 pages.

[GUW08] H. Garcia-Molina, J. D. Ullman, J. Widom. Database systems: the complete book. 2008, 1101 pages.

[McM04] P. McMinn. "Search-based software test data generation: a survey". Software testing, Verification and reliability 14.2, 2004, pp. 105-156.

[SW00] C. Samuelsson, M. Wiren. Parsing techniques. Handbook of Natural Language Processing, 2000, pp. 59-91.

[Win12] M. Winand. SQL Performance Explained. Winand Markus, 2012, 193 pages.