VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Magistrinis darbas

Žurnalo „Lietuvos matematikos rinkinys“ duomenų bazės sukūrimas ir realizavimas

Atliko: 2 kurso, 9 grupės studentas
Evaldas Ruseckas .....

Darbo vadovė:
dr. Olga Štikonienė .....

Vilnius
2008

Turinys

I. Anotacija 4
II. Summary 5
III. Įvadas 6
III.I. Darbo probleminė sritis 6
III.II. Darbo tematika 6
III.III. Darbo tikslai 7
III.IV. Temo aktualumas 8
III.V. Tyrimo metodai 9
IV. Teorinė bazė 10
IV.I. Duomenų bazių galimybės 10
IV.I.I. Operacinių sistemų palaikymas 10
IV.I.II. Pagrindinių savybių palaikymas 11
IV.I.III. Pseudo lentelių palaikymas 12
IV.I.IV. Indeksų palaikymas 13
IV.I.V. Duomenų bazių galimybės 13
IV.I.VI. Duomenų bazių padalinimų galimybės 15
IV.I.VII. Duomenų bazių limitai 16
IV.II. MySQL Duomenų bazė 16
IV.II.I. Manipuliavimas fiziniu duomenų bazės modeliu 16
IV.II.II. Užklausos. Jų sudarymas, optimizavimas 18
IV.II.III. Duomenų bazės vientisumas, kokybė 20
IV.II.IV. Mobilumas 21
IV.III. Interneto svetainė 22
IV.III.I. Algoritmų tinkamumas. Programavimo ypatumai 22
IV.III.II. Suderinamumas 23
IV.III.II. AJAX 24
IV.III.III. LINQ 25
V. Eksperimentinė bazė 26
V.I. Duomenų bazė 26
V.II. Interneto svetainė 30
V.III. Pritaikymas 33
VI. Literatūros analizė 34
VII. Išvados. Rezultatai 35
VIII. Literatūros sąrašas 36
Priedas Nr.1 - Duomenų bazės schema
Priedas Nr.2 - Tikrintos duomenų bazės

Priedas Nr.3 - Duomenų bazių palaikomos operacinės sistemos

Priedas Nr.4 - Duomenų bazių pagrindinių savybių palaikymas

Priedas Nr.5 - Duomenų bazių pseudo lentelių palaikymas

Priedas Nr.6 - Duomenų bazių indeksų palaikymas, išskyrus tradicinius B- / B+ medžius

Priedas Nr.7 - Duomenų bazių galimybės I

Priedas Nr.8 - Duomenų bazių galimybės II

Priedas Nr.9 - Duomenų bazių padalinimų galimybės

Priedas Nr.10 - Duomenų bazių limitai

I. Anotacija

Darbo esmė – išanalizuoti naujausius galimus literatūros šaltinius, įgalinančius kurti galimus problemos sprendimus. Spręsti problemą. Pateikti gautus teorinius ir eksperimentinius rezultatus. Suformuluoti išvadas.

Darbo tikslas – suformuluoti magistrinio darbo problemą, parengti teorinę bazę, atlikti praktinius eksperimentus bei juos pagrįsti.

Pasiekti rezultatai: išanalizavus turimą literatūrą, paskaitinėjus kelis internetinius forumus buvo pastebėta, kad daugiausia dėmesio skiriama greičiui. T.y. dažniausiai norima pasiekti, kad konkreti interneto svetainė kuo greičiau pasiektų vartotoją. Tad pagrindinis galutinio magistrinio darbo akcentas ir buvo maksimalaus greičio pasiekimas, panaudojant kuo mažiau resursų. Buvo tiriami šie dalykai: duomenų bazė, programavimas, bendravimas su duomenų baze. Tyrimo objektas – tai bakalaurinio darbo metu sukurta duomenų bazė ir interneto svetainė http://www.mii.lt/lmrj. Dažnai pasitaiko toki atvejai, kai vienai problemai išspręsti teoriškai egzistuoja keli variantai. Tačiau tik praktiškai viską patikrinus išaiškėjo, kuris iš tų galimų variantų yra tinkamiausias. Nuspręsta apsistoti ties labiausiai tinkančiomis priemonėmis tiriamam objektui. Pasiekta tris kartus didesnė sparta.

II. Summary

The title of thesis: Journal „Lietuvos Matematikos Rinkinys“ database creation and realization.

Main goals of work were to formulate purposes and researches for a master‘s work, to prepare pure base, to make and to prove experimental jobs.

During my work I analyzed all possible literature, read some internet forums, where I find out that mostly attention is pointed to overall speed. Videlicet commonly developers want to achieve, that their internet pages could reach final destination as quickly as possible. So the main accent of the master‘s work was to reach maximum speed with least resources. Researches were made on these objects: database, programming, communication with database. The most important object of researches was database and internet site http://www.mii.lt/lmrj, which I created in my bachelor‘s work. I find out that often occurs such situations, when in theory exists few solutions for one problem. But only in practice I got some answers, and now I know, which variant is the best. I decided to stay on mostly fittable possibilities for the research object. And in final version performance increased around three times.

III. Įvadas

III.I. Darbo probleminė sritis

Magistrinio darbo problema - maksimalaus greičio pasiekimas, panaudojant kuo mažiau resursų.

Tiriamas objektas – bakalaurinio darbo metu sukurta žurnalo „Lietuvos Matematikos Rinkinys“ duomenų bazė ir interneto svetainė.

Iškelti uždaviniai:

- Manipuliavimas fiziniu duomenų bazės modeliu
- Užklausos, jų sudarymas, optimizavimas
- Duomenų bazės vientisumas, kokybė
- Mobilumas, suderinamumas
- Optimalus ir saugus duomenų bazės naudojimas
- Algoritmų tinkamumas, programavimo ypatumai
- Įvairių duomenų bazių savybių tikrinimas

III.II. Darbo tematika

Šiais, sparčiai besivystančių technologijų, laikais vis daugiau informacinių dalykų yra pasiekiami vos per kelias sekundes. Visa tai pasiekti leidžia internetas. Dokumentus, straipsnius pateikiant internete, jie tampa matomais visam pasauliui, besinaudojančiam šia populiaria technologija.

Tačiau yra ir kita medalio pusė - vartotojai pripranta prie aukšto spartos lygio. O kai jiems yra reikalinga tam tikra informacija, slypinti lėtesniuose sluoksniuose, tai naudotojai paprasčiausiai nelaukia, o tiesiog išjungia tą interneto puslapį ir eina prie kito. Svetainių savininkai tą, žinoma, pastebi ir pradeda ieškoti būdų, kaip to išvengti, kaip pasiekti geresnę spartą, leisiančią išlaikyti visus galimus vartotojus.

Bene pats greičiausias, paprasčiausias, tačiau brangiausias būdas yra naujos techninės įrangos įsigijimas. Lėtesnis, pigesnis būdas yra duomenų bazės, užklausų optimizavimas, manipuliavimas, kuris galutiniam rezultate netgi gali duoti daug geresnius rezultatus nei kad

nauja įranga. Toliau eina programinio kodo peržiūrėjimas, keitimas, optimizavimas, kuris nors ir nedaug lemia, tačiau netinkamai parašytas gali labai stipriai sumažinti bendrą spartą.

III.III. Darbo tikslai ir uždaviniai

Pagrindiniai tikslai:

- suformuluoti magistrinio darbo temą ir tyrimus;
- parengti teorinę bazę, paremiančią analitinę ir eksperimentinę darbo dalį. Šis tikslas buvo išskaidytas į smulkesnius, tai yra buvo rengiama teorija atskirai kiekvienai iškeltai užduočiai. Tik prie techninės įrangos buvo apsieita be teorijos, nes tai daugiau praktinis dalykas. Be to, dauguma programinės įrangos kūrėjų nurodo reikalavimus, kuriuos patenkinus, naudojamas produktas veiktų optimaliausiai;
- tirti analitines hipotezes, atlikti praktinius eksperimentus bei juos pagrįsti. Pirmoji dalis užėmė mažesnę dalį laiko, nes neatsirado pakankamai daug naujų literatūros šaltinių. Praktiniai eksperimentai buvo atliekami remiantis surinktomis teorinėmis hipotezėmis. O tinkamumas beveik visais atvejais buvo tikrinamas spartos atžvilgiu.

Uždaviniai: manipuliavimas fiziniu duomenų bazės modeliu; skirtingų duomenų bazių palyginimas; užklausos, jų sudarymas, optimizavimas; duomenų bazės vientisumas, kokybė; mobilumas, suderinamumas; optimalus ir saugus duomenų bazės naudojimas; algoritmų tinkamumas, programavimo ypatumai.

Apie uždavinius plačiau:

- Manipuliavimas fiziniu duomenų bazės modeliu. Į šį punktą pateko toki dalykai kaip indeksai, trigeriai, įsimintos procedūros, laikinos lentelės, transakcijos, apkrovų numatymas, normalizavimas, raktai, bylų struktūros. Apie kiekvieną iš jų pateikta informacija teorinėje dalyje.
- Skirtingų duomenų bazių palyginimas. Kaip žinoma, yra sukurta nemažai įvairių duomenų bazių, kurios naudoja skirtingas struktūras. Pats kol kas esu susidūręs su MySql, MsSql, DB2, PostgreSql ir galiu pasakyt, kad bent jau iš sintaksės pusės jos daug kuo skiriasi. Tačiau šioje dalyje buvo nagrinėjama toki dalykai kaip indeksavimo, duomenų saugojimų struktūros ir kt.

- Užklausos, jų sudarymas, optimizavimas. Tos pačios užklausos gali būti parašytos įvairiais būdais. Taip pat jos yra suskirstytos į tipus. Buvo žiūrima, kas lemia lėtesnį ar greitesnį užklausos veikimą, išrenkami optimaliausi variantai.
- Duomenų bazės vientisumas, kokybė. Kokybė – tai vykstantis procesas, kuris padeda užtikrinti vientisumą. Pastarasis reikalingas duomenų tinkamumui. T.y. jeigu tam tikri duomenys bus sugadinti kokiu nors būdu, tai interneto puslapis nesitikėdamas tokio pokyčio gali paprasčiausiai tapti neveiksnus. Tad šis punktas nagrinėja vientisumo užtikrinimo metodus, priemones. Prie šios dalies taip pat priskirta perteklinės informacijos suderinimas, duomenų administravimas.
- Mobilumas, suderinamumas. Mobilumas – tai galimybė perkelti duomenis į kitokios architektūros duomenų bazę. Suderinamumas – tai galimybė naudotis interneto svetaine skirtingose operacinėse sistemose, naršyklėse, kompiuteriuose, neprarandant greičio. Tai yra reikalinga, nes negali iš anksto nuspėt, iš kur ir kas ateis į tavo interneto portalą. Todėl reikia užtikrinti vienodą greitį skirtingose sistemose.
- Optimalus ir saugus duomenų bazės naudojimas. Čia buvo analizuojami reikšmingi, nauji apsvarstymai šiais klausimais, atsižvelgiant į tokias technologijas, kaip .NET, ASPX, C#.
- Algoritmų tinkamumas, programavimo ypatumai. Šioje dalyje buvo nagrinėjami visi dalykai, susiję su programavimu. Išryškinta algoritmų tinkamumas, nes dažnai pasitaiko tokių atvejų, kai tą patį veiksmą galima atlikti keliais skirtingais būdais, kurių sparta yra skirtinga. Tad buvo žiūrima, kaip koks algoritmas veikia, ir buvo palikti tik tinkamiausi. Taip pat įėjo darbas su duomenimis, informacijos išvedimas, įvedimas, perdavimas, laikinas laikymas ir kiti panašūs dalykai.

III.IV. Temos aktualumas

Galbūt temos aktualumas ir nėra toks svarbus tarptautiniu ar šalies mastu, nes prie to dirba gan daug įvairių profesijų žmonių, tačiau tai man asmeniškai suteiks daugiau žinių, kurios padėtų ateityje dirbant atlikti savo užduotis. Be to, darbo vadovė turėtų puikiai veikiantį produktą.

Taigi perskaičius turimą literatūrą bei kelis internetinius šaltinius paaiškėjo, kad dažniausiai nepasitenkinimas reiškiamas tokiems dalykams, kaip sparta, dizainas, patogumas, aiškumas, paprastumas, saugumas ir kiti mažiau pasitaikantys. Bet bene svarbiausias iš jų yra

sparta, nes būtent pastarajam daugiausia dėmesio skiria ir programinės įrangos kūrėjai. Tad vėliau buvo atliktas tyrimas, ieškant literatūroj įvairių aspektų, susijusių su sparta, kurie leistų iškelti magistrinio darbo uždavinius. Šie yra pateikti prieš tai buvusiam skyriuj.

Iškeltos užduotys leido pažvelgti giliau į pačią duomenų bazės struktūrą, daugiau sužinoti apie techninę bei programinę įrangą, patobulinti programavimo įgūdžius, išmokti analizuoti, sukurti analitinį, eksperimentinį, praktinį modelius.

III.V. Tyrimo metodai

Tyrimas buvo atliekamas dviem požiūriais, t.y. analizuojama literatūra, ieškomi galimi sprendimai problemos išsprendimui, bei skaitomi forumai, įvairios vartotojų diskusijos, kad būtų aišku, kur labiausiai reikėtų pritaikyti gautus sprendimus.

Taip pat buvo žiūrima, ar bus įmanoma įvairius gautus sprendimus pritaikyti tiriamam objektui, t.y. bakalaurinio darbo metu sukurtai duomenų bazei ir interneto svetainei. Taigi nustatyta, kad visko tikrai nesigaus panaudot, nes Matematikos ir Informatikos Instituto serveriai, kuriuose šiuo metu patalpintas tiriamas objektas, yra pritaikyti ne visoms galimoms technologijoms. Tačiau analizė buvo atliekama, nes tai gali praversti ateityje.

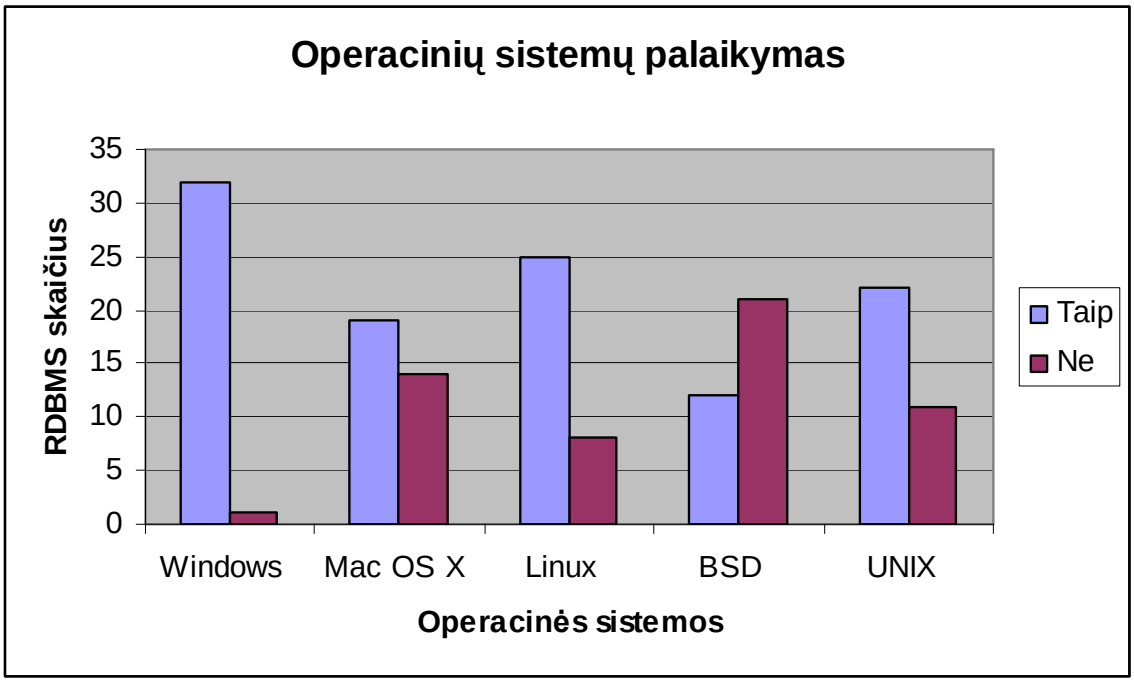
Gauti sprendimai buvo išbandomi praktiškai eksperimentuojant ant mano asmeninio kompiuterio. O tinkamiausi ir įmanomi buvo pritaikomi tiriamam objektui.

IV. Teorinė bazė

IV.I Duomenų bazių galimybės

Galimybių tikrinimas buvo vykdomas, tikrinant oficialiai internete pateiktus duomenis apie duomenų bazes ir Wikipedia, bei naudojant tam skirtus įrankius, tokius kaip Zidsoft produktas CompareData 1.4.3, DB SOLO LLC – DB SOLO 3, DBBALANCE – Cross-Database 6.0. Iš esmės visi įrankiai skirti tam pačiam tikslui, tačiau šiek tiek skyrėsi jų teikiami pasirinkimai, todėl panaudoti buvo visi. Iš viso tikrinimui buvo pasirinktos 37 reliacinės duomenų bazės. Pilnas sąrašas pateiktas antrame priede.

IV.I.I Operacinių sistemų palaikymas



1pav. Operacinių sistemų palaikymas

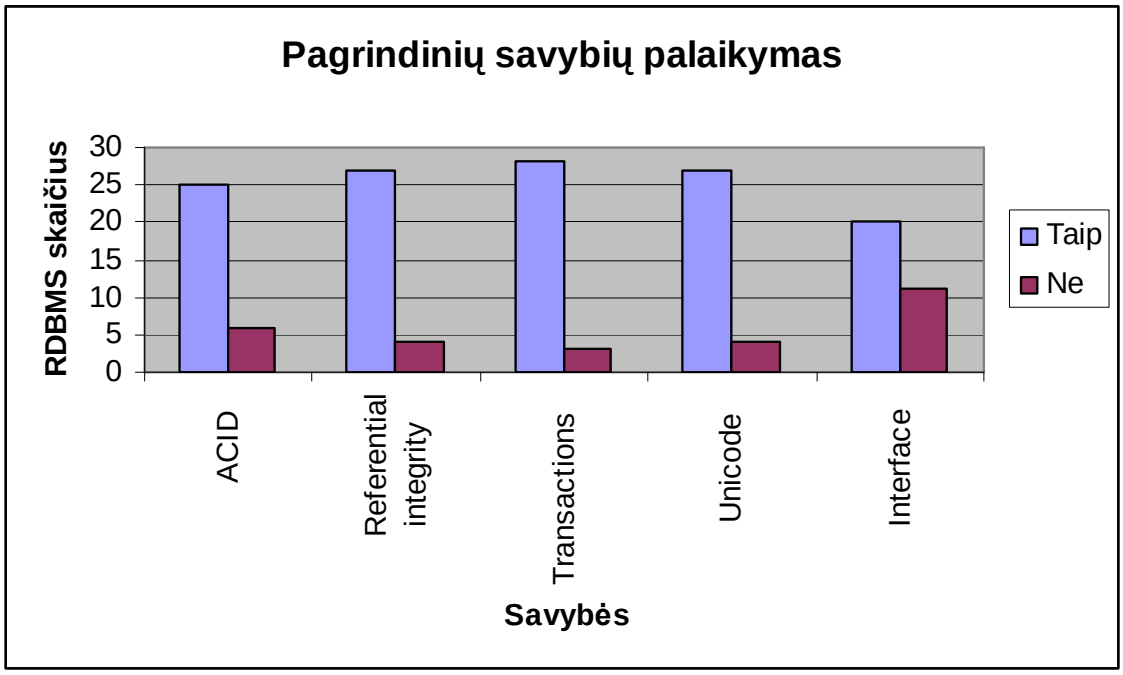
Dauguma reliacinių duomenų bazių yra pritaikytos Windows operacinei sistemai, išskyrus vieną – Oracle Rdb, kuri buvo sukurta OpenVMS platformai (Windows versija nebuvo sukurta, nes Oracle nepateikė reikalingų kompiliatorių). Tolesnis pasiskirstymas yra gan tolygus, tačiau kas yra pažymėtina dėl populiaresnių duomenų bazių:

- MsSQL veikia tik su Windows;
- Oracle nepritaikyta BSD;

- DB2 nepritaikyta BSD ir Mac OS C.

Pilnas sąrašas pateiktas trečiame priede.

IV.I.II Pagrindinių savybių palaikymas

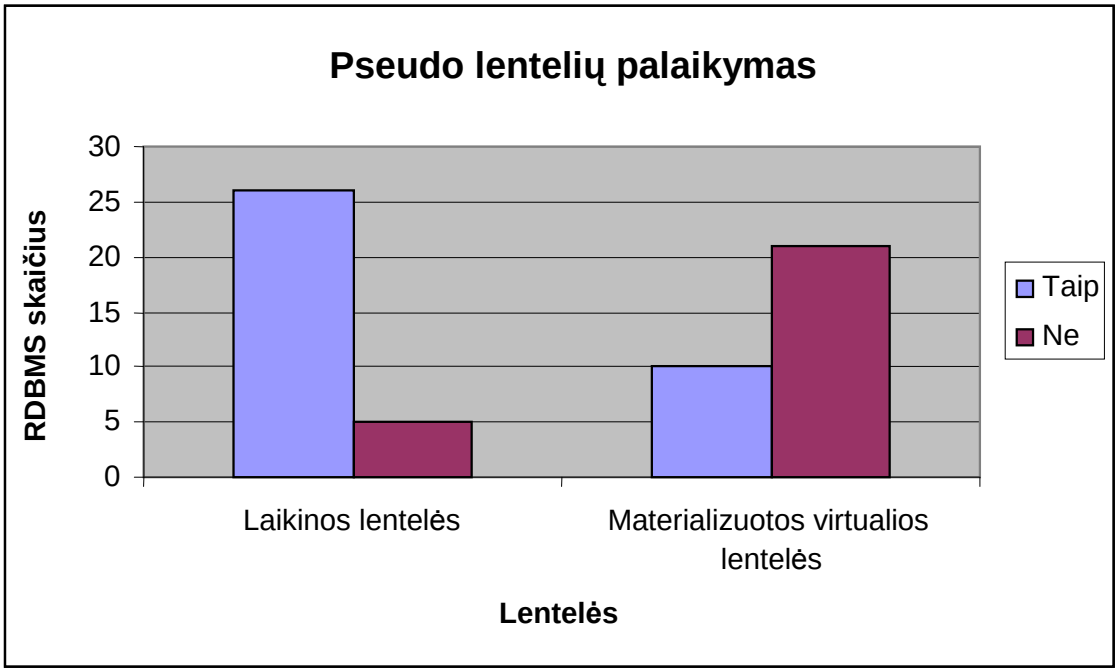


2pav. Pagrindinių savybių palaikymas

Apie pačias savybes, kurios pateiktos anglų kalba. ACID – Atomicity (atomiškumas), Consistency (pastovumas), Isolation (izoliacija), Durability (ilgaamžiškumas). Atomiškumas – tai duomenų bazės sugebėjimas užgarantuoti, kad visi transakcijos veiksmai bus įvykdyti arba nė vienas iš jų. Pastovumas – tai duomenų bazės pastovios būsenos išlaikymas prieš prasidedant transakcijai ir jai pasibaigus. Izoliacija – tai galimybė atskirti vykstančią transakciją nuo kitų procesų. Ilgaamžiškumas – tai garantija, kad transakcija bus įvykdyta, jeigu vartotojui buvo pranešta apie sėkmę, tačiau pavyzdžiui staiga įvyksta sisteminė klaida. Referential integrity (sąryšių vientisumas) – tai sąryšių tarp lentelių, panaudojant pirminius ir šalutinius raktus, palaikymas. Transactions (transakcijos) – transakcijų palaikymas. Unicode – tai tarptautinis standartas, kurio palaikymas leidžia naudoti įvairių kalbų raides. Interface (vartotojo sąsaja) – tai galimybė dirbti su duomenų baze, naudojantis grafine sąsaja, o ne komandine eilute.

Iš esmės visi populiarūs įrankiai palaiko šias savybes. Tačiau MsAccess ir MsVisual Foxpro nepatenka į ACID sąrašą. Taip pat gan didelis skaičius, neturinčių normalios vartotojo sąsajos. Pilnas sąrašas pateiktas ketvirtame priede.

IV.I.III Pseudo lentelių palaikymas



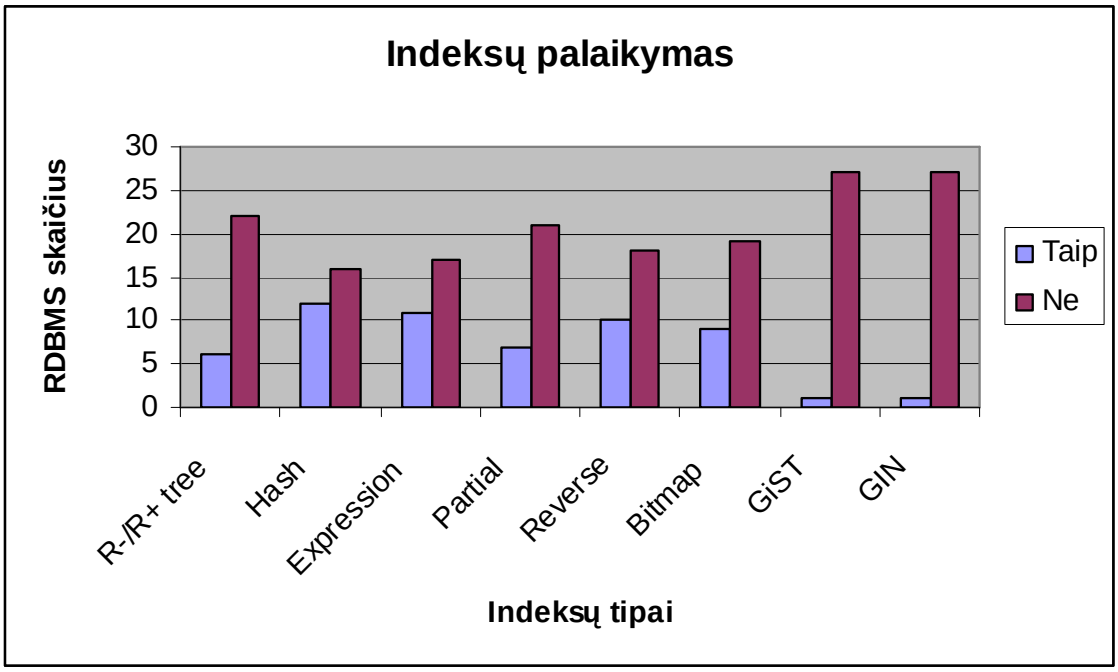
3pav. Pseudo lentelių palaikymas

Laikinų lentelių nepalaiko vos keletas mažiau žinomų reliacinių duomenų bazių. Tuo tarpu materializuotų virtualių lentelių nepalaiko gana daug duomenų bazių, tarp kurių yra tokios kaip MySQL, MsAccess ir PostgreSQL (kai kur yra siūloma naudoti įsimintinas procedūras su trigeriais, taip gaunant panašų rezultatą). Paprastos virtualios lentelės yra gausiai palaikomos, tačiau jos nėra materializuotos. Pastarosios yra išsaugomos atsargos atmintyje ir laikas nuo laiko atnaujinamos, kad jose esantys duomenys nepasentų. Šios lentelės yra trijų tipų:

1. Tik skaitymui. Negali būti atnaujinamos.
2. Atnaujinamos. Gali būti atnaujintos net tada, kai nebendraujama su vartotojo sąsaja. Gali būti atnaujinama bet kuriuo metu, kai tik to reikia. Naudoja mažiausiai resursų iš visų trijų tipų.
3. Rašymui. Sukuriamos su prierašu atnaujinimui. Prarandami pakeitimai, kai įvyksta atnaujinimas.

Pilnas sąrašas pateiktas penktame priede.

IV.I.IV Indeksų palaikymas



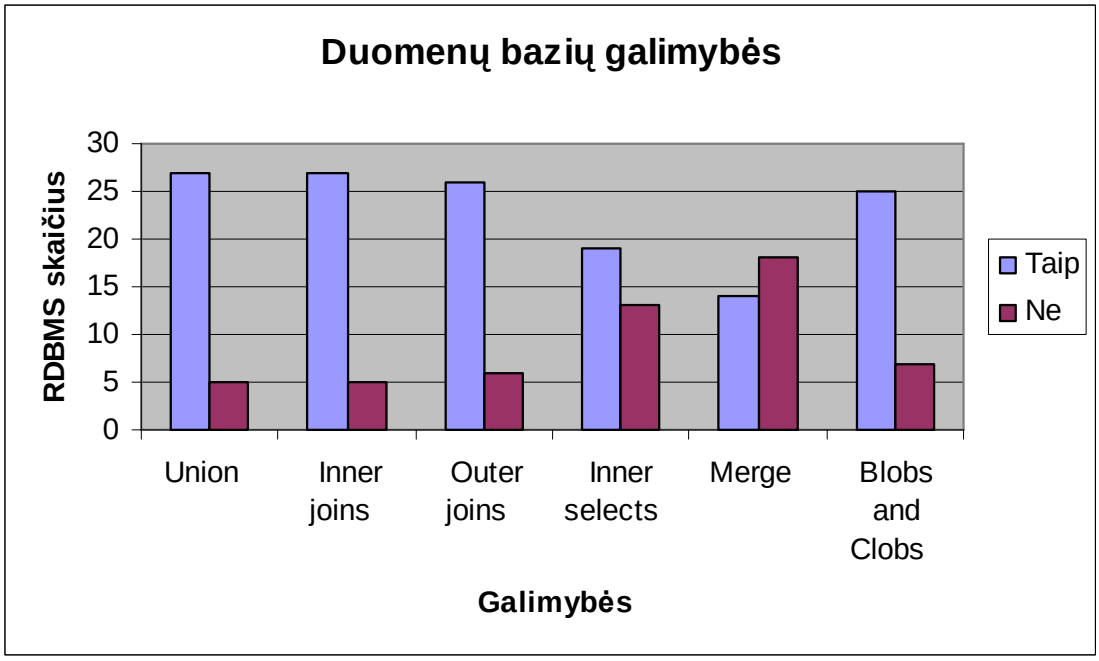
4pav. Indeksų palaikymas

Į grafiką neįtrauktas tradicinis ir kartu populiariausias indeksų tipas, tai yra B-/B+ medžiai, kurį pagal nutylėjimą privalo palaikyti visos duomenų bazės. Apie pačius indeksų tipus daugiau nieko nebus pateikta, nes tai yra gana didelis kiekis informacijos, kuris nesiderina su darbo tematika. Tad šioje kategorijoje geriausiai pasirodė „PostgreSQL“, kuri palaiko visus tipus. Informix, Oracle ir MsSQL visiškai nepalaiko tik GiST (apibendrinti paieškos medžiai) ir GIN (apibendrinti perversti indeksai), kiti tipai yra palaikomi pilnai arba su dalinėmis išimtimis, pavyzdžiui, tik klasterių lentelėse. Taigi žiūrint iš indeksavimo pusės, dauguma reliacinių duomenų bazių dar turi kur tobulėti. Pilnas sąrašas pateiktas šeštame priede.

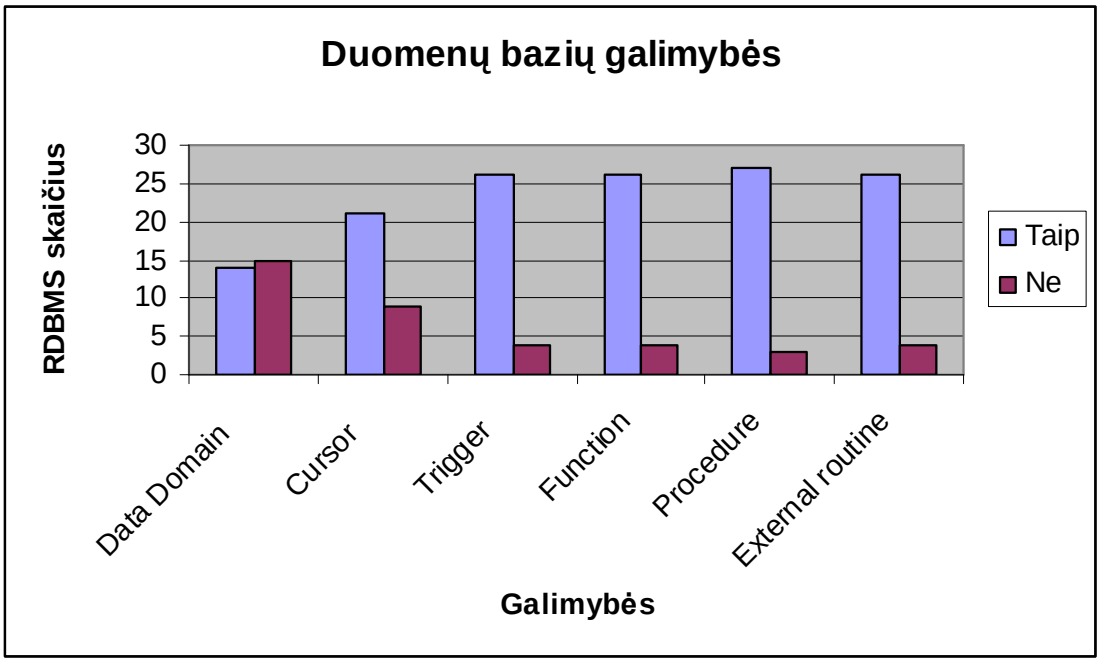
IV.I.V Duomenų bazių galimybės

Šioje vietoje buvo tikrinamas užklausose naudojamų operacijų palaikymas. Dažniau naudojamose reliacinėse duomenų bazėse pirmos trys operacijos yra palaikomos šimtu procentų. Sekančių dvejų operacijų neturi tik MsAccess duomenų bazė, žvelgiant iš žinomesnių pusės. O MsVisual Foxpro nepalaiko tik Merge (sujungimas). Paskutinis pateiktas stulpelis – tai duomenų tipas, skirtas saugoti dideliems duomenų kiekiams

dvejetainiu arba simboliniu formatu. Tai suteikia patogų būdą saugoti bylas duomenų bazėse. Pilnas sąrašas pateiktas septintame priede.



5pav. Duomenų bazių galimybės I



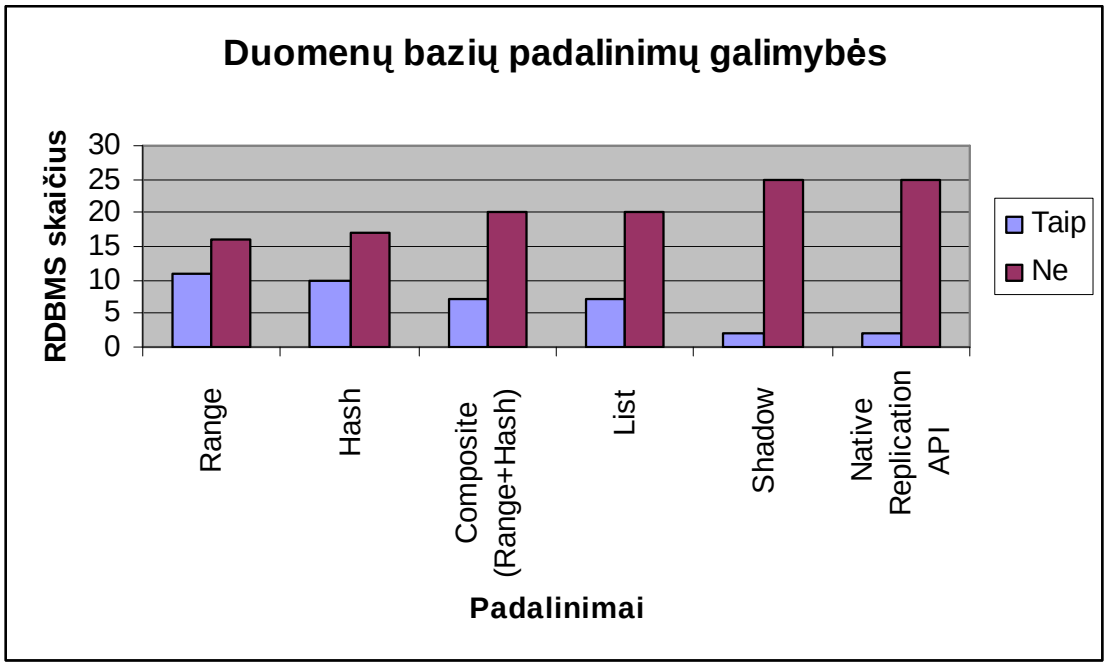
6pav. Duomenų bazių galimybės II

Data domain (duomenų sritis) – tai visos unikalios reikšmės, kurias gali turėti konkretus duomenų elementas. Pavyzdžiui, giminė gali turėti dvi reikšmes: vyriška, moteriška. Šios dvi reikšmės sudaro duomenų sritį. Cursor (kursorius) – tai iteratorius per eilutes, esančias

duomenų aibėje, gautoje įvykdžius užklausą. Trigger (trigeris) – tai automatiškai vykdoma procedūra, kuri yra atliekama, kai yra įvykdomi tam tikri specifiniai veiksmai su konkrečia lentele ar visa duomenų baze. Function (funkcija), Procedure (procedūra) ir External routine (išorinė paprogramė) yra tam tikras programavimo kodas, atliekantis konkretų veiksmą ir grąžinantis norimą rezultatą.

Šių galimybių palaikymas yra aukštas, išskyrus duomenų sritis. Pastarųjų nepalaiko tokios duomenų bazės, kaip MySQL, MsVisual Foxpro, DB2 ir daugelis kitų. Pažymėtina, kad MsAccess iš visų pateiktų galimybių palaiko tik išorines paprogrames. Pilnas sąrašas pateiktas aštuntame priede.

IV.I.VI Duomenų bazių padalinimų galimybės



7pav. Duomenų bazių padalinimų galimybės

Padalinimai atskiria duomenų bazes ar jas sudarančius elementus į nepriklausomas atskiras dalis (partitijas). Tai yra reikalinga, kad pagerint šias savybes: sparta, paprastumas, pasiekiamumas. Range (ribos) – tai eilučių paėmimas, kurių konkretaus stulpelio reikšmės patenka į nurodytą diapazoną, pvz. 0-10. Hash išlaiko sąryšius tarp atskirų dalių. Composite (mišrūs) gali apimti kelis padalinimų tipus, kurių panaudojama tvarka taip pat gali skirtis. List (sąrašai) – tai eilučių paėmimas, kurių konkretaus stulpelio reikšmės turi konkrečias reikšmes, pvz. stulpelis „Valstybė“, particija „Baltijos valstybės“, reikšmės „Lietuva, Latvija,

...“. Shadow (šešėlinis) padalinimas sukuria laikiną particiją pagal bet kokius duomenis. Native replication API – tai vidinis tik konkrečiai duomenų bazei pritaikytas padalinimas.

Apskritai imant padalinimai nėra išplitę tarp visų sistemų, tačiau tarkim MySQL žada palaikymą naujoje versijoje. Geriausiai šias operacijas palaiko PostgreSQL, Oracle, Teradata, Informix, DB2, Ingres. Kitos reliacinės duomenų bazės ženkliai atsilieka. Pilnas sąrašas pateiktas devintame priede.

IV.I.VII Duomenų bazių limitai

Maksimalus galimas duomenų bazės dydis daugumoje yra neapibrėžiamas, tai yra viskas priklauso nuo kieto disko dydžio. Bet tarkim MsAccess ar MsVisual Foxpro galimas tik 2GB dydis, kas šiais laikais tikrai nėra daug.

Maksimalus lentelės dydis neapibrėžtas tik dvejose sistemose: Ingres ir Teradata. Visi kiti taiko tam tikrus limitus. Dažniausiai pasitaikantis yra 2GB.

Maksimalus eilutės dydis neapribotas taip pat tik dvejose duomenų bazėse: Oracle ir Polyhedra DBMS. Kitur dydžiai labai skiriasi, bet vyraujanti tendencija yra 16-64MB.

Maksimalus stulpelių skaičius vienoje lentelėje limituotas absoliučiai visur, ir dar yra priklausomybė nuo duomenų tipų, kurie priskiriami stulpeliams. Daugiau nei 65000 leidžia sudaryti tik Polyhedra DBMS. Visi kiti svyruoja tarp 255-1024.

Maksimalus Blob (dvejetainio formato duomenys) ir Clob (simbolinio formato duomenys) tipus turinčių stulpelių dydis apribotas visur. Dažniausiai 2-4GB. Tačiau ScimoreDB leidžia tam panaudoti net 16TB.

Maksimalus simbolinio tipo dydis yra limituotas ir kartu labiausiai varijuojantis. Nėra jokių tendencijų. Didžiausias leidžiamas kiekis yra pas Polyhedra DBMS – 4GB.

Maksimalus skaitinio tipo dydis dažniausiai leidžiamas iki 8B. Tačiau PostgreSQL jo nelimituoja.

Pilnas sąrašas yra pateiktas dešimtame priede.

IV.II. MySQL duomenų bazė

Toliau sekantis aprašas remiasi MySQL duomenų baze, nes tiriamas objektas yra serveryje, kuris naudoja būtent šią duomenų bazę. Realiai, jeigu būtų galima rinktis, tai būtų naudojama kuri nors iš šių trijų: MsSQL, Oracle, InterSystems Cache. Pirmos dvi dėl teikiamų galimybių. Paskutinė dėl savo spartos.

IV.II.I. Manipuliavimas fiziniu duomenų bazės modeliu

Ši dalis remiasi [BE06] ir [PUR00].

Indeksavimas pagreitina užklausų veikimą, paiešką indeksuojamuose stulpeliuose, palengvina rūšiavimą ir grupavimą. Kuo daugiau duomenų yra lentelėje, tuo svarbesnis tampa tinkamas indeksavimas. Vienai lentelei galima sukurti kelis indeksus, tačiau kuo daugiau yra indeksų, tuo lėčiau veikia įterpimo, keitimo ir trynimo operacijos. Todėl viską reikia suderinti. Indeksai yra dviejų tipų: paprasti ir unikalūs. Paprasti indeksai gali būti naudojami su visais duomenimis, kurie nėra šių tipų: TEXT, NTEXT, IMAGE. Tuo tarpu unikalūs indeksai naudojami tuose stulpeliuose, kuriuose duomenys nepasikartoja, tačiau jie negalimi raktiniams stulpeliams, nes gali turėti tuščias reikšmes.

Tiriamo objekto galimos lentelių struktūros:

- InnoDB. Duomenys laikomi diske. Galimas eilučių blokavimas. Saugus transakcijų atlikimas. Rekomenduojamas lentelėms, naudosiančioms transakcijas;
- Falcon. Palaiko transakcijas. Rekomenduojamas aplikacijoms, programoms, bet ne interneto svetainėms.
- MyISAM. Labai greitas. Duomenys laikomi diske. Nepalaiko transakcijų. Palaiko paiešką visame tekste, suarchyvuotus raktus. Pasirinkta naudoti šią struktūrą, nes nebus naudojamos transakcijos, ir tai viena greičiausiai veikiančių struktūrų.
- Merge. Ši struktūra apjungia visas lenteles, naudojančias MyISAM, kurių stulpeliai ir indeksai yra identiški. Rekomenduojama naudoti lentelėms, kuriose būtų saugomi archyvuoti duomenys arba įvairūs protokolų žurnalai.
- Memory. Itin greitas. Duomenys laikomi atmintyje. Naudoja ‚hash‘ funkcijas. Rekomenduojamas naudoti su laikinomis lentelėmis.
- BDB. Saugus transakcijų atlikimas. Galimas eilučių blokavimas.
- NDB. Tik MySQL palaikoma klasterinė struktūra. Saugus transakcijų naudojimas. Duomenys laikomi atmintyje. Galimas eilučių blokavimas. Rekomenduojamas realaus laiko kritinėms aplikacijoms.
- Archive. Visi duomenys yra suarchyvuojami, tad naudoja mažiau vietos diske. Rekomenduojamas tokios lentelėms, kuriuos būtų gan retai naudojamos.
- CSV. Duomenys saugomi kaip sąrašas, atskirtas kableliais. Nepalaiko indeksavimo. Rekomenduojamas lentelėms, kurių duomenys būtų dažnai eksportuojami.

- Blackhole. Ši struktūra veikia kaip „juodoji skylė“: priima naujus duomenis, tačiau jų neišsaugo, tai yra iškart ištrina.

Sql serveriai naudoja atsargos atminties (cache) lenteles, kuriose saugo visas vykdomas užklausas ir jų įvykdymų skaičių. Jeigu yra naudojamos paprastos ‚select‘ tipo užklausos, tai kiekvieną kartą šiek tiek pakeitus užklausą, jai perdavus kitokius parametrus arba pačią užklausą įvykdžius kitam vartotojui, yra sukuriamas naujas įrašas atsargos atminties lentelėje. Kuo daugiau tokių įrašų, tuo lėtesnis tampa veikimas. Kad to išvengti, buvo sukurta įsimintinos procedūros. Jos yra sukompiliuojamas tik vieną kartą. Tai reiškia, kad į atsargos atminties lentelę yra įterpiamas tik vienas įrašas. Ir po to visiškai nepriklausomai nuo įsimintinos procedūros įvairių naudojimų, nauji įrašai nėra įterpiami, tik yra skaičiuojama, kiek kartų procedūra buvo panaudota. Tai leidžia išlaikyti spartų duomenų gavimą. Taip pat pliusas yra tas, kad į šias procedūras galima perkelti dalį logikos, atliekamos vartotojo sąsajos dalyje. Žinoma, bus bandoma ta pati logika abejose dalyse, ir bus paliekamas spartesnis variantas.

Jeigu nėra palaikomos arba tiesiog nenaudojamos įsimintinos procedūros, tai tada rekomenduojama naudoti trigerius. Trigeris – tai objektas, susietas su konkrečia realia arba virtualia lentele, kuris įvykdo kokį nors veiksmą, jeigu toje lentelėje yra įterpiami, keičiami, trinami duomenys. Trigeris gali iškviesti reikalingą veiksmą ne tik konkretaus įvykio metu, bet ir prieš jį arba po jo. Juos gan sunku testuoti, tačiau tai gali stipriai pagerinti duomenų bazės vientisumą.

Normalizavimas padeda išvengti duomenų anomalijų, vykdant pakeitimo procedūras. Tačiau tai taip pat naudoja daugiau atminties ir užima daugiau laiko. Tad šioje vietoje bus bandoma surasti vidurkį tarp normalizavimo ir spartos palaikymo.

IV.II.II. Užklausos. Jų sudarymas, optimizavimas

Ši dalis remiasi [BE06].

Dažniausiai lėtai pradeda veikti ilgos užklausos. Norint to išvengti, naudojama nuoseklus užklausos sudarymo planas arba pseudo lentelės.

Užklausos sudarymo planas:

1. Pasirinkti, kuri lentelė bus pagrindinė, kokie duomenys bus reikalingi;
2. Pasirinkti papildomas lenteles (realias arba pseudo) bei reikalingus duomenis iš jų, tinkamai sujungti su pagrindine lentele. Lentelių jungimui yra naudojama du

būdai: JOIN operacija arba naudojant kitas užklausas užklausos viduje. Egzistuoja dvi pagrindinės problemos, pagal kurias reikia nuspręsti, kurį jungimo panaudoti. Pirma, reikia nuspręsti, kaip bus gaunama informacija. Jeigu vienam būde atsiras kažkoks apribojimas, trukdysiantis paimti duomenis, tai teks rinktis kitą būdą. Antra, reikia kreipti dėmesį į užklausos veikimo spartą, kas šiam darbe yra svarbiausia. Dažnai tokios pačios užklausos skirtingose duomenų bazėse veikia skirtingai, todėl viską reikia testuoti. Taip pat yra svarbu, per kokius stulpelius yra jungiamos lentelės. Jeigu tai indeksuojami arba raktiniai stulpeliai, tai užklausos veiks greičiau. Jeigu tai nėra toki stulpeliai, tai reikia, kad stulpeliai lentelėje stovėtų kuo kairiau, nes būtent iš kairės į dešinę yra nuskaitoma lentelė. Tai yra rečiausiai naudojamus stulpelius reikėtų pastatyti kuo dešiniau. Užklausas užklausoje galima panaudoti bet kurioje vietoje. Jeigu tokios užklausos yra naudojamos filtravimo dalyje, tai ten jas galima pritaikyti dviem būdais: naudojant IN arba EXISTS. Kuris iš jų geresnis negalima pasakyti, reikia išbandyti abu atvejus ir palikt tinkamesnį;

3. Nustatyti filtrus. Jeigu filtrų yra daugiau nei du, tai rekomenduojama naudoti suskliaudimus, nes tada galima aiškiau matyti bei nustatyti eiliškumą. Taip pat palengvėja filtrų derinimas. Filtruoti galima dviem būdais: naudojant WHERE arba HAVING. Visais įmanomais atvejais reikia naudoti pirmą variantą, nes jis veikia greičiau. Be to, antras variantas veikia tik su paimtais duomenimis;
4. Pasirinkti rūšiavimą;
5. Pasirinkti grupavimą.

Pseudo lentelės tampa naudingos, kai užklausos būna labai didelės ir sunkiai sekasi susigaudyti jose. Tada užklausa yra išskaidoma ir iš tam tikros jos dalies yra sudaroma pseudo lentelė. Pastarosios yra kelių tipų, tačiau pagrinde yra naudojama šios:

1. Virtualios. Tai mechanizmas gauti tam tikrą duomenų kiekį, esantį realiose lentelėse. Pagrindinis pliusas tas, kad virtualios lentelės užima mažai vietos, nes neturi savo duomenų. Kiti pliusai, kurie nėra reikšmingi šiam darbui: galima panaudoti tam tikros duomenų bazės dalies matymo uždraudimui; panaudoti abstrakcijos sudarymui; duomenų gavimas iš skirtingų šaltinų; naudojama taip pačiai, kaip reali lentelė; galima naudoti kitų užklausų arba virtualių lentelių kūrimui;
2. Laikinos. Kiekvienas sql serveris turi laikiną duomenų bazę, kurioje ir yra laikomos laikinos lentelės. Jeigu serveris yra paleidžiamas iš naujo, tai laikina

duomenų bazė yra sukuriama taip pat iš naujo, tai yra visos laikinos lentelės yra pašalinamos. Minusas tas, kad užima tuo daugiau vietos, kuo daugiau duomenų turi. Pliusas, kad duomenų keitimas joje neliečia tų duomenų, kurie yra realioje lentelėje, kai tuo tarpu virtuali lentelė keičia visur. Laikinos lentelės būna lokalios (veikia tik vienoje duomenų bazėje) ir globalios (veikia visose serveryje esančiose duomenų bazėse).

IV.II.III. Duomenų bazės kokybė, vientisumas

Ši dalis remiasi [BE06] ir [PUR00].

Duomenų vientisumas – tai duomenų patikimumas, teisingumas, tinkamumas. T.y. jeigu tam tikri duomenys bus sugadinti kokiu nors būdu, tai interneto puslapis nesitikėdamas tokio pokyčio gali paprasčiausiai tapti neveiksnus. Duomenys atomų lygmenyje bei tarpusavyje susaistyti išoriniais raktais turi būti teisingi. Sugadinti duomenis gali vartotojai, nepavykusios operacijos, kurios neturi tinkamos sugrąžinimo procedūros, netikėtos klaidos, virusai, piratai. Nevientisi duomenys gali nustelbti kitus duomenų bazės procesus, kurie yra atsakingi už spartų veikimą, todėl ši dalis taip pat yra svarbi. Patikimumo užtikrinimui siūloma atkreipti dėmesį į šiuos dalykus:

- Duomenų balansavimas;
- Apskaitos laikotarpio sustabdymas;
- Duomenų bazės išbaigtumas;
- Duomenų bazės atstatymas;
- Procesų atkūrimas;
- Serviso atitikimas;
- Duomenų pateikimas tam tikriems vartotojams;
- Duomenys apibūdinimai turi būti pilni;
- Saugumo pakankamumas;
- Individualių duomenų apsauga;
- Duomenų tikslumas;
- Importuotų duomenų tinkamas panaudojimas;
- Atsakomybių priskyrimo atitikimas;
- Skirtingų duomenų bazių palaikymas.

Apribojimai yra labai efektyvus būdas vientisumo užtikrinimui. Tarkim duomenys yra neteisingi, tai tada sakoma, kad trūksta vientisumo. Apribojimai taip pat naudojami įtvirtini ryšiams tarp lentelių. Apribojimų tipai:

- Būtina reikšmė. Įterpimo operacijos metu yra tikrinama, ar stulpeliui priskiriama reikšmė nėra tuščia, jei yra, tai operacija nutraukiama;
- Pagrindinis raktas. Patikrina, kad visos reikšmės būtų unikalios. Automatiškai sukuria būtinos reikšmės apribojimą ir paprastą indeksą. Vienai lentelei galimas tik vienas toks apribojimas;
- Nepasikartojančios reikšmės. Užtikrina reikšmių unikalumą. Automatiškai sukuria unikalų indeksą;
- Koks nors reikšmės patikrinimas. Galima nustatyti tam tikras reikšmės ribas, kurių netenkinant operacija būtų nutraukiama;
- Šalutinis raktas. Naudojamas lentelių sujungimui. Draudžiama įterpti, jeigu neegzistuoja pagrindinis raktas prijungtoje lentelėje, ir ištrinti, jeigu egzistuoja šalutinis raktas prijungtoje lentelėje.

Kokybė – tai vykstantis procesas, kuris padeda užtikrinti vientisumą. Šiam procesui galima panaudoti FDS (foundation data store) konstruktorių. Pastarasis susideda iš reliatyviai išliekančios informacijos, kuri yra tradiciškai randama duomenyse. Ši informacija laikosi bendrų standartų, esančių duomenų aprašuose, struktūrose. FDS teikiama nauda:

- Sumažina pastangų kiekį duomenų saugojimo aplikacijų kūrimui, nes duomenų vientisumo pasiekimas tampa paprastesnis;
- Geresnis klientų aptarnavimas, nes kokybė tampa aukštesnio lygio;
- Išlaidų sumažinimas, nes centralizuojama duomenų priežiūra ir tampa efektyvesnis duomenų importavimas iš išorinių šaltinų;
- Geresnis sprendimų priėmimas, nes gaunama tikslesnė informacija;
- Efektyvesnis transakcijų panaudojimas, nes nebandoma tikrinti tų pačių duomenų du kartus ir daugiau.

IV.II.IV. Mobilumas

Ši dalis remiasi [PUR00].

Mobilumas – tai galimybė perkelti duomenis į kitokios architektūros duomenų bazę. Duomenų bazės, naudojamos interneto svetainėse, turi būti aukšto mobilumo lygio, kad būtų

galima greitai atlikti smulkius pakeitimus, ir padaryti interneto svetainę veiklią. Niekada negali žinoti, ar reiks pereiti prie kito tipo duomenų bazės ar ne. Kad tai būtų galima padaryti kuo greičiau, reikia naudoti standartinius dalykus lentelių sudaryme, duomenų aprašyme, užklausų sudaryme, apribojimuose, stulpeliuose, abstrakcijose. Žinoma, dažnai yra naudojamos pačių duomenų bazių funkcijos, tad reikėtų stengtis panaudot tokias, kurios turi atitikmenis kitokiose duomenų bazėse. Iš esmės reikia išspręsti šiuos tris dalykus, norint pasiekti gerą mobilumą:

1. Aptikti ir išspręsti struktūrinius nesuderinamumus;
2. Aptikti ir išspręsti duomenų nesuderinamumus;
3. Subalansuoti tikslaus duomenų bazės prižiūrėjimo kainą su klaidų, įvykusių dėl gautų blogų duomenų, kaina.

IV.III. Interneto svetainė

IV.III.I. Algoritmų tinkamumas. Programavimo ypatumai

Ši dalis remiasi [ABC07] ir [LIB05].

Tiriamo objekto interneto svetainė sukurta, naudojant .NET technologijas: c-sharp, asp2.0, javascript, css. Tad buvo skaitoma literatūra, kaip geriau panaudoti būtent šias technologijas.

Darbas su duomenimis: išvedimas, įvedimas, perdavimas, laikymas. Duomenų išvedimui galima panaudoti tris komponentus, kurie yra visiškai skirtingi. Pirmasis iš jų yra *repeater*. Tai greičiausiai veikiantis komponentas, tačiau jis turi mažiausiai galimybių, papildomų funkcijų. Rekomenduojamas naudoti tuo atveju, jeigu tiesiog reikia išvesti duomenis ir nieko daugiau. Antrasis yra *datalist*. Veikia lėčiau už pirmąjį, tačiau turi tokių papildomų galimybių, kaip kad rūšiavimas, eilutės pasirinkimas (tai leidžia dinamiškai pakeisti šablonus). Rekomenduojamas naudoti tada, kai norima daugiau funkcionalumo ir galimybės keisti išvaizdą. Trečiasis yra *listview*. Tai naujai išleistas komponentas, kurį stengtasi padaryti kuo įvairiapusiškesniu. Tai tikrai yra pavykę, tačiau beveik kiekvieną veiksmą reikia detaliai pačiam aprašyti. Sparta priklauso tik nuo to, kaip yra atliekami visi aprašymai, ir gali būti tiek greitesnis už *repeater*, tiek pats lėčiausias. Tačiau žiūrint iš funkcionalumo pusės, tai yra puikus įrankis duomenų išvedimui, nes suteikia tikrai daug galimybių. Paskutinis komponentas yra *datagrid*. Tai lėčiausiai veikiantis iš visų galimų, tačiau taip pat turintis daugiausiai galimybių, funkcijų: rūšiavimas, dinamiškas stiliaus parinkimas, atpažįsta duomenų bazės schemas, duomenų tipus. Bene pagrindinis jo koziris tas, kad galima

redaguoti duomenis neperkraunant puslapio. Tad ir rekomenduojamas tada, kai reikalingas duomenų redagavimas. Bus pritaikytas administravimo dalyje. Duomenų įvedimui patariama naudoti javascript funkcijas, kurios patikrintų jų tinkamumą. Tai leistų užtikrinti didesnį duomenų bazės vientisumą jau vartotojo sąsajos dalyje. Duomenims perduoti paprastiems vartotojams prieinamoje interneto svetainės dalyje rekomenduojama naudoti neprieinamas formas, kaip kad sesija arba *viewstate*. Nes kitu atveju reikėtų naudoti įvairius patikrinimus, reikalingus užtikrint apsaugai, o tai tik sulėtintų bendrą spartą. Tuo tarpu administravimo dalyje galima naudoti ir tokius metodus, kaip *querystring* arba informacijos talpinimas *header* dalyje, nes ten saugumas nėra toks svarbus ir šis perdavimo būdas nenaudoja tiek atminties, kiek saugesni metodai. Iš duomenų bazės paimtus duomenis vieno seanso metu galima laikyti *dataset* arba *typed dataset* komponentuose. Antrasis turi visas galimybes kaip ir pirmasis bei kelias papildomas. Tai yra antrasis komponentas labiau pritaikytas saugumo, duomenų vientisumo atžvilgiu, yra labiau automatizuotas, greičiau pritaikomas. Rekomenduojama naudoti būtent šį variantą. Jeigu duomenys yra pastovūs, tai yra perkraunant puslapį visada išlieka ta pati informacija, tai tokiu atveju patartina tokius duomenis pirmo paleidimo metu įdėti į laikiną atmintį. Tad kai puslapis bus paleidžiamas sekantį kartą, tai tereiks duomenis paimti iš laikinos atminties, kas žymiai paspartintų puslapio paleidimą.

Šioje dalyje buvo nagrinėjami visi dalykai, susiję su programavimu. Išryškintas algoritmų tinkamumas, nes dažnai pasitaiko tokių atvejų, kai tą patį veiksmą galima atlikti keliais skirtingais būdais, kurių sparta yra skirtinga. Tad buvo žiūrima, kaip koks algoritmas veikia, ir buvo palikti tik tinkamiausi.

Žvelgiant iš dizaino pusės, viskas turėtų tinkamai ir tiksliai atrodyti, turėtų būti kuo mažiau grafinių elementų. Patartina visą stilių aprašyti ir laikyti css dalyje, nes tai prisideda prie didesnės spartos, nors ir mažesne dalimi.

IV.III.II. Suderinamumas

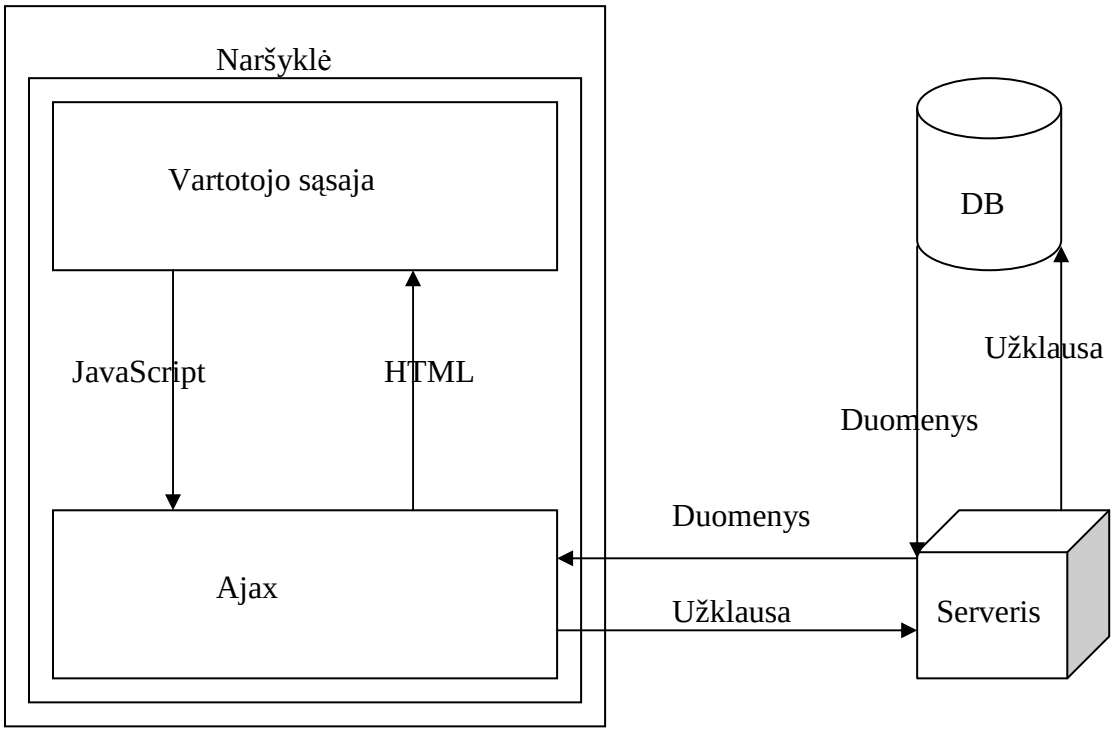
Suderinamumas – tai galimybė naudotis interneto svetaine skirtingose operacinėse sistemose, naršyklėse, kompiuteriuose, neprarandant greičio. Tai yra reikalinga, nes negali iš anksto nuspėt, iš kur ir kas ateis į tavo interneto portalą. Todėl reikia užtikrinti vienodą spartą skirtingose sistemose.

IV.III.III. AJAX

Ši dalis remiasi [PER06], [ESP07] ir [ZMF06].

Turbūt gali iškilti klausimas, kodėl išskirta Ajax iš kitų technologijų, nes daugeliui tai siejasi su interaktyvumu. Tačiau būtent šį interaktyvumą galima panaudoti didesnio greičio pasiekimui.

Technologija buvo pradėta naudoti 2005 metais. Ajax – tai asinchroninis JavaScript ir XML, padedantis pasiekti daug didesnį interaktyvumą. Jo esmė yra ta, kad su serveriu yra keičiamasi duomenimis, o ne visais interneto puslapiais. O tai reiškia, kad vartotojas gali gauti reikiamą informaciją žymiai greičiau. Žemiau pateikiamas veikimo principas:



8pav. Ajax veikimo schema

Kadangi yra palaikomas asinchroniškumas, tai galima atlikti tą patį veiksmą kelis kartus iš eilės arba pereiti prie kito veiksmo, ir nuo to nepasikeis sistemos sparta. Ajax šiuo metu yra sparčiai vystomas, tad bėgant laikui atsiranda vis daugiau pritaikymų, pasiūlymų, idėjų. Tačiau anksčiau buvo sudaryti šios technologijas principai, kurių vertėtų laikytis:

- Kuo mažesnis duomenų srautas, pereinantis per Ajax karkasą. Tik būtiniausi duomenys. Tai leis dirbti sparčiau;
- Nuoseklumo išlaikymas. Tai yra leisti suvokti vartotojui, kas po ko turi sekti, nes su Ajax galima be to apsieiti;

- Nustatytos tvarkos laikymasis. Panaudojamas įprastų, žinomų komponentų, formų. Nes ne visos naujovės gali būti suprastos, priimtinos. Nebent tai taptų labai inovatyviu sprendimu;
- Vengti blaškančių dalykų: animacijos, mirgančios internetinio puslapio dalys ir kita;
- Prieinamumas, suderinamumas. Būtinai reikia patikrinti, ar su Ajax padaryti pakeitimas bus prieinami su visomis interneto naršyklėmis, nes kitaip ne visi vartotojai galės pasinaudoti produktu;
- Vengti pilno puslapio parsisiuntimo vien tiktai su šia technologija. Patartina naudoti dalinį informacijos atnaujinimą;
- Viską padaryti taip, kad vartotojas pats viską suprastų.

Visgi bendras Ajax principas būtų praktiškumas. Su juo galima pagerinti svetainę, galima pasiekti didesnę spartą, tačiau galima gauti ir visiškai priešingą variantą. Tad akcentuojamas testavimas, tikrinimas daugiau nei po vieną kartą.

Kaip realiai veikiančius pavyzdžius rekomenduojama pažiūrėti Google Suggest, Google Mail, Google Maps, Yahoo News, Bitflux Blog.

IV.III.IV LINQ

LINQ (language integrated query) – tai naujas .NET 3.5 komponentas, kuris suteikia galimybę paimti duomenis iš įvairių duomenų aibių, pvz. masyvai, kolekcijos, duomenų bazė. Naudojama sintaksė yra beveik identiška SQL, tačiau tai tik jos supaprastintas variantas, nes LINQ sakiniai iš tiesų yra praplėtimo metodų iškvietimų su lambda išraiškomis grandinė. Lambda išraiška – tai galimybė tam tikrą kodą panaudoti kaip parametrą kitoje funkcijoje.

LINQ palaikomos operacijos:

- Select / SelectMany – duomenų išrinkimas.
- Where – duomenų filtravimas.
- Join / GroupJoin – dviejų duomenų aibių apjungimas pagal tam tikrą nurodomą sutapimą.
- Take / TakeWhile – leidžia nurodyti išrenkamų duomenų kiekį.
- Skip / SkipWhile – panašus principas, kaip prieš tai esančios operacijos, bet šioji nepaima nurodyto kiekio duomenų, o visus likusius paima.
- OfType – duomenų, esančių konkretaus tipo, išrinkimas.

- Concat – dviejų duomenų aibių apjungimas.
- OrderBy / ThenBy – duomenų rūšiavimas.
- Reverse – duomenų rikiuotės pakeitimas priešinga puse.
- GroupBy – duomenų grupavimas.
- Distinct – skirtingų duomenų išrinkimas.
- Union / Intersect / Except – atlieka identiškas operacijos, kaip ir SQL.
- EqualAll – dviejų duomenų aibių palyginimas.
- First / FirstOrDefault / Last / LastOrDefault – paima pirmą ar paskutinį elementą. Jeigu tokių nėra, tai ima imtinai nustatytą reikšmę.
- Single – paima vieną elementą.
- ElementAt – paima vieną elementą pagal nurodytą indeksą.
- Any / All / Contains – tikriną duomenų egzistavimą.
- Count – gražiną elementų kiekį.
- Sum / Min / Max / Average / Aggregate – atlieka atitinkamas operacijas, kaip ir SQL.

LINQ sakiniai nėra vykdomi juos aprašant, nes duomenų paėmimas vyksta tik tada, kai bandoma pasinaudoti pačiais duomenimis. Aprašytas sakinys yra sukompiliuojamas į išraiškų medį, kuris yra išanalizuojamas ir optimizuojamas taip, kad duomenų paėmimo sparta būtų didžiausia. Tai tampa ypač akivaizdu, kai vyksta bendravimas su duomenų baze. LINQ išraiškos yra išverčiamos į analogiškus SQL sakinius ir į duomenų bazę kreipiamasi tik vieną kartą. Šiuo metu yra kuriamas praplėtimas, kuris paskirstys apkrovas procesoriaus branduoliams.

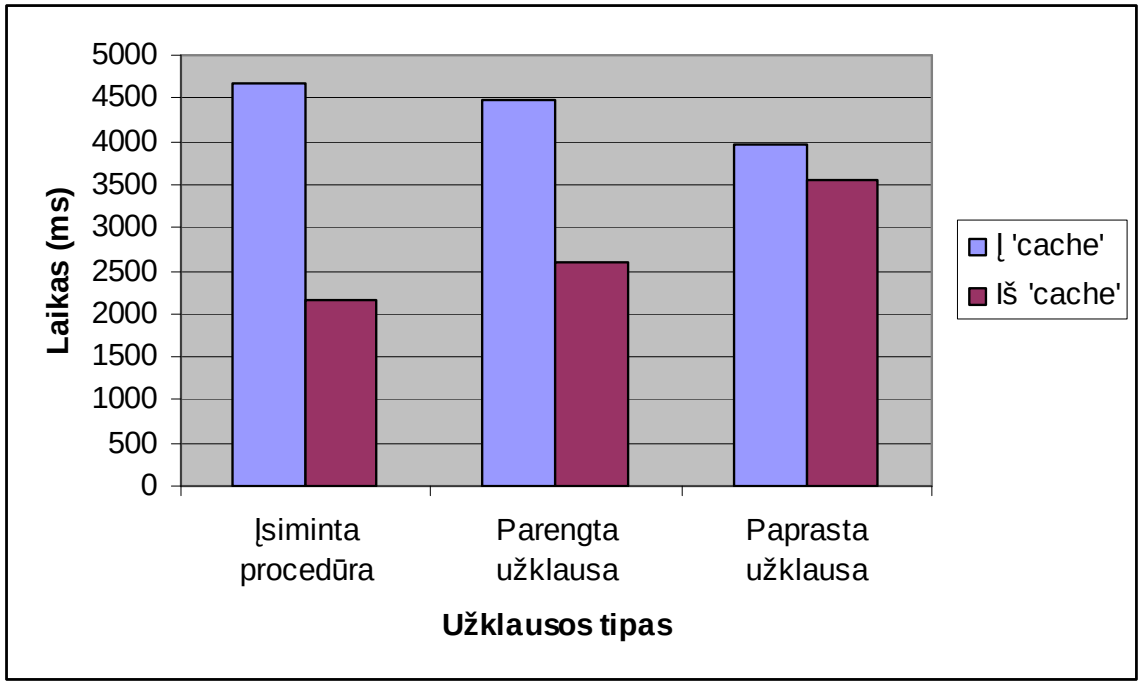
V. Eksperimentinė bazė

V.I. Duomenų bazė

Šioje dalyje visas dėmesys buvo skirtas užklausų optimizavimui, testavimui. Bei duomenų bazės vientisumui, mobilumui ir kokybei. Taip pat buvo tikrinama, koks yra duomenų bazės atsakas į skirtingus užklausų tipus.

Išbandant duomenų bazės vientisumą buvo naudojamas atvirkštinis variantas – simuliuojamos blogos situacijos ir stebimas poveikis. Galiausiai padarytas tikrinimo sluoksnis vartotojos sąsajos dalyje, neleisiantis patekti blogiems duomenims į duomenų bazę. O jei taip ir atsitiktų, tai joje pačioje padarytas FDS konstruktorius, užtikrinantis kokybės proceso vyksmą ir duomenų vientisumą. Mobilumui užtikrinti buvo nagrinėjamos skirtingos duomenų bazės, tokios kaip MySQL, MsSQL, PostgreSQL, Oracle, bei jų naudojami duomenų tipai. Ir tiesiog teliko pasirinkti tuos, kurie naudojami jose visose.

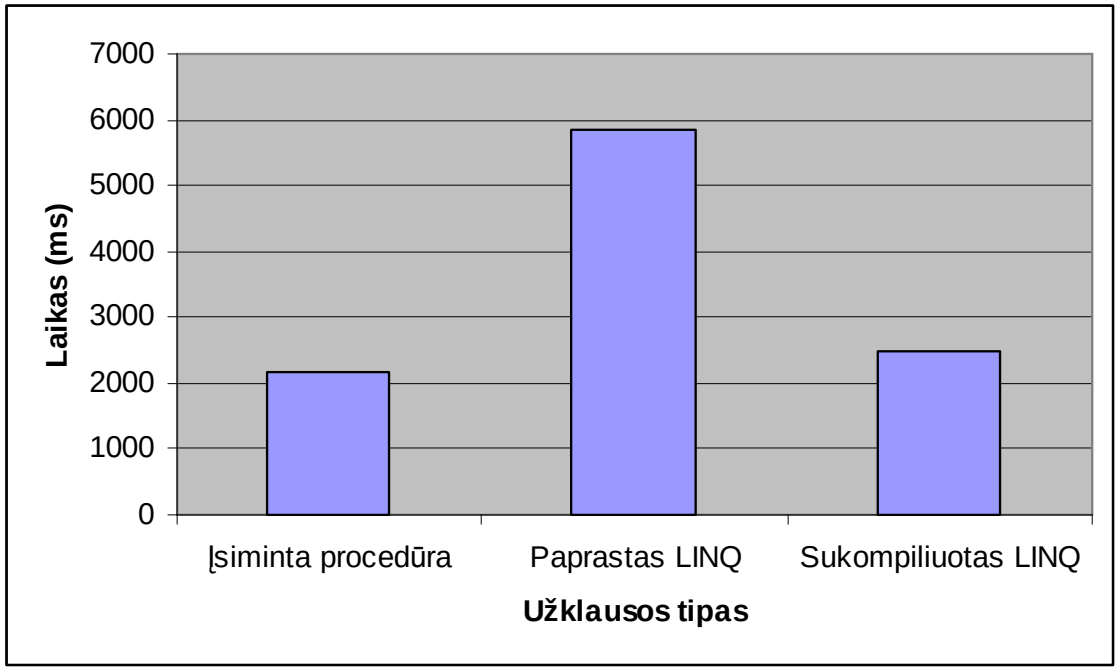
Dirbant su užklausomis buvo tikrinama kiekviena detalė – kiekvieno padaryto pakeitimo poveikis atlikimo spartai. Didžiausia atrasta problema buvo tose užklausose, kurios naudoja rūšiavimą ir paiešką. Jose kiekvienas laukas būdavo tikrinamas kiekvieną kartą su reikiamu duomenų bazės lentelės lauku. Visa tai buvo perdaryta taip, kad būtų atliekamas vos vienas toks tikrinimas. Su mažu duomenų kiekiu šis pokytis yra vos ne vos pastebimas, tačiau dirbant su didelėmis duomenų bazėmis, tai suteikia gan ryškų pasikeitimą į gerąją pusę. Užklausose visur lentelių jungimas atliekamas per indeksuotus laukus arba raktus, nes būtent tokiais atvejais duomenų bazė sugaišta mažiau laiko tinkamų duomenų sujungimui. Visas filtravimas buvo perdaromas taip, kad kuo mažiau veiksmų reikėtų atlikti šioje dalyje. Toliau buvo tiriamas skirtingų užklausų tipų reakcijos laikas. Eksperimentams buvo sukurta duomenų bazė su 4 lentelėmis. Šiose lentelėse atsitiktiniu būdu sugeneruota po pora milijonų įrašų. Užklausa apjungdavo visas lenteles ir filtruodavo duomenis.



9pav. Skirtingų užklausų tipų testo rezultatai

Tad buvo išbandoma paprastos užklausos, parengtosios užklausos ir įsimintinos procedūros. Vertikaliame stulpelyje pateiktas bendras reakcijos laikas. Mėlynas stulpelis žymi patį pirmą iškvietimo kartą, kai nauja užklausa yra įtraukiama į atsargos atmintį. Rausvas stulpelis žymi visus kitus šios užklausos panaudojimo kartus (jų laikų vidurkis). Kaip matyti pirmą kartą visos užklausos veikia lėtokai, tačiau vėlesni naudojimai yra žymiai geresni, nes jau yra atsargos atmintyje. Geriausius rezultatus, kaip ir prognozuoja teorija, pasiekė įsimintinos procedūros.

Sekantis bandymas sekė su LINQ tipo užklausomis. Pastarosios gali būti dviejų tipų: paprastos ir iš anksto sukompiliuotos. Apie paprastas yra pateiktas aprašas teorinėje darbo dalyje. Sukompiliuotos nuo paprastų skiriasi tuo, kad joms yra sukuriamos atskiros lambda išraiškos.



10pav. LINQ testo rezultatai

Jeigu imant patogumą, tai paprastas LINQ be abejo atrodytų geriausiai. Tačiau matant spartos rezultatus, kurie yra tikrai prasti, iškart yra atmetama galimybė šios technologijos panaudojimui tiriamame objekte bent jau tol, kol vyksta darbas su mažais duomenų kiekiais. Visgi lieka variantas – sukompiliuotas LINQ, bet tam reiktų rašyti atskiras lambda išraiškas, kurios savo sudėtingumu žymiai lenkia įsimintų procedūrų rašymą. Be to, iškilo vienas toks įdomus pastebėjimas beeksperimentuojant su LINQ. Jeigu dažnai perkompiliuoji tą pačią užklausą, tai sparta pradeda kristi.

Nr.	Kompiliavimų skaičius	Užklausų skaičius	Laikas (ms)
1.	1	5000	2472
2.	2	2500	2473
3.	5	1000	2503
4.	100	100	2582
5.	1000	5	3597
6.	2500	2	4963
7.	5000	1	7896

1 lentelė. Sukompiliuoto LINQ testas

Pirma eilutė reiškia, kad užklausa buvo sukompiliuota vieną kartą. Po to įvykdyta 5000 kartų. Laikas pateikiamas, išvedus vidurkį. Antra eilutė reiškia, kad užklausa buvo

sukompiliuota vieną kartą, įvykdyta 2500 kartų, po to perkompiliuota ir vėl įvykdyta. Ir taip toliau. Taigi geriausias atvejis yra tada, kai kompiliavimas vykdomas vieną kartą. O kai tai vyksta kiekvieną kartą, kai yra paleidžiama užklausa, tai sparta yra netgi blogesnė nei paprasto LINQ.

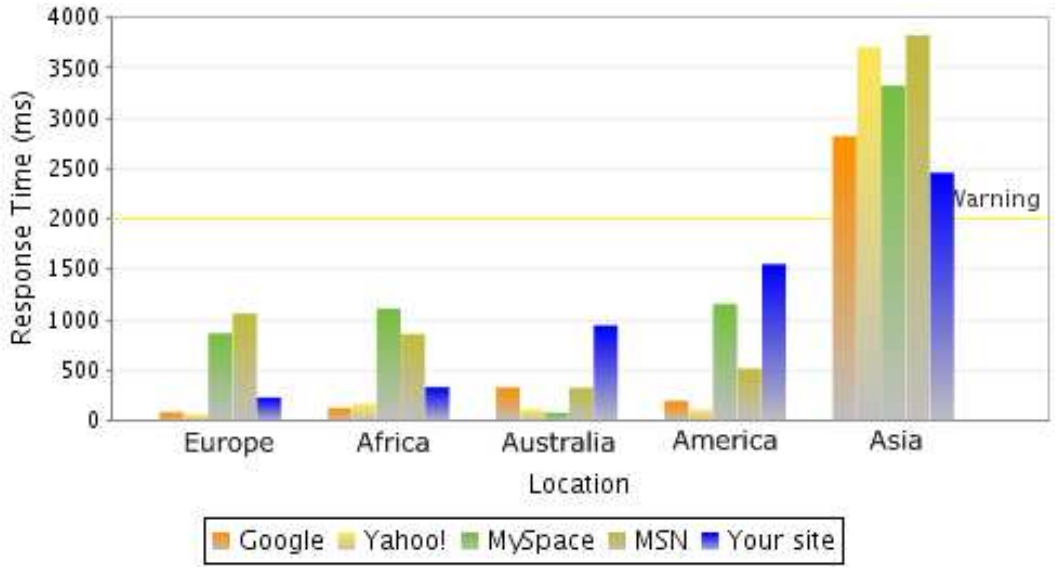
V.II. Interneto svetainė

Šiuo metu internete patalpinta naujausia tiriamo objekto versija, kurioje yra atlikti visi suplanuoti eksperimentavimo, testavimo darbai. Kad būtų galima atlikti palyginimą, kaip pasikeitė sparta tarp versijų, buvo išmatuota senosios versijos sparta. Rezultatai pateikti antroje lentelėje.

	Be kreipimosi į duomenų bazę	**Su kreipimusi į duomenų bazę**
14.4K	~8s	~10,2s
28.8K	~4,3s	~5,4s
33.6K	~3,8s	~4,7s
56K	~2,5s	~3,1s
ISDN 128K	~1,2s	~1,4s
T1 1.44Mbps	~0,6s	~0,7s

2 lentelė. Tiriamo objekto pradinės spartos matavimo rezultatai

Dar radau tokią įdomią interneto svetainę YouMonitor.us, kuri išmatuoja norimo portalo greitį iš visų pasaulio kontinentų. Tokio tipo svetainių yra ir daugiau, bet dažniausiai jos yra mokamos. Ši taip pat ne išimtis, tačiau suteikia galimybę 30 dienų naudotis nemokama versija, kurioje yra mažiau galimybių nei mokamoje. Bei palyginimui pateikia žinomų interneto svetainių rezultatus, kas tikrai yra įdomu. Rezultatai pateikiami imant vidurkius bendro naudojimosi visais puslapiais.



11pav. YouMonitor.us spartos matavimo rezultatai

Nežinau, su kokios spartos linija šis greitis yra matuojamas, tačiau iš rezultatų gali susidaryti įspūdis, kad imamos skirtingos linijos, o rezultatas pateikiamas bendras. Kad ir kaip ten bebūtų, tačiau galima pamatyti tam tikrus dėsningumus. Iš Azijos tiriamą objektą galima pasiekti netgi greičiau nei tokius populiarius ir žinomus portalus, kurie yra pateikti, kaip pavyzdys, nors ir visi yra perkopę dviejų sekundžių ribą. Tačiau žymiai geresni rezultatai yra iš kitų pasaulio dalių, iš kurių Australijoj ir Amerikoj yra prastesni rezultatai, o, kaip ne keista, Afrikoj ir, be abejo, Europoj sparta yra labai gera.

Apie įvykdytus pakeitimus ir eksperimentus:

Iš pradžių buvo visiškai pakeista interneto svetainės struktūra. Atskirtos visos dalys viena nuo kitos, kurias tik buvo galima atskirt. Tas buvo padaryta tam, kad būtų galima lengviau atlikti bandymus atskiroms dalims. Tad šiuo metu atskirai egzistuoja JavaScript, CSS, duomenų bazės lygmuo, Ajax, elektroninių laiškų siuntimas, duomenų tikrinimas, tekstai, papildomi komponentai. JavaScript ir CSS atskyriau dėl dviejų priežasčių:

1. mažiau teksto yra siunčiama ir matoma vartotojui;
2. šis būdas yra pritaikytas greitesniam veikimui.

Duomenų bazės lygmeniui sukūriau visiškai atskirą projektą, kurį pritaikiau įvairiapusiam naudojimui. Tai yra galima dirbti tiek su paprastomis užklausomis, tiek su parengtomis užklausomis, tiek su įsimintinomis procedūromis. Ko man ir reikėjo bendravimo su duomenų baze eksperimentams atlikti.

Bene daugiausia laiko užėmė Ajax integravimas. Šiuo metu tai yra absoliučiai visuose tiriamo objekto svetainės puslapiuose. Kodėl taip padariau:

- nereikia perkrauti (parsisiųsti iš serverio) viso puslapio iš naujo;
- panaudotas puslapiavime, rūšiavime, filtravime, paieškoje, duomenų tikrinime, darbe su duomenimis;
- panaudotas vidinių puslapių rodyme, tai yra tarkim nueinate į forumą, ten norite perskaityti žinutes, sukurti naują temą ir panašiai. Tad visai tai yra atliekama su Ajax, kas žymiai pagreitina darbą.

Kitas žingsnis buvo išbandyti duomenų (teksto) išvedimą vartotojui. Pagrinde buvo tiriama atsargos atmintis (cache). Išbandyta įvairių dydžių teksto blokai, bet tiksliai nenustatyta, nuo kokio dydžio yra geriausia dėt duomenis į šią atmintį. Nes daug taip pat priklauso ir nuo kompiuterinės technikos. Bet esminis dalykas yra tas, kad mažą žodžių kiekį laikyti atsargos atmintyje nėra verta, nes dažnai kreipimasis į ją gali užtrukti ilgiau nei tų žodžių paprastas paėmimas. Tad nuspręsta į šią atmintį dėti tik tą tekstą, kuriame yra daugiau nei 10 žodžių. Gerėjant technikai šis skaičius tik didėtų. O dėti į atsargos atmintį reikia todėl, kad užeinant antrą kartą ar daugiau į tą patį interneto puslapį nereiktų dar kartą siųstis tų pačių duomenų. Tad sutaupomas laikas, ko ir reikia.

Po visų šių pakeitimų buvo pamatuota bendra sparta.

	Be kreipimosi į duomenų bazę	**Su kreipimusi į duomenų bazę**
14.4K	~2,7s	~3,4s
28.8K	~1,4s	~1,8s
33.6K	~1,3s	~1,6s
56K	~0,8s	~1,0s
ISDN 128K	~0,4s	~0,5s
T1 1.44Mbps	~0,2s	~0,3s

3 lentelė. Tiriamo objekto galutinės spartos matavimo rezultatai

Gauti rezultatai apytiksliai yra tris kartus geresni negu kad buvo prieš pradedant spartos kėlimo darbus. Retkarčiais šis santykis kinta į vieną ar kitą pusę. Bet dažniausiai tai išorinių aplinkybių poveikis, kuris nėra pastovus.

V.III. Pritaikymai

Šioje dalyje paminėti dalykai, kurie buvo pritaikyti tiriamam objektui, bei šiuo metu ten naudojami. Priminsiu, kad tiriama ir testuota buvo daugiau negu pritaikyta.

Duomenų bazė:

- Indeksai;
- Pagrindiniai, išoriniai raktai;
- Įsimintos procedūros;
- Optimizuotos užklausos;
- Foundation data store konstruktorius;
- Universalių duomenų tipų panaudojimas.

Interneto svetainė:

- Ajax;
- Atsargos atmintis;
- Padidintas saugumo lygis, atsisakyta QueryString;
- Užtikrintas duomenų vientisumas;
- Kontroleriai: typed dataset, repeater, gridview;
- Bendravimas su duomenų baze vyksta per sukurtą universalų įrankį.

VI. Literatūros analizė

[PUR00] yra vienintelė analizuota knyga iš duomenų bazės pusės. Galbūt gali pasirodyti, kad to yra per mažai, tačiau tai nėra paprasta vieno autoriaus parašyta knyga. Ji yra sudaryta iš daugybės straipsnių, kurie buvo parašyti skirtingų autorių. Šie straipsniai yra surūšiuoti, apjungti, sudaryta puiki struktūra. Taigi kadangi autorių yra daug, tai ir įvairių minčių yra daug, netgi skirtingai galvojama apie tuos pačius dalykus. Tai iš tikro šiek tiek klaidina, tačiau juk galima pabandyt visus siūlomus būdus ir atrinkti tuos, kurie geriausiai tinka tiriamam objektui. Tad ir tyrime atsispindės tai, nes kai kurie dalykai, pvz. indeksavimas, bus nagrinėjami iš skirtingų pusių.

[TL01] yra aprašytas kompiliatorius, jo veikimo principai, galimybių panaudojimas, kurios nepasiekiamos per Visual Studio paketą. Tai suteikė bendrą supratimą, ką reikėtų programuojant naudot, kad būtų galima pasiekt maksimalų veikimo greitį.

[TLN01] yra paaiškintas reikalingų programų veikimas bei derinimas. Čia aiškiai aprašyta, kaip sukurtas Csharp (tai tam tikras Java ir C/C++ mišinys), kaip juo naudotis, tačiau esmė yra labai panaši į kitų kalbų, kurias mokė universitete. Kita šioje knygoje išdėstyta technologija yra sparčiai populiarėjantis ASP.NET, kurio universitete neteko mokytis, bet šioje e-knygoje yra puikiai paaiškinta teorinė dalis.

[TSK01] aprašytas XML technologijų susiejimas ir panaudojimas su ASP bei CSharp. Bet kadangi tai buvo sena e-knyga, tai į ją per daug nesigilinta, nes dabar yra efektyvesnių XML panaudojimo būdų, kurie aprašyti naujesnėse knygose.

[DAV02] labai panaši į [TLN01]. Skirtumas tik tas, kad tai naujesnė e-knyga ir joje paaiškinti per vienerius metus atsiradę patobulinimai.

[WAL02] daugiausia dėmesio skiriama praktiniam programavimo kalbų panaudojimui. Dažnai vien tik išmanyti teoriją neužtenka, nes negali tiksliai žinoti, kaip tai pritaikyti, o čia gerai matosi kaip viskas atrodo realiai praktikoje. Kodo pavyzdžiai, veikiančios programos leidžia pradėt pačiam programuot, pritaikyti teorines žinias ir stebėt, kurie dalykai veikia greičiau, kurie lėčiau.

[SUS03] paskirtis tokia pati, kaip ir prieš tai minėtos e-knygos. Tačiau joje minimos naujesnės technologijos, matoma išsivystymo lygis, tempas.

[RUV04] e-knygoje ne tik aprašytos patobulėjusios programavimo kalbos, bet jos ten pat yra ir palyginamos. Pateikiami jų išeities kodai, veikimo rezultatai. Prieinamos išvados, kad su visom programavimo kalbom galima pasiekti tą patį rezultatą, panaudojant tuos pačius resursus, tad geriausia išskirta nebuvo.

[LIB05] buvo išleista prieš pora metų. Joje atsispindi technologijų tendencijos, į kurią pusę dabar vyksta tobulinimai, ko labiausiai reikia klientams. Pagrinde magistrinio darbo užduotims sukurti iš programavimo pusės idėjų semtasi iš šios e-knygos.

[BE06] pateiktas sql pradžiamokslis. Daugiausia dėmesio skirta užklausų sudarymui, optimizavimui. Taip pat aprašytas indeksų bei suderinamumo suvaržymų panaudojimas.

[ABC07] pateikta pasiūlymai, kaip panaudoti stilius, skriptus, kad interneto svetainė įgytų tiek gražesnę išvaizdą, tiek didesnę veikimo spartą.

[PER06] duota tikslūs, praktiški sprendimai, kaip panaudoti kuo daugiau Ajax galimybių. Kiekvienam skyriui nurodytas protingas būdas įvykdyti konkrečiai užduočiai, kas leidžia sutaupyti laiko, jeigu tarkim reiktų kažko tokio, ko dar nežinai. Tą rasi šioje knygoje. Taip pat nurodyti patarimai, kaip turi atrodyti JavaScript kodas, kad interneto svetainė būtų greitesnė, patikimesnė.

[ESP07] knyga apibendrina būtent tą Ajax karkasą, kurį aš naudoju kartu su tiriamu objektu. Joje yra viskas nuo pradžiamokslio iki sudėtingų dalykų, ką ir bandžiau pritaikyti.

[ZMF06] parašyta patyrusiems interneto svetainių kūrėjams. Ši autoritetinga knyga parodo, kaip suderinti išbandytas ir patikimas technologijas su Ajax, kad būtų galima sukurti modernias ir greitas vartotojų sąsajas.

[CH07] suteikia galimybę išmokti rašyti SQL kodą su c-sharp kalba, kuris būtų kompiliuojamas pačiame SQL serveryje. Tai labiau skirta tiems asmenims, kurie geriau nusimano vartotojo sąsajos kūrime, tačiau turi pakankamai žinių, kad galėtų dirbti su duomenų bazėmis, prie kurių priskaičiuoju aš ir pats save. Taip pat kreipiamas dėmesys į taisyklingą programavimą.

Iš http://msdn.microsoft.com, MSDN Library for Visual Studio 2005, http://www.asp.net semtasi žinių apie optimalų programavimo kalbų panaudojimą, kad būtų galima panaudot kuo mažiau kodo konkrečiam dalykui aprašyti, tačiau nepaaukoti greičio. Šiose vietose pateikti naujausi išeities kodų pavyzdžiai, leidžiantys palyginti, kaip tą patį dalyką panaudoji tu ir kaip jį panaudoja kažkas kitas. Taip žymiai greičiau gali pastebėti įvairias algoritmų spragas, kurios gali sulėtini bendrą spartą.

Skaitant šias e-knygas, pilnai suvokta, kaip reiktų spręsti iškeltas užduotis, vykdant magistrinį darbą.

VII. Išvados. Rezultatai

1. Išanalizuoti turimi literatūros šaltiniai.
2. Sukonkretinta problema - sparta.
3. Sudarytas užduočių planas pagal literatūroje rastas hipotezes.
4. Sudaryta teorinė bazė.
5. Sudaryta eksperimentinė bazė.
6. Remiantis teorija ir eksperimentais gauti rezultatai:
    - Sparčiausia duomenų bazių valdymo sistema - InterSystems Cache;
    - Maksimalus duomenų bazių mobilumas pasiekiamas Windows operacinėse sistemose;
    - Duomenų bazėse naudojamas indeksavimas nėra pilnai išvystytas;
    - Duomenų bazių padalinimai pamažu pradedami naudoti visur;
    - Kiekviena skirtinga užklausa yra unikali, todėl negalima aklai pasirinkti vieno jos sudarymo būdo, tikintis, kad sparta bus geriausia;
    - Nevientisa duomenų bazė kenkia spartai;
    - Naudojant .NET technologijas, geriausiai duomenų tinkamumą užtikrina tipizuotos duomenų aibės; sparčiausią atvaizdavimą, atsižvelgiant į papildomai teikiamas galimybes, leidžia pasiekti listview komponentas;
    - Atsargos atminties ir Ajax bendras panaudojimas duomenų atvaizdavimui iš duomenų bazės žymiai pakelia spartą, bet yra rizikingas. Tai galėtų būti atskiro darbo tema tokiam moduliui sukurti;
    - LINQ nėra tinkamas spartos atžvilgiu, tačiau labai patogus.
7. Bendra tiriamo objekto sparta pagerėjo tris kartus.

VIII. Literatūros sąrašas

[PUR00] S. Purba. High-performance web databases: design, development and deployment. 2000

[DAV02] H. Davis. Visual C#.NET Programming. 2002

[LIB05] J. Liberty & Associates. Building .NET Aplications with C#. URL: http://www.oreilly.com. 2005

[RUV04] Z. Ruvalcaba. Build Your Own ASP.NET Website Using C# & VB.NET. URL: http://www.sitepoint.com. 2004

[SUS03] D. Sussman. Beginning Dynamic Websites With ASP.NET Web Matrix. 2003

[TLN01] A. Turtschi, W. Lee and S. Nandu. C#.NET Web Developers Guide. 2001

[TL01] T. Thai and H. Lam. .NET Framework Essentials. 2001

[TSK01] D. Tidwell, J. Snell and P. Kulchenko. Programming Web Services With SOAP. 2001

[WAL02] S. Walther. ASP.NET Kick Start. 2002

[BE06] S.S. Bagui, R.W. Earp. Learning SQL on SQL Server 2005. 2006

[ABC07] C. Adams, M. Boulton, A. Clarke, S. Collison, J. Croft, D. Featherstone, I. Lloyd, E. Marcotte, D. Rubin, R. Weychert. Web Standarts Creativity: Innovations in Web Design with XHTML, CSS and DOM scripting. 2007

[PER06] B.W. Perry. Ajax Hacks – Tips and Tools For Creating Responsive Websites. 2006

[ESP07] D. Esposito. Introducing Microsoft ASP.NET AJAX. 2007

[ZMF06] N.C. Zakas, J. McPeakand, J. Fawcett. Professional Ajax. 2006

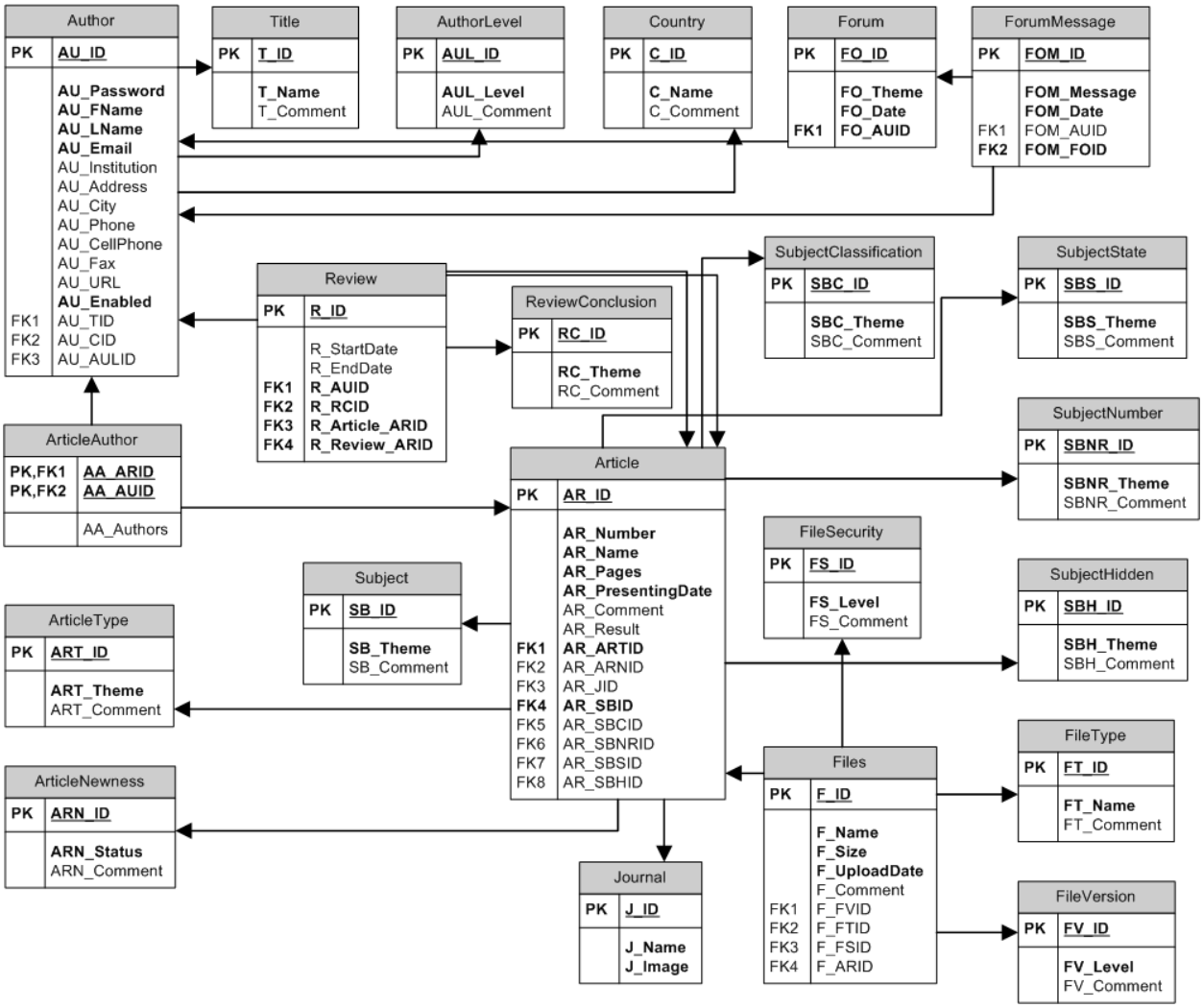
[CH07] D. Comingore, D. Hinson. Professional SQL Server 2005 CLR Programming With Stored Procedures, Functions, Triggers, Aggregates and Types. 2007

http://msdn.microsoft.com

MSDN Library for Visual Studio 2005

http://www.asp.net

Priedas Nr.1 - Duomenų bazės schema



Priedas Nr.2 – Tikrintos duomenų bazės

RDBMS	Įmonė	Pirmoji išleidimo data	Naujausia versija
4th Dimension	4D s.a.s	1984	v11 SQL
ADABAS	Software AG	1970	
Adaptive Server Enterprise	Sybase	1987	15.0
Advantage Database Server	Sybase	1992	8.1
Apache Derby	Apache	2004	10.3.1.4
DB2	IBM	1982	9.5
DBISAM	Elevate Software		4.25
Datawasp	Significant Data Systems	04.2008	1.0.1
ElevateDB	Elevate Software		1.01
FileMaker	FileMaker	1984	9
Firebird	Firebird project	25.06.2000	2.1.0
Informix	IBM	1985	11.10
HSQLDB	HSQL Development Group	2001	1.8.0
H2	H2 Software	2005	1.0
Ingres	Ingres Corp.	1974	Ingres 2006 r2 9.1.0
InterBase	CodeGear	1985	2007
MaxDB	SAP AG		7.06
Microsoft Access	Microsoft	1992	12 (2007)

Microsoft Visual Foxpro	Microsoft		9 (2005)
Microsoft SQL Server	Microsoft	1989	9.00.3042 (2005 SP2)
MonetDB	The MonetDB Developer Team	2004	4.16 (02. 2007)
MySQL	Sun Microsystems	11.1996	5.0.51
HP NonStop SQL	Hewlett-Packard	1987	SQL MX 2.0
Oracle	Oracle Corporation	11.1979	11g Release 1 (09.2007)
Oracle Rdb	Oracle Corporation	1984	02.Lie
OpenEdge	Progress Software Corporation	1984	10.1B
OpenLink Virtuoso	OpenLink Software	1998	5.0.5 (01.2008)
Pervasive PSQL	Pervasive Software		9
Polyhedra DBMS	ENEA AB	1993	7.01
PostgreSQL	PostgreSQL Global Development Group	06.1989	8.3.1 (18.03.2008)
Pyrrho DBMS	University of Paisley	11.2005	0.5
ScimoreDB	Scimore	2005	2.05
SmallSQL	SmallSQL	16.04.2005	0.19
SQL Anywhere	Sybase	1992	10.0
SQLite	D. Richard Hipp	17.08.2000	3.5.7 (17.03.2008)
Teradata	Teradata	1984	V12

Valentina	Paradigma Software	02.1998	3.0.1

Priedas Nr.3 – Duomenų bazių palaikomos operacinės sistemos

RDBMS	Windows	Mac OS X	Linux	BSD	UNIX
4th Dimension	Taip	Taip	Ne	Ne	Ne
ADABAS	Taip	Ne	Taip	Ne	Taip
Adaptive Server Enterprise	Taip	Taip	Taip	Taip	Taip
Advantage Database Server	Taip	Ne	Taip	Ne	Ne
Apache Derby	Taip	Taip	Taip	Taip	Taip
Datawasp	Taip	Ne	Ne	Ne	Ne
DB2	Taip	Ne	Taip	Ne	Taip
Firebird	Taip	Taip	Taip	Taip	Taip
HSQLDB	Taip	Taip	Taip	Taip	Taip
H2	Taip	Taip	Taip	Taip	Taip
FileMaker	Taip	Taip	Ne	Ne	Ne
Informix	Taip	Taip	Taip	Taip	Taip
Ingres	Taip	Taip	Taip	Taip	Taip
InterBase	Taip	Taip	Taip	Ne	Taip
MaxDB	Taip	Ne	Taip	Ne	Taip
Microsoft Access	Taip	Ne	Ne	Ne	Ne
Microsoft Visual Foxpro	Taip	Ne	Ne	Ne	Ne
Microsoft SQL Server	Taip	Ne	Ne	Ne	Ne
MonetDB	Taip	Taip	Taip	Ne	Taip
MySQL	Taip	Taip	Taip	Taip	Taip
Oracle	Taip	Taip	Taip	Ne	Taip
Oracle Rdb	Ne	Ne	Ne	Ne	Ne
OpenEdge	Taip	Ne	Taip	Ne	Taip
OpenLink Virtuoso	Taip	Taip	Taip	Taip	Taip
Polyhedra DBMS	Taip	Ne	Taip	Ne	Taip

PostgreSQL	Taip	Taip	Taip	Taip	Taip
Pyrrho DBMS	Taip	Ne	Taip	Ne	Ne
ScimoreDB	Taip	Ne	Ne	Ne	Ne
SmallSQL	Taip	Taip	Taip	Taip	Taip
SQL Anywhere	Taip	Taip	Taip	Ne	Taip
SQLite	Taip	Taip	Taip	Taip	Taip
Teradata	Taip	Ne	Taip	Ne	Taip
Valentina	Taip	Taip	Taip	Ne	Ne

Priedas Nr.4 – Duomenų bazių pagrindinių savybių palaikymas

RDBMS	ACID	Referential integrity	Transactions	Unicode	Interface
4th Dimension	Ne	Ne	Ne	Ne	Ne
ADABAS	Ne	Ne	Ne	Ne	Ne
Adaptive Server Enterprise	Taip	Taip	Taip	Taip	Ne
Advantage Database Server	Taip	Taip	Taip	Ne	Taip
Apache Derby	Taip	Taip	Taip	Taip	Taip
Datawasp	Ne	Taip	Taip	Taip	Taip
DB2	Taip	Taip	Taip	Taip	Taip
Firebird	Taip	Taip	Taip	Taip	Taip
HSQLDB	Taip	Taip	Taip	Taip	Taip
H2	Taip	Taip	Taip	Taip	Taip
Informix	Taip	Taip	Taip	Taip	Ne
Ingres	Taip	Taip	Taip	Taip	Taip
InterBase	Taip	Taip	Taip	Taip	Taip
MaxDB	Taip	Taip	Taip	Taip	Taip
Microsoft Access	Ne	Taip	Taip	Taip	Taip
Microsoft Visual Foxpro	Ne	Taip	Taip	Ne	Taip

Microsoft SQL Server	Taip	Taip	Taip	Taip	Taip
MonetDB	Taip	Taip	Taip	Taip	Ne
MySQL	Taip	Taip	Taip	Taip	Taip
Oracle	Taip	Taip	Taip	Taip	Taip
Oracle Rdb	Taip	Taip	Taip	Taip	Ne
OpenEdge	Taip	Ne	Taip	Taip	Ne
OpenLink Virtuoso	Taip	Taip	Taip	Taip	Ne
Polyhedra DBMS	Taip	Taip	Taip	Taip	Taip
PostgreSQL	Taip	Taip	Taip	Taip	Taip
Pyrrho DBMS	Taip	Taip	Taip	Taip	Ne
ScimoreDB	Taip	Taip	Taip	Taip	Taip
SQL Anywhere	Taip	Taip	Taip	Taip	Ne
SQLite	Taip	Ne	Taip	Taip	Taip
Teradata	Taip	Taip	Taip	Taip	Taip
Valentina	Ne	Taip	Ne	Taip	Ne

Priedas Nr.5 – Duomenų bazių pseudo lentelių palaikymas

RDBMS	Laikinos lentelės	Materializuotos virtualios lentelės
4th Dimension	Ne	Ne
ADABAS	Ne	Ne
Adaptive Server Enterprise	Taip	Ne
Advantage Database Server	Taip	Ne
Apache Derby	Taip	Ne
Datawasp	Taip	Taip
DB2	Taip	Taip
Firebird	Taip	Ne

HSQLDB	Taip	Ne
H2	Taip	Ne
Informix	Taip	Taip
Ingres	Taip	Ne
InterBase	Taip	Ne
MaxDB	Taip	Ne
Microsoft Access	Taip	Ne
Microsoft Visual Foxpro	Taip	Taip
Microsoft SQL Server	Taip	Taip
MonetDB	Taip	Ne
MySQL	Taip	Ne
Oracle	Taip	Taip
Oracle Rdb	Taip	Taip
OpenEdge	Taip	Ne
OpenLink Virtuoso	Taip	Taip
Polyhedra DBMS	Ne	Ne
PostgreSQL	Taip	Ne
Pyrrho DBMS	Ne	Ne
SQL Anywhere	Taip	Taip
ScimoreDB	Ne	Ne
SQLite	Taip	Ne
Teradata	Taip	Taip
Valentina	Taip	Ne

Priedas Nr.6 – Duomenų bazių indeksų palaikymas, išskyrus tradicinius B-/B+ medžius

RDBMS	R-/R+ tree	Hash	Expression	Partial	Reverse	Bitmap	GiST	GIN
4th Dimension	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
ADABAS	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne

Adaptive Server Enterprise	Ne	Ne	Ne	Ne	Taip	Ne	Ne	Ne
Apache Derby	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
DB2	Ne	Ne	Taip	Ne	Taip	Taip	Ne	Ne
Firebird	Ne	Ne	Taip	Ne	Taip	Ne	Ne	Ne
HSQLDB	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
H2	Ne	Taip	Ne	Ne	Ne	Ne	Ne	Ne
Informix	Taip	Taip	Taip	Taip	Taip	Taip	Ne	Ne
Ingres	Taip	Taip	Taip	Ne	Ne	Taip	Ne	Ne
InterBase	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
MaxDB	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
Microsoft Access	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
Microsoft Visual Foxpro	Ne	Ne	Taip	Taip	Taip	Taip	Ne	Ne
Microsoft SQL Server	Ne	Taip	Taip	Taip	Taip	Ne	Ne	Ne
MonetDB	Ne	Taip	Ne	Ne	Ne	Ne	Ne	Ne
MySQL	Taip	Taip	Ne	Ne	Ne	Ne	Ne	Ne
Oracle	Taip	Taip	Taip	Taip	Taip	Taip	Ne	Ne
Oracle Rdb	Ne	Taip	Ne	Ne	Ne	Ne	Ne	Ne
OpenLink Virtuoso	Taip	Taip	Taip	Ne	Ne	Taip	Ne	Ne
Polyhedra DBMS	Ne	Taip	Ne	Ne	Ne	Ne	Ne	Ne
PostgreSQL	Taip	Taip	Taip	Taip	Taip	Taip	Taip	Taip
Pyrrho DBMS	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
ScimoreDB	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
SQL Anywhere	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
SQLite	Ne	Ne	Ne	Ne	Taip	Ne	Ne	Ne
Teradata	Ne	Taip	Taip	Taip	Ne	Taip	Ne	Ne
Valentina	Ne	Ne	Taip	Taip	Taip	Taip	Ne	Ne

Priedas Nr.7 – Duomenų bazių galimybės I

RDBMS	UNION	INNER JOIN	OUTER JOIN	INNER SELECT	MERGE	Blob, Clob
4th Dimension	Ne	Ne	Ne	Ne	Ne	Ne
ADABAS	Ne	Ne	Ne	Ne	Ne	Ne
Adaptive Server Enterprise	Taip	Taip	Taip	Taip	Ne	Taip
Advantage Database Server	Taip	Taip	Taip	Taip	Taip	Taip
Apache Derby	Taip	Taip	Taip	Ne	Ne	Taip
Datawasp	Taip	Taip	Taip	Taip	Taip	Taip
DB2	Taip	Taip	Taip	Taip	Taip	Taip
Firebird	Taip	Taip	Taip	Taip	Taip	Taip
HSQLDB	Taip	Taip	Taip	Ne	Ne	Ne
H2	Taip	Taip	Taip	Ne	Ne	Taip
Informix	Taip	Taip	Taip	Taip	Taip	Taip
Ingres	Taip	Taip	Taip	Taip	Taip	Taip
InterBase	Taip	Taip	Taip	Ne	Ne	Taip
MaxDB	Taip	Taip	Taip	Taip	Ne	Taip
Microsoft Access	Taip	Taip	Taip	Ne	Ne	Taip
Microsoft Visual Foxpro	Taip	Taip	Taip	Taip	Ne	Taip
Microsoft SQL Server	Taip	Taip	Taip	Taip	Taip	Taip
MonetDB	Ne	Ne	Ne	Ne	Ne	Ne
MySQL	Taip	Taip	Taip	Taip	Taip	Taip
Oracle	Taip	Taip	Taip	Taip	Taip	Taip
Oracle Rdb	Taip	Taip	Taip	Taip	Taip	Taip
OpenEdge	Taip	Taip	Taip	Ne	Ne	Taip
OpenLink Virtuoso	Taip	Taip	Taip	Taip	Ne	Taip
Polyhedra DBMS	Taip	Taip	Ne	Ne	Ne	Taip
PostgreSQL	Taip	Taip	Taip	Taip	Taip	Taip
Pyrrho DBMS	Ne	Ne	Ne	Ne	Ne	Ne
ScimoreDB	Taip	Taip	Taip	Taip	Taip	Taip

SmallSQL	Ne	Ne	Ne	Ne	Ne	Ne
SQL Anywhere	Taip	Taip	Taip	Taip	Ne	Ne
SQLite	Taip	Taip	Taip	Ne	Ne	Taip
Teradata	Taip	Taip	Taip	Taip	Taip	Taip
Valentina	Taip	Taip	Taip	Taip	Taip	Taip

Priedas Nr.8 – Duomenų bazių galimybės II

RDBMS	Data domain	Cursor	Trigger	Function	Procedure	External routine
4th Dimension	Ne	Ne	Ne	Ne	Ne	Ne
ADABAS	Ne	Ne	Ne	Taip	Taip	Ne
Adaptive Server Enterprise	Taip	Taip	Taip	Taip	Taip	Taip
Advantage Database Server	Taip	Taip	Taip	Taip	Taip	Taip
Apache Derby	Ne	Taip	Taip	Taip	Taip	Taip
DB2	Ne	Taip	Taip	Taip	Taip	Taip
Firebird	Taip	Taip	Taip	Taip	Taip	Taip
HSQLDB	Ne	Ne	Taip	Taip	Taip	Taip
H2	Taip	Ne	Taip	Taip	Taip	Taip
Informix	Ne	Taip	Taip	Taip	Taip	Taip
Ingres	Taip	Taip	Taip	Taip	Taip	Taip
InterBase	Taip	Taip	Taip	Taip	Taip	Taip
MaxDB	Taip	Taip	Taip	Taip	Taip	Ne
Microsoft Access	Ne	Ne	Ne	Ne	Ne	Taip
Microsoft Visual Foxpro	Ne	Taip	Taip	Taip	Taip	Taip
Microsoft SQL Server	Taip	Taip	Taip	Taip	Taip	Taip
MonetDB	Ne	Ne	Taip	Taip	Taip	Taip
MySQL	Ne	Taip	Taip	Taip	Taip	Taip
OpenEdge	Taip	Taip	Taip	Taip	Taip	Taip

Oracle	Taip	Taip	Taip	Taip	Taip	Taip
Oracle Rdb	Taip	Taip	Taip	Taip	Taip	Taip
OpenLink Virtuoso	Taip	Taip	Taip	Taip	Taip	Taip
Polyhedra DBMS	Ne	Ne	Taip	Taip	Taip	Taip
PostgreSQL	Taip	Taip	Taip	Taip	Taip	Taip
Pyrrho DBMS	Taip	Taip	Taip	Taip	Taip	Taip
ScimoreDB	Ne	Ne	Ne	Ne	Taip	Taip
SQL Anywhere	Taip	Taip	Taip	Taip	Taip	Taip
SQLite	Ne	Ne	Taip	Ne	Ne	Taip
Teradata	Ne	Taip	Taip	Taip	Taip	Taip
Valentina	Ne	Taip	Taip	Taip	Taip	Ne

Priedas Nr.9 – Duomenų bazių padalinimų galimybės

RDBMS	Range	Hash	Range + Hash	List	Shadow	Native replication API
4th Dimension	Ne	Ne	Ne	Ne	Ne	Ne
ADABAS	Ne	Ne	Ne	Ne	Ne	Ne
Adaptive Server Enterprise	Taip	Taip	Ne	Taip	Ne	Ne
Apache Derby	Ne	Ne	Ne	Ne	Ne	Ne
IBM DB2	Taip	Taip	Taip	Taip	Ne	Ne
Firebird	Ne	Ne	Ne	Ne	Taip	Ne
HSQLDB	Ne	Ne	Ne	Ne	Ne	Ne
Informix	Taip	Taip	Taip	Taip	Ne	Ne
Ingres	Taip	Taip	Taip	Taip	Ne	Ne
InterBase	Ne	Ne	Ne	Ne	Taip	Taip
MaxDB	Ne	Ne	Ne	Ne	Ne	Ne
Microsoft Access	Ne	Ne	Ne	Ne	Ne	Ne
Microsoft Visual Foxpro	Ne	Ne	Ne	Ne	Ne	Ne

Microsoft SQL Server	Taip	Ne	Ne	Ne	Ne	Ne
MonetDB	Taip	Taip	Taip	Ne	Ne	Ne
MySQL	Ne	Ne	Ne	Ne	Ne	Ne
Oracle	Taip	Taip	Taip	Taip	Ne	Ne
Oracle Rdb	Taip	Taip	Ne	Ne	Ne	Ne
OpenLink Virtuoso	Taip	Ne	Ne	Ne	Ne	Ne
Polyhedra DBMS	Ne	Ne	Ne	Ne	Ne	Ne
PostgreSQL	Taip	Taip	Taip	Taip	Ne	Ne
Pyrrho DBMS	Ne	Ne	Ne	Ne	Ne	Ne
ScimoreDB	Ne	Taip	Ne	Ne	Ne	Taip
SQL Anywhere	Ne	Ne	Ne	Ne	Ne	Ne
SQLite	Ne	Ne	Ne	Ne	Ne	Ne
Teradata	Taip	Taip	Taip	Taip	Ne	Ne
Valentina	Ne	Ne	Ne	Ne	Ne	Ne

Priedas Nr.10 – Duomenų bazių limitai

RDBMS	Max DB size	Max table size	Max row size	Max columns per row	Max Blob, Clob size	Max char size	Max number size
Advantage Database Server	Neribota	16 EB	65530 B	65135 / (10 + vidutinis stulpelio pavadinimo ilgis)	4 GB		64 b
Datawasp	Neribota	2 GB	32678 B	256	2 GB	1024 B	64 b
DB2	512 TB	512 TB	32677 B	1012	2 GB	32 KB	64 b
Firebird	Neribota	~32 TB	65536 B	Priklauso nuo naudojamų duomenų tipų	2 GB	32767 B	64 b
Ingres	Neribota	Neribota	256 KB	1024	2 GB	32000 B	64 b
Microsoft Access	2 GB	2 GB	16 MB	255	1 GB	255 B	32 b
Microsoft Visual Foxpro	2 GB	2 GB	16 MB	255	2 GB	16 MB	32 b
Microsoft SQL Server	524258 TB	524258 TB	8060 B	1024	2 GB	8000 B	64 b
MySql 5	Neribota	2 GB	64 KB	3398	4 GB	64 KB	64 b
Oracle	Neribota	4 GB * bloko dydis	Neribota	1000	4 GB	4000 B	126 b
Polyhedra DBMS	Priklauso nuo RAM	2^{32} eilučių	Neribota	65536	4 GB	4 GB	32 b
PostgreSQL	Neribota	32 TB	1.6 TB	250-1600 priklausomai nuo duomenų tipų	1 GB	1 GB	Neribota
ScimoreDB	Neribota	16 EB	8050 B	255	16 TB	8000 B	64 b

Teradata	Neribota	Neribot a	64 KB	2048	2 GB	10	64 b