KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

VERSLO INFORMATIKOS KATEDRA

Andrius Laukaitis

XML dokumentų palaikymas laisvo kodo duomenų bazių valdymo sistemose

Magistro baigiamasis darbas

Darbo vadovas
doc. dr. D. Makackas

Kaunas
2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

VERSLO INFORMATIKOS KATEDRA

XML dokumentų palaikymas laisvo kodo duomenų bazių valdymo sistemose

Magistro baigiamasis darbas

Vadovas
doc. dr. D. Makackas
2009-01-

Recenzentas
E. Toldinas
2009-01-

Atliko
IFM-3/4 gr. studentas
Andrius Laukaitis
2009-01-12

Kaunas
2009

Turinis

Paveikslėlių sąrašas

Lentelių sąrašas

1. Įvadas

Projektuojant programinius komponentus, kurie sąveikauja su duomenų bazių valdymo sistemomis (DBVS), yra galimi šių komponentų keli realizavimo būdai. Naudojant ODBC, ADO, ADO.NET, JDBC ir kitas programinės sąsajas su duomenų bazių valdymo sistemomis.

Praktikoje yra naudojamas ir plintantis duomenų gavimas naudojant XML dokumentus su SQL užklausomis. Ši sąveika realizuojama naudojant Internet Information Service su papildomu programiniu komponentu. Tai užtikrina paprasta priėjimą prie duomenų iš kliento žiniatinklio naršyklės be papildomos programinės realizacijos ir programinės įrangos diegimo. Tai užtikrina paprastą žiniatinklio puslapių kūrimą. Kadangi su duomenų bazės valdymo sistema bendraujama XML dokumentais yra nesudėtinga tokias sistemas integruoti į bendro uždavinio sprendimą naudojant servisus grįstą architektūrą (angl. Service Oriented Architecture).

Darbo tikslas išanalizuoti kokias funkcines galimybes turi laisvai platinamos duomenų bazių valdymo sistemos su XML struktūros duomenimis. Be to pasiūlyti sprendimo būdą, vieningai prieigai prie duomenų DBVS naudojant XML dokumente aprašytas SQL užklausas.

Tam tikslui atlikti darbe analizuota, laisvo kodo programinė įranga, kokiomis sąlygomis ji pasižymi ir platinama. Apžvelgtas laisvo kodo programų licencijavimas ir kokiomis savybėmis ši programinė įranga skiriasi nuo laisvai (ne atviro kodo) platinamos programinės įrangos.

Analizuota, mokamos duomenų bazių valdymo sistemos: Microsoft SQL, Oracle, IBM DB2, ar yra galima duomenų gavyba per XML dokumentus. Analizės metu nustatyta, kad laisvai platinamose duomenų bazių valdymo sistemose, tokiose kaip MySQL, PostgreSQL nėra šios galimybės. Po žodžiu „duomenų gavybas per XML dokumentą“ yra suprantama, kad XML dokumente galime aprašyti SQL užklausas operacijoms su DBVS duomenimis.

Išnagrinėjus trijų didžiausių duomenų bazių gamintojų XML dokumentų sąveika, pastebėta, kad tiktai Microsoft SQL yra išleidusi programinę įrangą, kuri leidžia duomenų bazių valdymo sistemoms sąveikauti su XML dokumentais. Todėl buvo pasirinkta Microsoft XML dokumentų sąveikos programinę įrangą – SQLXML, kaip pavyzdys. Atlikta jos analizė ir jos veikimo pagrindu suprojektuota programinę įrangą, kuri leistų XML dokumentams sąveikauti su šiomis duomenų bazių valdymo sistemomis, t.y. Microsoft SQL, Oracle, MySQL. Taip pat vienas iš tikslų buvo realizuoti tokia programinė įranga arba programinės įrangos tarpininką, kad jisai tiktų bet kokiai duomenų bazių valdymo sistemai. Į šį servisą

perdavus XML dokumentą, būtu atlikta jame aprašyta SQL užklausa, t.y. nereikėtu skirtingai aprašinėti XML dokumento skirtingoms duomenų bazių valdymo sistemoms.

Kitas analizės tikslas buvo išanalizuoti, kaip veikia Internet Information Service, ir šiam paslaugai (servisui) parašyti papildymus sąveikai su duomenų bazių valdymo sistemomis. Tam buvo išanalizuota, kokia yra Internet Information Service architektūra, kokias programinės įrangas (CGI) arba dinamiškai užkraunamas bibliotekas (ISAPI dll) palaiko ir kuri iš jų yra efektyvesnė. Buvo pasirengta naudoti ISAPI programinius komponentus. Tam tikslui atlikti reikėjo suprojektuoti ISAPI programinį komponentą, taip kad jis būtų universalus visoms duomenų bazių valdymo sistemoms.

Tyrimo metu patikrinta kaip greitai šios duomenų bazių valdymo sistemos sąveikauja su savo gamintojų sukurtais sujungimo ryšiais. Taip pat atliktas tyrimas greitaveikos su duomenų bazių valdymo sistemomis, kai kreipiamasi per XML dokumentus.

Šios ataskaitos analitinėje dalyje yra išanalizuota ir aprašyta kas yra laisvas kodas, kokios yra laisvo kodo licencijos, kokiomis sąlygomis yra taikomos vienokios ar kitokios licencijos, ko skiriasi atviras kodas nuo išeitinio kodo, kas yra laisvoji programinė įranga. Taip pat šioje dalyje yra išanalizuota į servisą orientuotos architektūros principas, kokiomis savybėmis ši architektūra pasižymi, kokius privalumus suteikia šios architektūros naudotojams. Taip pat pastebėta, kad dauguma gamintojų XML dokumentą naudoja kaip komunikavimo sąsają, todėl šiame skyriuje yra apžvelgta ir išanalizuota kas yra XML dokumentas, kokius privalumus ir trūkumus jis turi, taip pat pradedant kalbėti apie XML dokumentą buvo trumpai apžvelgtos šio standarto sudėtinės dalys, tokios kaip XML dokumento schemos, ir XML dokumento transformacijos. Tačiau kaip šis darbas yra susijęs su duomenų bazių valdymo sistemomis, buvo išanalizuotos populiariausios duomenų bazės valdymo sistemos, palygintos tam tikros jų savybės.

Šios ataskaitos projektinėje dalyje yra pateiktas visos šios sistemos veikimo principas, apibudinta kokios technologijos yra naudojamos, kodėl šios technologijos buvo pasirinktos, pateikti ir išaiškinti šios sistemos veikimo algoritmai. Taip pat šioje dalyje yra išanalizuotas žiniatinklio/internetinio serverio veikimas, kokius privalumus šis serveris turi, kodėl buvo pasirinktas būtent šis žiniatinklio/internetinis serveris.

Paskutinėje šios ataskaitos dalyje yra pateikti testavimo eksperimentai, kurie buvo atlikti testuojant sujungimo sąsajos su duomenų bazės valdymo sistemos greitaveika, vykdant atitinkamas SQL užklausas. Galiausiai buvo atlikti eksperimentai su jau sukurta ir veikiančia sistema, kad palyginti kiek pablogėjo šios sistemos greitaveika. Šioje dalyje yra pateikti grafikai su paaiškinimais, kodėl buvo gauti vienoki, o ne kitokie rezultatai.

2. Analitinė dalys

2.1. Laisvas kodas ir atviras kodas (Tyrimo objekto analizė)

Atviro kodo (laisvo kodo) programinė įranga. Atvirasis kodas reiškia ne tik priėjimą prie išeitinio kodo. Atvirojo kodo programinės įrangos platinimas turi sutapti su žemiau esamomis nuostatomis. **Išeitinis kodas** – bet kokia sakinių seka, užrašyta žmogui suprantama programavimo kalba. Šiuolaikinėse programavimo kalbose programos išeitinis kodas paprastai susideda iš keleto tekstinių failų, tačiau tas pats kodas galėtų būti spausdinamas knygose ar įrašomas į magnetinę juostą (be failinės sistemos).

Atviro kodo programinė įranga. Tai kompiuterinė programa, kuri platinama pagal atvirojo kodo licenciją. Tokia programinė įranga dažniausiai pasižymi šiomis savybėmis:

- nemokama;
- laisvai prieinami išeities kodai;
- galima laisvai platinti ir modifikuoti, nekeičiant licencijos.

Atviro kodo programinės įrangos vartotojams nauda yra kur kas didesnė, nei naudojantis nemokama programine įranga (angl. *Freeware*), kadangi kiekvienas vartotojas gali:

- prisidėti prie tokios programos tobulinimo – gali laisvai pranešti kūrėjams surastas klaidas,
- tas klaidas pats ištaisyti,
- siūlyti savus patobulinimus (pataisas angl. *patch*),
- prisidėti prie tokios programos vertimo į kitą kalbą (lokalizacija),
- padėti rašyti dokumentaciją, programos aprašymus, vartotojo instrukcijas ir pan.

Laisvoji programinė įranga (laisvos programos, angl. *Free Software* bet ne Freeware) – tai programinė įranga, kurios naudotojai turi keturias laisvės rūšis:

- laisvė paleisti programą bet kuriam tikslui
- laisvė analizuoti kaip veikia programa ir pritaikyti ją savo reikmėms (būtina sąlyga – atviras kodas)
- laisvė platinti programos kopijas (kad galėtume padėti draugui ar kaimynui)
- laisvė tobulinti programą ir viešai platinti nuosavus pataisymus, kad kiekvienas galėtų jais pasinaudoti (būtina sąlyga – atviras kodas).

Laisvas programas dažnai vadina atviro kodo programomis, iš dalies tai tiesa, bet egzistuoja tam tikri skirtumai. Atviro kodo programos akcentuoja priėjimą prie programos išeities tekstų (programos kodo) bei pabrėžia kūrimo modelio atvirumą. Laisvos programos akcentuoja programų laisvę. Visos laisvos programos yra atviro kodo programos, bet ne visos

atviro kodo programos yra laisvos. FSF (angl. Free Software Foundation) kritikuoja atvirojo kodo judėjimą be kita ko ir už tai, kad atvirojo kodo judėjimo tikslai yra vien techniniai ir nesiremia jokiais socialiniais ar etiniais kriterijais.

Laisvas platinimas. Licencija nedraudžia kam nors parduoti ar perduoti programinės įrangos kaip programinės įrangos distribucijos dalies, kurią sudaro programos iš kelių skirtingų šaltinių. Licencija nereikalauja honoraro ar kokio kito mokesčio už tokį pardavimą, tačiau nebūtinai privalo šitai drausti. Mokami automatiniai GPL ir LGPL licencijas turinčių programų atnaujinimai – labiausiai paplitusi komercinė veikla. Apie šias licencijas aptarsime vėliau.

Išeitinis kodas. Programa privalo turėti išeitinį kodą bei turi leisti platinimą išeitiniu kodu, o taip pat sukompiliuota forma. Jei kažkuri dalis neplatinama su išeitiniu kodu, turi būti aiškiai apibrėžtos galimybės įsigyti atvirąjį kodą apmokant ne daugiau nei motyvuotas kopijavimo išlaidas, arba labiau priimtinai – atsisiunčiant internetu nemokamai. Išeitinis kodas turi būti pateiktas forma, kuria bet koks programuotojas galėtų jį modifikuoti. Tyčinis kodo modifikavimas, kuris suklaidintų programuotoją yra draudžiamas. Tarpinės formos, tokios kaip prie-procesoriaus ar transliatoriaus rezultatai yra neleidžiamos.

Išvestiniai darbai. Licencija turi leisti atlikti modifikacijas bei kurti išvestinius darbus, o taip pat turi leisti tuos darbus platinti su tokia pat licencija kaip ir išeitinis kodas.

Autoriaus išeinamojo kodo neliečiamumas. Licencija gali uždrausti platinti modifikuotą išeinamąjį kodą tiktai tuo atveju, jei ji leidžia platinti originalų kodą kartu su pataisymų failais (angl. patches), kurie leistų modifikuoti galutinį produktą kompiliacijos metu. Licencija privalo aiškiai leisti iš modifikuoto išeinamojo kodo sukompiliuotų programų platinimą. Licencija gali reikalauti, kad modifikuoti produktai turėtų kitokį pavadinimą ar versijos numerį, nei autoriaus originalus produktas. GPL, LGPL ir daugelis kitų atviro kodo licencijų modifikuoto išeinamojo kodo platinti nedraudžia, tačiau pakeitimai turi būti dokumentuoti.

Jokių apribojimų panaudojimo sritims. Licencija neturi uždrausti programos naudojimo tam tikroje specifinėje srityje. Pavyzdžiui, licencija negali uždrausti naudoti programos komerciniams tikslams, arba genetinių tyrimų atlikimui.

Licencija neturi riboti kitos programinės įrangos. Licencija neturi riboti programinės įrangos platinimo kartus su licencijuojamomis programomis. Pavyzdžiui, licencija neturi

reikalauti, kad visa programinė įranga platinama toje pačioje laikmenoje privalėtų būti Atvirojo Kodo (angl. Open Source).

2.2. Licencijų tipai

GPL arba GNU Bendra Viešoji Licencija (angl. *GNU General Public License* arba GPL) yra laisvosios programinės įrangos licencija, pradžioje sukurta GNU projektui, šiuo metu tai viena populiariausių atvirojo kodo licencijų.

Pagrindiniai licencijos reikalavimai yra šie:

- Kiekvienas gali platinti originalius išeities tekstus (išeities kodus). Galima imti pinigus už fizinį kopijos perdavimą bei garantinį aptarnavimą.
- Kiekvienas gali taisyti pagal savo poreikius išeities tekstus ir lygiai taip pat platinti savo pakeistą versiją (nurodant, jog ji buvo pakeista).
- Kiekvienas gali platinti programą ir sukompiliuota forma (tai paruošta naudojimui forma), įsipareigodamas paprašius pateikti išeities kodus.
- Draudžiama platinti kitaip, nei numatyta licencijoje.
- Programos modifikavimo metu licencija negali būti pakeista.
- Licencija nesuteikia jokios garantijos – visą riziką dėl programos naudojimo prisiima kiekvienas naudotojas asmeniškai.

Naujojoje GPL3 versijoje šią licenciją turinčios programos autorius taip pat turi teisę (bet ne pareigą) reikalauti:

1. Išsaugoti programoje esančius jos autorystę ar kopijavimo teises nusakančius įrašus.
2. Pašalinti iš platinamos programos jos pradinės autorystės įrašus ir laikytis šalies įstatymais numatytų joje naudojamo prekinio ženklo taisyklių. Šios pataisos prašo stambūs laisvosios programinės įrangos gamintojai, siekiantys jog puikią kokybę garantuojančiu jų prekiniu ženklu pažymėtas laisvąsias programas būtų galima įsigyti tik tiesiogiai iš jų pačių.
3. Reikalauti išsaugoti programoje esančias jos išeities kodo gavimo galimybes.
4. Jei vartotojas savo naudai kelia programinės įrangos patento ieškinį prieš autoriaus parašytoje programoje (ar jos vėlesnėse kitų keistose versijose) naudojamą algoritmą, šios programos autorius turi galimybę atimti tokiam vartotojui (ir glaudžiai su juo susijusiems kitiems vartotojams) teisę naudotis tąja programa.

GNU Laisvoji Bendroji Viešoji Licencija (GNU Laisvoji GPL) (angl. *GNU Lesser General Public License* arba LGPL) yra laisvosios programinės įrangos licencija, taikoma programinėms bibliotekoms. Kadangi bet kuri biblioteka laikoma ją naudojančios programos

dalimi, GPL licenciją turinčios bibliotekos negali būti panaudotos uždaro kodo komerciniuose projektuose. Norint tai leisti, tuo pačiu vis dar išsaugant programuotojų laisves, buvo parengta LGPL, kuri tapo tarpine jungtimi tarp GPL licencijos ir daugiau leidžiančių licencijų (tokių, kaip MIT licencija ir BSD licencija). LGPL licenciją turinčią biblioteką galima naudoti uždaro kodo projektuose tol, kol šie projektai naudoja nepakeistą bibliotekos versiją. LGPL licencijoje nurodyta, jog bet kuris asmuo bet kada savo kopijos licenciją gali pakeisti į GPL.

MIT licencija yra nemokamos programinės įrangos licenciją, kuri yra kilusi iš Masačiusetso technologijos institutas (MIT), ir kurią naudoja MIT X konsorciumas. Tai yra atlaidi, leidžianti (angl. permissive) licenciją, o tai reiškia, kad ši licencija leidžia pakartotinai naudoti patentuota programinę įrangą, su sąlyga, kad licencija yra platinamas kartu su šia programine įranga, bei ši licencija yra suderinta su GPL licencija, o tai reiškia, kad GPL licencijos leidimų derinys ir perskirstymas su programinę įrangą, kuri naudoja MIT licencija.

BSD licencija – viena iš licencijų skirtų atviro kodo programinei įrangai. Pradžioje ji buvo naudojama Berklio Programinės įrangos Unix tipo operacinės sistemos distribucijoje. Iš to ir kilo licencijos pavadinimas. Pirmoji versija buvo pakoreguota ir taip atsirado šiuo metu plačiai naudojama BSD licencijos versija.

Kiekvienas gali modifikuoti ir vėliau platinti už mokestį tiek programinį, tiek ir dvejetainį kodą laikantis šių sąlygų:

- Platinant programinį kodą turi likti nurodytos autorinės teisės, šios sąlygos bei garantijų atsisakymas.
- Platinant dvejetaini programos kodą dokumentacijoje ar kitoje pridedamoje medžiagoje turi būti nurodytos autorinės teisės, šios sąlygos bei garantijų atsisakymas.
- Universiteto pavadinimas bei jo darbuotojų vardai negali būti naudojami produkto palaikymui ir platinimui be raštiško sutikimo.

NetBSD – BSD UNIX operacinė sistema, pasižyminti itin geru perkeliamumu tarp įvairių aparatinių platformų. Tai pirma iš nemokamai platintų atviro kodo operacinė sistema, formaliai išleista kaip NetBSD 0.8 versija 1993 metų gegužę.

FreeBSD yra laisva daugia platforme BSD Unix šeimos operacinė sistema, sukurta 4.4BSD-Lite operacinės sistemos pagrindu. Ji palaiko daugumą procesorių, tarp jų – x86, amd64, Alpha/AXP, IA-64, PC-98 ir UltraSPARC. Naudojant FreeBSD operacinę sistemą galima realizuoti įvairius tinklo, saugumo ir suderinamumo sprendimus, kurie gali būti

sunkiai įgyvendinami kitų, laisvų, ar komercinių operacinių sistemų pagalba. FreeBSD dažniausiai naudojama interneto arba intraneto serverio darbui valdyti. Ji geba pateikti patikimas tinklo paslaugas esant didelei apkrovai, o efektyvus atminties valdymas ir daugia programinis apdorojimas leidžia sparčiai vykdyti daug lygiagrečių procesų. FreeBSD dera su pigia kompiuterine įranga ir tokiu būdu sudaro ekonomišką alternatyvą komercinėms UNIX operacinėms sistemoms. FreeBSD turi platų programinės įrangos pasirinkimą, jos įdiegimo procedūra yra palyginti lengviausia tarp kitų BSD šeimos operacinių sistemų.

OpenBSD yra daugia platformė Unix šeimos operacinė sistema platinama laisvąja BSD licencija. Dedamos visos pastangos užtikrinti jos pernešamumą, atitikimą standartams, korektiškumą, pro-aktyvų saugumą ir integruotą kriptografiją. OpenBSD palaiko daugelio operacinių sistemų programų binarinę emuliaciją [1].

Dažniausiai naudojamų licencijų statistiką, pateiksime žemiau lentele [2].

1 lentelė. Naudojamų licencijų statistika

Licencijos pavadinimas	Procentinis panaudojimas
GNU (GPL) Licencija	60%
GNU (LGPL) Licencija	7%
BSD Licencijos	6%
MIT Licencija	2%
GNU 3 Versija Licencija	2%
Freeware Licencija	2.5%

2.3. Į paslaugas grįsta architektūra (angl. SOA Service Oriented Architecture)

Į paslaugas orientuota architektūra (toliau – *SOA)* – tai naujas programinės įrangos architektūros modelis, kuris gali padėti įmonei lengviau prisitaikyti prie nuolat kintančių verslo sąlygų.

Kas yra paslauga? Paslauga – tai standartizuota pasikartojanti verslo užduotis, pvz., jei patikrinti, ar prekė yra sandėlyje, arba tos prekės pardavimo procedūra. Kiekviena didesnė įmonė turi daugybę tokių sukurtų paslaugų, ir visos jos turi „bendradarbiauti“ tarpusavyje. Įmonei augant, kuriamos vis naujos paslaugos, jos integruojamos į vieną visumą su jau egzistuojančiomis paslaugomis, taip sudarydamos bendrą sudėtingą statinį tinklą. Kiekvieną kartą, atsiradus naujai paslaugai ar jai pasikeitus, reikia vėl iš naujo derinti sąsajas.

Programinės įrangos architektūra – tai reikalavimų visuma, kuria vadovaujantis kuriama programinė įranga. Kuo SOA skiriasi nuo įprastinės architektūros? SOA leidžia padaryti pagrindinį įmonės verslą nepriklausomą nuo jos informacinių technologijų. SOA pagrindas - standartizuotas centrinis modulis (angl. Enterprise Service Bus), per kurį jungiami visi kiti

moduliai. Šis modulis užtikrina, kad visos paslaugos naudotų vienodus standartus, jungia ir transformuoja skirtingus standartus ir sumažina jungčių skaičių. Visos paslaugos yra derinamos tik prie centrinio modulio, todėl jas lengva pakeisti kitomis. Tarkime, jūsų sandėlininkystės programa paseno, tačiau kiti versle naudojami programiniai moduliai jums vis dar tinka. Jeigu norite jį pakeisti nesinaudodami SOA, jums gali tekti redaguoti visas programas, kurios yra susijusios su sandėlininkystės moduliu. O naudojant SOA, jums tereiks pakeisti vienintelį modulį, ir vėl bus galima naudoti visas ankstesnes sąsajas. SOA padaro verslą lankstesnį ir greičiau prisitaikantį prie pokyčių.

Standartizuotas centrinis modulis yra tarpinė programinė įranga, skirta atskirų programų arba sistemų integracijai ir atliekanti duomenų transformavimo ir persiuntimo funkciją. Paprasčiau tariant, tai jungiamoji grandis, sujungianti visas įmonės paslaugas ir leidžianti joms bendradarbiauti.

Paimkime kaip pavyzdį muzikinę namų sistemą. Diegdami ją, turite du pasirinkimus – galite įsigyti paprastą muzikinį grotuvą, kuris bus nebrangus, tačiau jo komponentus keisti ateityje bus labai sudėtinga. Jie sujungti nestandartinėmis jungtimis, tad užsimanius vietoj CD įmontuoti DVD ar MP3 grotuvą taps praktiškai neįmanoma. Kitu atveju, nusipirkę garso stiprintuvą ir atskirai prie jo prijungę norimus komponentus (garsiakalbius, CD grotuvą ir t.t.), juos lengvai pakeisite kada panorėję; jie visi bus moduliniai, t.y. jungiami prie centrinio modulio (šiuo atveju – garso stiprintuvo), o visų jų jungtys bus standartinės. Tokia sistema sukurta taip, kad ją būtų galima pakeisti paprastai ir greitai. Tokia pati sistema yra ir SOA.

Minėtas pavyzdys parodo, jog prie garso stiprintuvo jungti ir keisti komponentus lengva, nes jungtys yra standartizuotos. Būtent šią funkciją IT atlieka atviri standartai. Vadovai jau senokai susiduria su situacija, kai sumanius reformuoti įmonės veiklą ar ją išplėsti, tenka nusivilti ir atsisakyti naujovių, nes įmonėje veikianti IT sistema nespėja naujųjų užduočių įvykdyti laiku. Staiga paaiškėja, kad viską keisti yra per brangu ir beveik neįmanoma.

Gali būti, kad programuotojas, kūręs paslaugų sąsajas, jau nebedirba įmonėje, o jums prireikia kažką keisti. Jeigu tos sąsajos buvo sukurtos tik vienam programuotojui suprantama kalba, gali tekti perdaryti didžiąją IT infrastruktūros dalį. Tačiau jei programuotojas naudojosi atvirais standartais, visa jo sukurta infrastruktūra bus lengvai perjungiama ir pakeičiama, kas ją betvarkytų. Būtent todėl, kad pagrindinis SOA tikslas yra sukurti greitai adaptuojamą ir modifikuojamą infrastruktūrą, šios architektūros diegimui būtini atviri standartai.

Skirtingai nuo įprastos architektūros, kai turėtumėte tarpusavyje susieti visas paslaugas, SOA leidžia naujai sukurtą paslaugą tiesiog prijungti prie centrinio modulio.

Savaime suprantama, kad tai daug paprasčiau, nei kurti naujas sąsajas, ir kuo daugiau modulių turi įmonė, tuo daugiau naudos gaunate iš SOA. Pakeisti modulį yra visai paprasta, nereikia perkurti visų sąsajų - tiesiog išimate vieną modulį ir įdedate kitą. Naudodami SOA jūs įgysite pranašumą prieš konkurentus, nes naujas paslaugas galėsite įdiegti greičiau, o tai leis jūsų įmonei efektyviau prisitaikyti prie kintančios rinkos ir geriau išnaudoti atsiradusias galimybes [3].

Kaip pereiti prie SOA? Yra rekomenduojama naudotis dviem pagrindiniais principais:

Pirmas principas – perėjimas turi būti vykdomas palaipsniui. Jei pasirinkote SOA diegimą, tai nereiškia, kad reikia keisti visą turimą IT infrastruktūrą ar veiklos procesus. Tiesiog visus ateinančius projektus kurkite taip, kad jie atitiktų SOA principus, ir pereikite prie SOA žingsnis po žingsnio.

Antras principas – rekomenduoja pradėti nuo vieno arba kelių išeities taškų:

- **Žmonės:** leiskite žmonėms tiesiogiai ir efektyviai dirbti su turimomis programomis ir išnaudoti turimą informaciją, kad jie galėtų laiku priimti teisingus sprendimus.
- **Procesai:** pradėkite procesų valdymą. Pirmas žingsnis gali būti jau esančių ir veikiančių procesų dokumentavimas ir formalizavimas. Tai padės susidaryti tikresnį vaizdą apie vykstančius procesus ir palengvins jų atnaujinimą bei valdymą ateityje.
- **Informacija:** sutvarkykite ir organizuokite informaciją taip, kad ji būtų patikima, standartizuota ir lengvai prieinama, ir kad ją būtų galima laksčiai panaudoti skirtingose situacijose.
- **Turimų resursų panaudojimas:** stenkitės išnaudoti turimus unikalius resursus, likviduokite pasikartojančias funkcijas. Ši sąlyga galioja ir pagrindinei įmonės veiklai, ir IT infrastruktūrai – jei naudojama programa tenkina veiklos poreikius, ją tereikia integruoti į naują arba plečiamą sistemą.
- **Integracija:** integruokite veikiančius procesus ir veiksmus, kad jie būtų tarpusavyje susiję ir užtikrintų sklandų veiklos tęstinumą atsiradus sutrikimams arba pasikeitimams. Integruokite IT programas ir sistemas lanksčiais standartiniais ryšiais (kaip muzikini centrą). [4]

2.4. XML dokumentas, XML standartas

Norint suprasti, kas yra XML, naudinga suprasti, kas yra duomenų žymėjimas. Žmonės per amžius kūrė dokumentus ir tiek pat laiko juos žymėjo. Pavyzdžiui, mokyklose mokytojai visą laiką žymi studentų darbus. Jie nurodo studentams perkelti pastraipas, aiškiau išreikšti sakinį, ištaisyti rašybos klaidas ir t. t. Dokumentų žymėjimas yra dokumento informacijos

struktūros, prasmės ir išvaizdos apibrėžimas. Jei esate naudoję programos „Word“ keitimų sekimo priemonę, naudojote kompiuterizuotą žymėjimo formą.

Kompiuterijoje „žymėjimas“ tapo „žyme“ (angl. tag). Žymėjimas yra kodų, vadinamų žymėmis (arba kartais ženklais), naudojimo procesas norint apibrėžti bet kokių duomenų struktūrą, išvaizdą ir (XML atveju) prasmę.

HTML ir XML dokumentuose yra duomenų, aplink kuriuos yra žymės, bet tai yra vienintelis dviejų kalbų panašumas. HTML žymės apibrėžia jūsų duomenų išvaizdą ir naudojimą – čia yra antraštės, čia prasideda pastraipa ir t. t. XML žymės apibrėžia jūsų duomenų struktūrą ir prasmę – kas yra duomenys, t.y. meta duomenimis aprašomi duomenys.

Kai aprašote savo duomenų struktūrą ir prasmę, tuos duomenis galima pakartotinai naudoti įvairiais būdais. Pavyzdžiui, jei turite pardavimo duomenų bloką, o kiekvienas bloko elementas aiškiai apibrėžtas, į pardavimo ataskaitą galite įkelti tik tuos elementus, kurių jums reikia, o kitus elementus įkelti į apskaitos duomenų bazę. Kitaip tariant, galite naudoti vieną sistemą savo duomenims generuoti ir žymėti XML žymėmis, tada apdoroti tuos duomenis bet kokioje kitoje sistemoje, neatsižvelgiant į aparatūros platformą ar operacinę sistemą. Šis mobilumas yra pagrindinė priežastis, kodėl XML tapo viena iš populiariausių duomenų mainų technologijų. Taip pat XML dokumentai pasižymi tuo, kad jie nepriklauso nuo kompiuterio platformos ir kad dažnai naudojami, kaip sąsaja tarp programų ar procesų. [5]

Tęsdami atsiminkite šiuos faktus:

- Vietoj XML negalima naudoti HTML. Tačiau XML duomenis galima įkelti į HTML žymes ir rodyti tinklalapyje.
- HTML apribotas iki iš anksto apibrėžto žymių rinkinio, kurį bendrai naudoja visi vartotojai.
- XML leidžia kurti bet kokią žymę, kurios reikia norint apibrėžti jūsų duomenis ir jų struktūrą. Pavyzdžiui, jūs norite saugoti ir bendrai naudoti informaciją apie naminius gyvūnus. Galite kurti tokį XML kodą:

```
<?xml version="1.0"?>
<KATĖ>
  <VARDAS>Murkė</VARDAS>
  <VEISLĖ>Siamo</VEISLĖ>
  <AMŽIUS>6</AMŽIUS>
  <STERILIZUOTA>taip</STERILIZUOTA>
  <NUKIRPTI_NAGAI>ne</NUKIRPTI_NAGAI>
  <LICENCIJA>Izz138bod</LICENCIJA>
  <ŠEIMININKAS>Vardenis Pavardenis</ŠEIMININKAS>
</KATĖ>
```

Matote, kad naudojant XML žymes, galima tiksliai žinoti, į kokius duomenis žiūrima. Pavyzdžiui, žinote, kad šie duomenys yra apie katę, ir galite lengvai rasti katės vardą, amžių ir

kt. Dėl gebėjimo kurti žymes, kurios apibrėžia beveik bet kokio tipo duomenų struktūrą, XML yra „išplėstinis“.

Nesupainiokite šiame kodų pavyzdyje esančių žymių su HTML failo žymėmis. Pavyzdžiui, jei įklijuosite šią XML struktūrą į HTML failą ir peržiūrėsite failą savo naršyklėje, rezultatas atrodys maždaug taip:

```
Murkė Siamo 6 taip ne Izz138bod Vardenis Pavardenis
```

Naršyklė nepaiso jūsų XML žymių ir rodo tik duomenis. Norint matyti gražiai pateiktus duomenys reikia pasirašyti XSLT transformacijas.

XML yra nepriklausomas nuo platformos, tai reiškia, kad bet kuri programa, skirta naudoti XML duomenis, gali skaityti ir apdoroti jūsų XML duomenis, neatsižvelgiant į aparatūrą ar operacinę sistemą. XML yra labai mobili, todėl ji tapo viena iš populiariausių duomenų mainų tarp duomenų bazių ir vartotojų darbalaukių technologijų. [6]

Schemos. Neleiskite terminui „Schema“ jūsų įbauginti. Schema yra tiesiog XML failas, kuriame yra taisyklės, kas gali ir ko negali būti XML duomenų faile. Schemų failai paprastai naudoja .xsd failo vardo plėtinį, o XML duomenų failas naudoja .xml plėtinį.

Schemos leidžia programoms tikrinti duomenis. Jos pateikia duomenų struktūravimo pagrindą ir padeda užtikrinti, kad duomenys bus prasmingi tiek kūrėjui, tiek kitiems vartotojams. Pavyzdžiui, jei vartotojas įveda neleistinus duomenis, pavyzdžiui, tekstą į datos lauką, programa gali raginti vartotoją įvesti teisingus duomenis. Kol XML faile esantys duomenys atitinka pateiktos schemos taisykles, bet kuri XML palaikanti programa gali naudoti tą schemą duomenims skaityti, aiškinti ir apdoroti. [7]

Schemos gali tapti sudėtingos, o mokyti jus kurti schemas nėra šio magistrinio darbo paskirtis. Tačiau naudinga žinoti, kaip schemos atrodo. Ši schema apibrėžia žymių rinkinio <KATĖ> ... </KATĖ> taisykles.

```
<xsd:element name="KATĖ">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="VARDAS" type="xsd:string"/>
      <xsd:element name="VEISLĖ" type="xsd:string"/>
      <xsd:element name="AMŽIUS" type="xsd:positiveInteger"/>
      <xsd:element name="STERILIZUOTA" type="xsd:boolean"/>
      <xsd:element name="NUKIRPTAIS_NAGAIS" type="xsd:boolean"/>
      <xsd:element name="LICENCIJA" type="xsd:string"/>
      <xsd:element name="ŠEIMININKAS" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Nesirūpinkite, jei supratote ne visą informaciją, pateiktą pavyzdyje. Tiesiog atsiminkite šiuos faktus:

- Pavyzdžio schemos eilučių elementai vadinami apibrėžimais. Jei jums reikėjo papildomos informacijos apie gyvūną, pvz., jo spalvas arba dėmes, jus galite pridėti šį apibrėžimą prie schemos. Plėsdami savo verslą galite keisti savo XML sistemą.
- Apibrėžimai leidžia labai griežtai valdyti duomenų struktūrą. Pavyzdžiui, apibrėžimas <xsd:sequence> reiškia, kad tokios žymės kaip <NAME> ir <BREED> turi būti rodomos tokia tvarka, kokia jos išvardytos aukščiau. Apibrėžimai taip pat valdo duomenų, kuriuos gali įvesti vartotojas, tipus. Pavyzdžiui, aukščiau esančiai schemai reikia teigiamo skaičiaus, nurodančio katės amžių, ir Bulio (TRUE arba FALSE) reikšmių žymėms ALTERED ir DECLAWED.

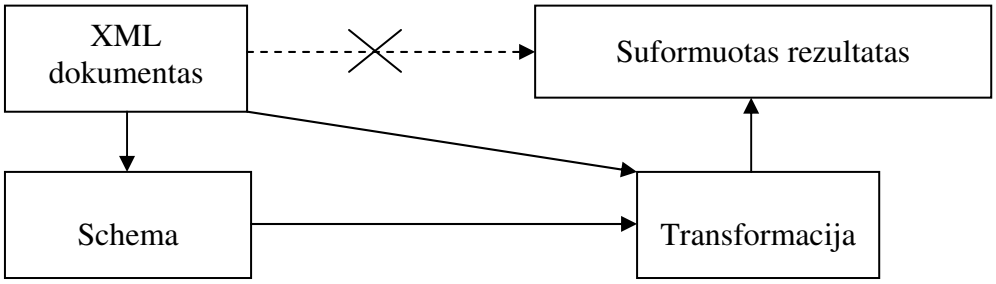
Kai XML faile esantys duomenys atitinka schemos pateiktas taisykles, duomenys laikomi leistinais. XML duomenų failo tikrinimo pagal schemą procesas vadinamas (pakankamai logiškai) tikrinimu. Didelis schemų naudojimo pranašumas yra tas, kad jos gali padėti uždrausti sugadintus duomenis. Taip pat jos leidžia lengvai rasti sugadintus duomenis, nes susidūrusi su problema, XML sustoja. [8]

Transformacijos. Anksčiau užsiminėme, kad XML taip pat pateikia patogių būdų naudoti arba pakartotinai naudoti duomenis. Pakartotinio duomenų naudojimo mechanizmas vadinamas išplėstinės stilių lapų kalbos transformacija (XSLT) arba tiesiog transformacija. Transformacijų aspektu XML tampa tikrai įdomi. Jūs (na gerai, jūsų IT skyrius) galite taikyti transformacijas norėdami mainytis duomenimis tarp galutinių sistemų, pvz., duomenų bazių. Tarkime, kad duomenų bazėje A pardavimo duomenys saugomi kaip lentelė, kuri tinka pardavimo skyriui. Duomenų bazėje B pajamų ir išlaidų duomenys saugomi kaip lentelė, kuri skirta buhalterijai. Duomenų bazė B gali naudoti transformaciją, kad priimtų duomenis iš duomenų bazės A ir rašytų juos į tinkamą lentelę.

Duomenų failo, schemos ir transformacijos derinys sudaro bazinę XML sistemą. Šioje iliustracijoje vaizduojama, kaip tokios sistemos paprastai veikia. Duomenys tikrinami pagal schemą, tada juos visais įmanomais būdais keičia transformacija. Šiuo atveju transformacija išdėsto duomenis tinklalapio lentelėje. Šis pavyzdys neskirtas parodyti, kaip rašyti transformaciją, jis tik vaizduoja vieną iš formų, kurią gali įgauti transformacija. Transformacijos dokumentas atrodo taip: [9]

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0">
<TABLE>
  <TR>
    <TH>Vardas</TH>
    <TH>Veislė</TH>
    <TH>Amžius</TH>
    <TH>Sterilizuota</TH>
    <TH>Nukirpti nagai</TH>
    <TH>Licencija</TH>
```

```
    <TH>Savininkas</TH>
  </TR>
  <xsl:for-each select="KATĖ">
  <TR ALIGN="LEFT" VALIGN="TOP">
    <TD>  <xsl:value-of select="VARDAS"/>   </TD>
    <TD>  <xsl:value-of select="VEISLĖ"/>   </TD>
    <TD>  <xsl:value-of select="AMŽIUS"/>   </TD>
    <TD>  <xsl:value-of select="STERILIZUOTA"/>  </TD>
    <TD>  <xsl:value-of select="NUKIRPTI_NAGAI"/></TD>
    <TD>  <xsl:value-of select="LICENCIJA"/> </TD>
    <TD>  <xsl:value-of select="SAVININKAS"/></TD>
  </TR>
</xsl:for-each>
</TABLE>
```



1 pav. XML dokumento patikrinimas, transformacija ir rezultato grąžinimas

XML dokumentų naudojimas Microsoft produktuose. Kaip jau buvo pastebėta, vis daugiau gamintojų ir programuotojų naudoja XML formatą, net gi naujas Microsoft Office 2007 pasižymi, to kad jo išsaugoti failai yra XML formate, kurio pranašumai yra tokie:

- Mažesni failai. Naujasis formatas naudoja ZIP ir kitas glaudinimo technologijas, kad sumažintų failo dydį iki 75 procentų palyginus su dvinariais formatais, kurie naudojami ankstesnėse Office versijose.
- Paprastesnis informacijos atkūrimas ir geresnė sauga. XML gali skaityti žmogus, todėl jei failas sugadinamas, galite atidaryti failą Microsoft užrašinėje ar kitoje teksto skaitymo programoje ir atkurti bent dalį savo informacijos. Taip pat naujieji failai yra saugesni, nes juose negali būti Visual Basic for Applications (VBA) kodo. Jei kurdami šablonus naudojate naująjį formatą, visi ActiveX valdikliai ir VBA makrokomandos laikomos atskiroje saugesnėje failo dalyje. Be to, galite naudoti tokius įrankius, kaip dokumentų inspektorius, kad pašalintumėte visus asmeninius duomenis.
- Didesnis mobilumas ir lankstumas. XML saugo duomenis teksto formatu, o ne savininko dvinariu formatu, todėl jūsų klientai gali apibrėžti savo schemas ir naudoti jūsų duomenis įvairiais būdais už tai nieko nemokėdami. [10]

2.5.Duomenų bazės samprata

Duomenų bazė (DB) yra dar vienas informacijos mokslo tyrimų ir tobulinimo objektų. DB yra logiškai susijusių duomenų visuma, kuri saugoma ir naudojama apibrėžtoje DS. Duomenų bazėje esanti informacija leidžia patenkinti tam tikrus individo ar organizacijos poreikius.

Duomenų ir jų tarpusavio ryšių aprašymas yra loginis ir fizinis. Fizinis duomenų aprašymas- tai fiziniai duomenų saugojimo būdai išorinėje atmintyje, o loginis duomenų bazės aprašymas yra skirtas jos vartotojams. Duomenų bazė yra organizuoti surinkti duomenys. Terminas “duomenų bazė” atsirado kartu su kompiuterine pramone, bet jo prasmė buvo praplėsta ilgainiui naudojant duomenų bazes. Tačiau iki šiol galima sutikti net keletą duomenų bazės apibūdinimų. Vienas iš galimų variantų, jog duomenų bazė- tai kolekcija sistemiškai įvestų įrašų, saugomų kompiuteryje taip, kad kompiuterinė programa gali naudoti duomenų bazės informaciją norėdama atsakyti į pateiktą klausimą. Efektyvesniam darbui užtikrinti kiekvienas įrašas duomenų bazėje yra pateikiamas kaip informacijos elementų (faktų) rinkinys. Tie faktai, kurie pateikiami sistemos, po to kai buvo suskurta užklausa gali būti naudojama kaip informacija sprendimui priimti. Kompiuterinė programa naudojama valdyti ir pateikti užklausas duomenų bazei vadinama duomenų bazės valdymo sistema.

Paprastai kiekviena duomenų bazė turi savo struktūra, dar vadinama schema. Schema parodo objektų, pateikiamų duomenų bazėje, tarpusavio ryšį. Yra daugybė būdų sukurti duomenų bazės schemą. Šios variacijos vadinamos duomenų bazės modeliais. Šiuo metu dažniausiai naudojamas reliacinis modelis, kuris kalbant neprofesionaliais terminais, visą informaciją pateikia daugialypiais ryšiais susijusiomis lentelėmis. Kiekviena iš jų susideda iš eilučių ir stulpelių. Kiti modeliai, tokie kaip hierarchinis ar tinklinis, naudoja detalesnį tarpusavio ryšių pateikimą. Duomenų bazių valdymo sistemos paprastai skirstomos į kategorijas pagal duomenų modelį: reliacinis, tinklinis ir pan. [11]

Duomenų bazių modeliai. Naudojamos pačios įvairiausios technologijos sukurti duomenų struktūrą. Duomenų modelis - tai ne tik informacijos struktūra, bet ir operacijos, kurias leidžia atlikti su turima informacija. Pavyzdžiui, reliacinis modelis apibrėžia tokias operacijas kaip išrinkimas, projektavimas ir susijungimas. [12]

Reliacinis modelis. Edgar Codd visuomenei pateikė Reliacinį duomenų modelį didelės apimties duomenų laikmenoms. Jis pasiūlė naują modelį didelėms duomenų bazėms naudojant lenteles su fiksuotu įrašų skaičiumi. Tai matematinis modelis paremtas predikatine logika ir aibių teorija.

Produktai, kurie yra laikomi reliacinėmis duomenų bazėmis (pavyzdžiui, Ingres, Oracle, DB2 ir SQL Server) realizuoja modelį, kuris tik apytikriai panašus į Kodo apibūdintą matematinį modelį. Šiuose produktuose duomenų struktūrą sudaro labiau lentelės nei ryšiai: pagrindinis skirtumas tas, jog čia lentelės gali turėti dubliuotas eilutes ir eilutės (arba stulpeliai) gali elgtis taip, kaip numatyta. Ta pati kritika tinka ir SQL kalbai, kuri yra pradinė šių produktų sąsaja. Šis klausimas gana ginčytinas net pačiam Kodui, ar teisinga apibūdinti SQL įrankius kaip „reliacinius“: bet faktas yra toks, kad visas pasaulis taip daro, ir sekantis apibūdinimas naudoja šį terminą populiariąja prasme.

Reliacinė duomenų bazė susideda iš keleto lentelių, panašių į lentelę „plokščiame“ duomenų bazės modelyje. Ryšiai tarp lentelių nėra aiškiai apibrėžiami, vietoje to raktai naudojami eilučių iš skirtingų lentelių suderinimui. Raktas – tai vieno ar daugiau stulpelių rinkinys vienoje lentelėje, kurios vertės sutampa su atitinkamais stulpeliais kitose lentelėse. Pavyzdžiui, lentelė *Darbuotojas* gali turėti stulpelį pavadintą *Vietovė*, kurio vertės sutampa su raktu lentelėje *Vietovė*. Bet kuris stulpelis gali būti raktu, arba keletas stulpelių gali būti sugrupuoti į vieną raktą. Nėra būtina iš anksto numatyti visus raktus, stulpelis gali būti panaudotas kaip raktas net ir tada, jei iš pradžių nebuvo tam numatytas.

Raktas, kuris gali būti panaudotas identifikuoti tam tikrą eilutę lentelėje, vadinamas unikaliu raktu. Dažniausiai vienas iš unikalių raktų yra identifikuojamas kaip lentelės pirminis raktas.

Raktas, kuris turi išorinę, realiai egzistuojančią reikšmę (tokią kaip asmens vardas, knygos ISBN numeris, automobilio registracijos numeris), kartais vadinamas natūraliu raktu. Jei neatsiranda tinkamo natūralaus rakto (pavyzdžiui, yra daug žmonių su pavarde Brown), sutartinis raktas gali būti suteikiamas (kaip darbuotojams suteikimas ID numeris). Daugeliu atveju duomenų bazės turi ir dirbtinius, ir natūralius raktus, nes dirbtiniai raktai gali būti naudojami vidinių ryšių tarp eilučių sukūrimui, o natūralūs raktai naudojami, nors ir mažiau patikimi, paieškai ir integracijai su kitomis duomenų bazėmis. (Pavyzdžiui, įrašai dviejose atskirai sukurtose duomenų bazėse gali būti susieti per socialinio draudimo numerį, išskyrus tos atvejus, kai šie numeriai yra neteisingi, dingę arba pasikeitę). [12]

Reliacinis veikimas. Vartotojai (arba programos) reikalauja duomenų iš reliacinės duomenų bazės siųsdami jai užklausą, kuri parašyta specialia kalba, dažniausiai SQL dialektu. Nors SQL buvo numatyta galutiniams vartotojams, dabar SQL užklausos dažniausiai įterptos programinėje įrangoje, kuri supaprastina vartotoja sąsaja.

Atsakydama į užklausą duomenų bazė pateikia rezultatų rinkinį, kuris yra tik eilučių, į kurias įeina atsakymas, sąrašas. Paprasčiausia užklausa pateikia visas lentelės eilutes, bet dažniausiai eilutės yra kokiu nors būdu filtruojamos ir pateikiamas tik pageidaujamas atsakymas.

Dažnai duomenys iš keleto lentelių sujungiamos į vieną. Teoriškai, tai daroma imant visas įmanomas eilučių kombinacijas („kryžminis produktas“), ir tada išfiltruojant viską, išskyrus atsakymą. Praktiškai, reliacinės DBVS perrašo („optimizuoja“) užklausas, kad galėtų jas vykdyti greičiau ir naudoja įvairius būdus.

Reliaciniu duomenų bazių lankstumas leidžia programuotojams pateikti tokias užklausas, kokios nebuvo numatytos duomenų bazės kūrėjų. Todėl reliacinės duomenų bazės gali būti naudojamos keleto programų tokiais būdais, kurių nenumatė jų kūrėjai, tai labai svarbu toms duomenų bazėms, kurios gali būti naudojamos ilgą laiką. Būtent ši savybė išpopuliarino reliacines duomenų bazes verslo pasaulyje. [13]

Duomenų bazes indeksavimas. Visų rūšių duomenų bazės naudojasi indeksavimo privalumu greičio padidinimui. Bendriausia indeksavimo rūšis, tai rūšiuotas turinio sąrašas, iš tam tikro lentelės stulpelio, su nuorodomis į eilutę susieta su reikšme. Indeksai leidžia greitai rasti lentelės eilučių, atitinkančių pateiktus kriterijus, rinkinį. Naudojami įvairūs indeksavimo metodai, B-medžiai, smulkinimas ir susiję registrai (B-trees, hashes, linked lists) – visi yra dažnai naudojamos indeksavimo technikos.

Reliacinės DBVS turi privalumą, jog indeksai gali būti sudaryti ar išardyti nekeičiant egzistuojančios programos, kuri naudos indeksus. Duomenų bazė renkasi tarp daugelio skirtingų strategijų, ir grindžiasi įvertinimu, kuri iš jų veiks greičiausiai.

Reliacinės DBVS naudoja daug skirtingų algoritmų SQL sakinio apskaičiavimui. Šios sistemos paruošia planą, kaip įvykdyti užklausą, kuri sugeneruojama analizuojant įvairių algoritmų trukmes ir pasirenkant greičiausią iš jų. [13]

Perdavimas ir sutapimas. Šalia savo duomenų modelio praktiškiausios duomenų bazės („transakcinės duomenų bazės“) bando vykdyti ir duomenų bazės perdavimo modelį, kuris turi pageidaujamų duomenų integruotumo savybių. Idealiu atveju duomenų bazės programinė įranga turėtų žemiau pateiktas taisykles (ACID rules):

Vientisumas (angl. Atomicity) – arba visos užduotys perdavimo metu turi būti įvykdytos, arba nė viena iš jų. Perdavimas turi būti užbaigtas, arba neįvykdomas (grįžtama į ankstesnį lygį).

Nuoseklumas (angl. Consistency) – kiekvienas perdavimas turi išlaikyti vientisumo nuostatas – deklaruotas vientisumo taisykles. Duomenys negali atsidurti prieštaringose pozicijose.

Izoliuotumas (angl. Isolation) – du vienalaikiai perdavimai negali trukdyti vienas kitam. Vidiniai perdavimo rezultatai nematomi kietiems perdavimams.

Patvarumas (angl. Durability) – atliktas perdavimas vėliau nebegali būti atšauktas, jo rezultatai negali būti atmesti. Jie turi išlikti po DBVS perkrovimo (pavyzdžiui, po „lūžimo“).

Praktikoje, daug DBVS leidžia šiokį tokį laisvumą šių taisyklių atžvilgiu, tam, kad būtų pagerintas veikimas.

Sutapimo kontrolė yra metodas naudojamas užtikrinti, kad perdavimas yra saugiai įvykdytas ir atitinka ACID taisykles. DBVS turi užtikrinti, kad tik serijinis ir grąžintinas rėžimas yra leistas, kad jokie atliktų perdavimų veiksmai nėra prarandami naikinant nutrauktus perdavimus.

Kopijavimas. Duomenų bazių kopijavimas glaudžiai susijęs su perdavimu. Jei duomenų bazė gali įsiminti savo pavienius veiksmus, ji gali realiu laiku kurti duomenų dublikatą. Dublikatas gali padėti pagerinti visos duomenų bazės sistemos veikimą ir prieinamumą.

Į įprastą kopijavimo sąvoką įeina:

Šeimininkas/Vergas (angl. Master/Slave) replikacija. Visos užklausos vykdomos pagrindinėje kopijoje, o paskui perduodamos šalutinei.

Kvorumas (angl. Quorum). Skaitymo ir rašymo užklausos rezultatai gaunami užklausiant daugumą kopijų.

Daugiakopijinis (angl. Multimaster). Dvi ar daugiau kopijų sinchronizuojasi transakcijos metu.

Duomenų bazės valdymo sistema (DBVS) – tai sisteminė programinė įranga, kuria vartotojai gali apibrėžti, kurti ir palaikyti DB, taip pat atlikti į ją kontroliuojamą kreipti. DBVS sąveikauja su taikomąja programine įranga, t. y. taikomosiomis vartotojo programomis ir duomenų baze bei suteikia šias galimybes**:** duomenų aprašymo kalba apibrėžti duomenų bazei. Duomenų aprašymo kalba teikia vartotojui duomenų tipų, jų struktūros ir apribojimų aprašymo priemones; duomenų manipuliavimo kalba skirta įterpti, atnaujinti, šalinti ir gauti informaciją iš duomenų bazės. Yra du duomenų manipuliavimo kalbos tipai - procedūrinės ir neprocedūrinės kalbos, kurios skiriasi viena nuo kitos duomenų gavimo būdais. Procedūrinės kalbos apdoroja informaciją duomenų bazėje nuosekliai, įrašas po įrašo, o neprocedūrinės - operuoja įrašų rinkiniais. Todėl procedūrinėmis kalbomis nurodoma, kaip galima gauti norimą rezultatą, o neprocedūrinėmis - aprašyti tai, ką norima gauti. Labiausiai paplitęs neprocedūrinių kalbų tipas yra struktūruota užklausų kalba (Structured Query Language - SQL), kuri dabar yra apibrėžta specialiu standartu ir yra būtina kalba visoms reliacinėms DBVS; saugumo ir vientisumo palaikymo, lygiagretaus darbo valdymo ir duomenų bazes atkūrimo po aparatinių arba programinių trikdžių sistemomis kontroliuoti kreiptį į duomenų bazę.

Lygiagretaus darbo valdymo sistema kontroliuoja bendrą vartotojų kreipti į duomenų bazę. Ji palaiko duomenų bazės struktūrinių vienetų (rinkinių, įrašų, laukų)

blokavimo ir atnaujinimo operacijas, kontroliuoja kreipties vykdymo laiką ir apdoroja transakcijas (transakcija - tai vartotojo darbo su duomenų baze seka). DBVS suteikia papildomų funkcijų duomenų bazei ir ji tampa lankstesne ir naudingesne duomenų kaupimo ir saugojimo priemone. Tačiau verta paminėti ir DBVS efektyvumą, kaip vien iš jos įvertinimo kriterijų. Efektyvumas gali būti vertinamas pagal užklausų atlikimo laiką, informacijos paieškos bei ataskaitų sudarymo greitį. DBVS darbo efektyvumą lemia duomenų vientisumo palaikymo priemonės ir duomenų bazės projektavimo bei realizacijos kokybė.

Labiausiai paplitusios ir populiariausios DBVS yra kuriamos pagal reliacinį modelį. Jos įgyvendina visas aukščiau išvardintas DBVS galimybes. [13]

2.6. MySQL duomenų bazės valdymo sistema

MySQL – viena iš reliacinių duomenų bazių valdymo sistemų, palaikanti daugelį naudotojų, dirbanti SQL kalbos pagrindu. MySQL yra atviro kodo programinė įranga (GPL ir kitos licencijos), vystoma ir palaikoma švedų kompanijos „MySQL AB“, kurios įkūrėjai – švedai David Axmark, Allan Larsson ir suomis Michael „Monty“ Widenius.

MySQL duomenų bazės valdymo sistema tapo vieną populiariausių pasaulyje atviro kodo duomenų bazės valdymo sistema, kadangi ji atsako šiems reikalavimam:, greito veikimo užtikrinimas, didelio patikimumo ir patogumo naudojant. Ši duomenų bazių valdymo sistema naudojama visuose žemynuose. Šia duomenų bazės valdymo sistemą naudoja individualių Interneto svetainių kūrėjai, o taip pat daugelis iš pasaulyje didžiausių ir sparčiausiai augančių organizacijų, norėdami sutaupyti laiko ir pinigų ir užtikrinti pastovu ir greitą jų didelės apimties tinklapių pasiekiamumą, taip pat užtikrinti verslui kritinių sistemų ir programinės įrangos patikimumą.

MySQL reliacinės duomenų bazės valdymo sistemos veikia daugelyje platformų, ji dažnai pasirenkama programuojant internetines svetaines. Šiame sektoriuje su MySQL bando konkuruoti PostgreSQL.

Pastaruoju metu MySQL vis dažniau pritaikoma labai didelėse informacinėse sistemose. Pavyzdžiui, yra tokie internetiniai puslapiai arba programinės įrangos sistemos, kurios apkrovimas kartais viršija 10 tūkstančių užklausų per sekundę, arba dar yra tokiu sistemų, kuriose yra labai daug duomenų bazės lentelių, pavyzdžiui JAV kabelinės televizijos tinklas „Cox Communications“, kurio duomenų bazėje – daugiau kaip 3600 lentelių. Šiame sektoriuje pagrindinis MySQL konkurentas yra Oracle duomenų bazės valdymo sistema. [14]

Nors prieigai prie MySQL duomenų bazių dažniausiai pasirenkama PHP kalba, ją taip pat galima pasiekti įvairiomis kitomis programinėmis priemonėmis – C, C++, C#, Java, Perl,

Python ir kitomis. Kiekvienai šių kalbų sukurtos specialios bibliotekos (API). Taip pat MySQL duomenų bazėms yra sukurta ODBC sąsaja MyODBC, leidžianti duomenis pasiekti bet kuria kalba, neturinčia specialios bibliotekos, tačiau palaikančia ODBC komunikavimo mechanizmą. Tačiau ne per seniausiai MySQL pristatę naują sąsajos sujungimo biblioteka, kuri yra pagrįsta .NET technologijomis, ir ją pavadino MySQL .NET Connector. Taip pat reikėtu paminėti, kad PHP kalba yra parašytas duomenų bazės valdymo įrankis - phpMyAdmin.

Galime teigti, kad MySQL duomenų bazės valdymo sistema pasižymi ne tik kad ji yra populiari pasaulyje atviro kodo duomenų bazės valdymo sistema, bet ji taip pat tampa duomenų bazė, suteikianti galimybę naudotis naujos kartos programas pastatytas ant LAMP platformos, t.y. Linux, Apache, MySQL, PHP / Perl / Python. MySQL duomenų bazių valdymo sistema veikia daugiau nei ant 20 platformų, įskaitant Linux, Windows OS / X, HP-UX, AIX, Netware, suteikiant platformos pasirinkimo lankstumą, kad mes galėtume šią duomenų valdymo sistemą, naudotis ant skirtingu platformų, neprisirišant prie vienokios ar kitokios. [15]

2.7. Microsoft SQL Server duomenų bazės valdymo sistema

Microsoft SQL Server - tai reliacinė duomenų bazės valdymo sistema, kurios gamintojas yra Microsoft. Microsoft SQL Server 2005 produktų šeimą, kad jie geriau atitiktų kiekvieno kliento poreikius ir išleido keturis leidimus: Express, Workgroup, Standard ir Enterprise versijas (angl. edition). Kaip nebrangi pagrindinė duomenų bazė SQL Server suteiks ypatingos naudos ir funkcijų, palyginti su konkuruojančiais sprendimais. Keturi nauji leidimai siūlo funkcijų asortimentą nuo prieinamumo ir galimybės keisti iki patobulintų verslo tyrimo įrankių, sukurtų, kad vartotojai visoje organizacijoje galėtų saugiau, patikimiau ir produktyviau naudoti duomenų valdymo platformą. Be to, programų prastovos laiko sumažinimas, galimybė keisti, funkcionavimo efektyvumas ir griežta saugos kontrolė stulbinamai pastūmėjo SQL Server į priekį pasaulyje daugiausia reikalaujančių įmonių sistemų palaikymo srityje. Kadangi SQL Server yra Windows serverių sistemos dalis, klientai gaus papildomos naudos dėl to, kad, Windows serverių sistemos produktuose įgyvendinus bendrą projektavimo strategiją pagerėjo valdymas ir integracija, todėl sumažėjo visa savikaina ir pagreitėjo kūrybos tempas. [16]

2.8. Oracle duomenų bazės valdymo sistema

Oracle – viena didžiausių ir dažniausiai naudojamų reliacinių duomenų bazių valdymo sistemų.

Oracle yra labai plačiai naudojama bankinėse, finansinėse ir mokslinėse sistemose duomenims saugoti, apdoroti ir analizuoti. Šia sistema dažniausiai pasirenkama, todėl kad ji dirba su dideliais duomenų kiekiais. **Oracle DBVS** - yra vienas iš daugelio klientas/serveris modelių, efektyviai valdančių savo resursus, informaciją esančią duomenų bazėje ir aptarnaujantis daugybę vartotojų, besikreipiančių į duomenų bazę su užklausomis, arba atnaujinančių informaciją per tinklą. Oracle serveris susideda iš komponentų, skirtų darbui su pagrindinėmis reliacinio modelio sritimis. [17]

2.9. Duomenų bazių specifikacijos

Taip pat reikėtu parodyti tam tikra duomenų bazių valdymo sistemų specifikaciją, kuriose platformose jos veikia, kokias funkcijas palaiko. Tai atsispindi žemiau pateiktoje lentele. [18]

2 lentelė. DBVS palyginimas

Produktas	IBM DB2 Express	Microsoft SQL Server Express	MySQL v.5.0	Oracle 10g XE
Procesoriaus palaikymo skaičius	Maksimaliai 2 procesoriai	Vienas procesorius (kiekvienam vartotojui suteikiama gyja)	Neribojama (kiek paliko OS arba kompiuterinė įranga)	Maksimaliai 4 branduoliai
Maksimalus duomenų bazės dydis	512 GB per lentelės dydį, maksimaliai 32768 lentelių, praktiškai šis apribojimas yra žymiai mažesnis	4 GB	4GB – 64TB per lentele priklausomai nuo saugojimo mechanizmo (angl. storename) arba OS apribojimo	Kiek palaiko kompiuterinė įranga. Teoriškai apribota iki 140TB
Operacinių sistemų palaikymas	Windows, Linux	Windows	Visi Unix, Linux, Windows, NetWare, Mac OS...	Windows, Linux, Solaris, HP OpenVMS, Mac OS, AIX, IBM...
Serverio atsarginių kopijų darymas ir atstatymas	Taip	Taip	Taip	Ne
Saugomos procedūros ir funkcijos	Taip	Taip	Nuo MySQL 5.0 iš dalies palaiko	Taip
Vartotojų sukurtos funkcijos	Taip	Taip	Taip	Nėra duomenų
Slaptažodžių valdymas	Taip	Taip	Taip	Taip
Duomenų kodavimas	Taip	Taip	Taip	Taip
Java	Taip	Taip	Taip	Taip
JDBC	Taip	Taip	Taip	Taip
XML	Taip	Taip	Ne	Taip
ODBC	Taip	Taip	Taip	Taip
.NET	ODBC.NET	Taip	.NET Connector	Taip

Lentelė sudaryta remiantis [5]

3 lentelė. DBVS parametrų palyginimas

	SQL Server 2000	**MySQL v 5.0**	**Oracle**
Atsarginės kopijos	Taip	Taip	Taip
Replikavimas	Momentinis, transakcinis Apjungtas	Vienos krypties	Momentinis, transakcinis Apjungtas
Grupė (Clustering)	Taip	Taip	Taip
Saugos ypatumai	Aukštas lygis	Aukštas lygis	Aukštas lygis
Užsirakinimas ir sutapimų palaikymas	Pilnai automatinis	Eilutės rakinimas (InnoDB)	Pilnai automatinis
Sistemos stabilumas	Aukštas	Aukštas	Aukštas
GUI Administravimo įrankiai	Taip	Taip (reikia papildomai atsisiųsti), pvz. phpMyAdmin	Taip

Funkcijų palaikymas duomenų bazių valdymo sistemose

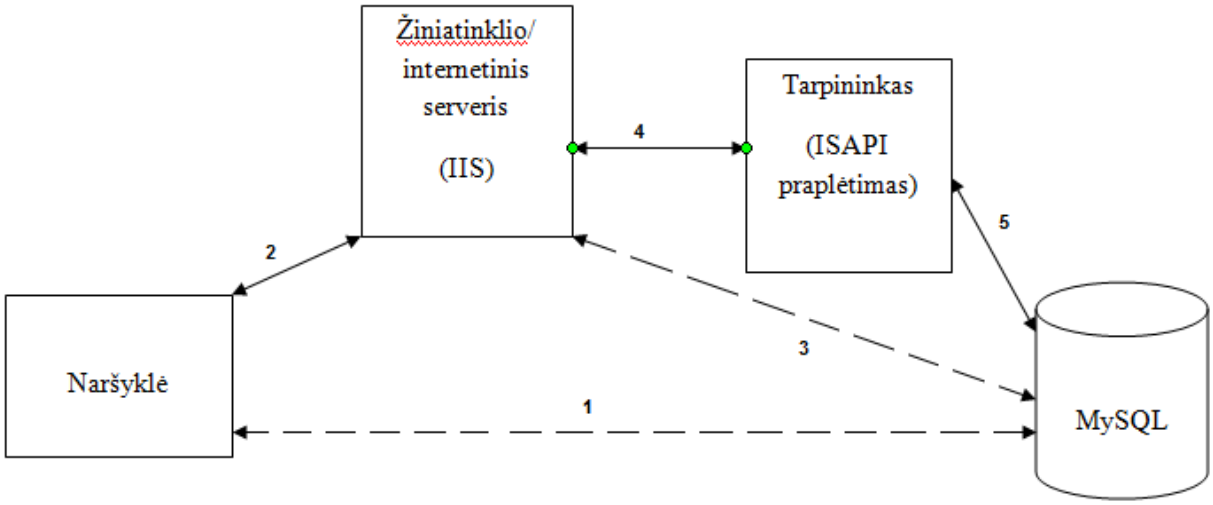
4 lentelė. DBVS funkcijų palyginimas

Funkcija	**Microsoft SQL**	**MySQL**	**Oracle**
Transact-SQL	Palaiko (SELECT FOR XML)	Palaiko	Palaiko
OpenXML	Palaiko (XML dokumento duomenų įvedimas į duomenų bazę)	Skirtingose šaltiniuose rašoma skirtingai (Daugiau šaltinių kad nepalaiko)	Palaiko
FOR XML	Palaiko	Nepalaiko	Palaiko/Nepalaiko
Stored Procedures	Palaiko	Palaiko	Palaiko
ACID	Palaiko	Palaiko	Palaiko
ANSI SQL	Palaiko	Palaiko	Palaiko

Išnagrinėjus šaltinius internete, pastebėta, kad XML dokumentai palaiko sąryši su mokomomis duomenų bazių valdymo sistemomis, tokiomis kaip MS SQL, Oracle, tačiau daug žmonių dirba su laisvai platinamomis duomenų bazių valdymo sistemomis, tokiomis kaip MySQL, Postgre SQL ir kitomis. Nes šios laisvai platinamos sistemos mažai ko nusileidžia mokomom sistemom, aišku neskaitant tai kad mokamos sistemos apdoroja duomenis greičiau, efektyviau, dirba su labai dideliais duomenų kiekiais, turi savyje integruotus įrankius surinkti statistikai ir pan.. Bet išnagrinėjus šias laisvai platinamas duomenų bazių valdymo sistemas, pastebėta, kad šitie produktai nepalaiko XML dokumentų, kitaip tariant XML dokumentai nesąveikauja, su laisvo kodo duomenų bazių valdymo sistemomis, ir šioje vietoje pastebėtas vienas trukumas, kad norint apjungti šias duomenų bazių valdymo sistemas nėra galimybės, nes nėra sąveikos palaikomo ryšio per XML dokumentus.

3. Projektinė dalys

3.1. Sistemos projektavimas



2 pav. Kuriamos sistemos projektavimas

Šio darbo tikslas sukurti sistemą, kurios pagrindinė savybė butu interneto naršyklės pagalba iš duomenų bazes išgauti duomenys XML pavidalu, nepriklausomai nuo to į kokia duomenų bazės valdymo sistemą yra kreipiamasi. Nesvarbu ar vartotojas iš naršyklės kreipiasi į MySQL arba Oracle duomenų bazių valdymo sistemas.

Nagrinėjant internetinės naršyklės galimybės, nebuvo rastos tiesioginės sąsajos tarp internetinės naršyklės ir duomenų bazės valdymo sistemos, t.y. nėra tokios galimybės tiesiogiai be jokiu tarpininkų išgauti duomenys iš duomenų bazės valdymo sistemos XML pavidalu. Kaip pavaizduota 2 paveiksliuke, pirmos sąsajos žingsnis yra negalimas. Tačiau analizės metu pastebėta, kad visos naršyklės užklausos turi kreiptis į žiniatinklio/internetinį serverį, kuris apdoroja užklausą ir jeigu neįvyko klaida gražina rezultatą arba priešingu atveju gražina klaidos pranešimą. Todėl 2 paveiksliuke pavaizduota antroji sąsaja yra galima. Vartotojas gali iš savo naršyklės kreiptis į žiniatinklio/internetinį serverį, atitinkamu adresu.

Tačiau atliekant tolimesni tyrimą, pastebėta, kad žiniatinklio/internetinis serveris neturi tiesioginės sąsajos su duomenų bazės valdymo sistema, t.y. žiniatinklio/internetinis serveris negali tiesiogiai kreiptis į duomenų bazę ir iš jos išgauti pavyzdžiui lentelės duomenys XML pavidalu, todėl 2 paveiksliuke trečia sąsaja yra negalima.

Išnagrinėjus žiniatinklio/internetinio serverio veikimo principą ir galimybės pastebėta, kad įmanoma parašyti tarpinę programą, dar kitaip vadinama kaip tarpininką, tarp žiniatinklio/internetinio serverio ir duomenų bazės valdymo sistemos, kuris galėtu pasijungti prie duomenų bazės valdymo sistemos ir išgauti reikiamus duomenys iš jos, ir šiuos duomenys sutvarkyti ir surašyti į XML dokumentą, ir šį dokumentą gražinti vartotojui į

naršyklę. Todėl kaip pavaizduota 2 paveiksliuke antros, ketvirtos ir penktos sąsajos veikimas yra galimas.

Sekantis šio darbo žingsnis buvo išanalizuota, kaip anksčiau minėta veikimą realizuoti. Išnagrinėta kokias tarpines programas palaiko žiniatinklio/internetinis serveris. Atliekant žiniatinklio/internetinio serverio analizę, pastebėjome kad tarpinę programą galima parašyti dviem budais, t.y. CGI programa arba ISAPI praplėtimas, tačiau kuria iš jų pasirinkti geriau. Šios sistemos vienas iš svarbiausių bruožų yra greitaveika. Kaip greitai duomenys galima išgauti iš duomenų bazės valdymo sistemos. Todėl sekantis žingsnis buvo išnagrinėti ir palyginti kokie pranašumai yra tarp CGI ir ISAPI praplėtimų. Šios pranašumus aptarsime sekančiame skyriuje.

3.2. Pranašumai tarp CGI ir ISAPI praplėtimų (angl. extension)

CGI programos – tai įprasta tinklų sąsaja (angl. Common Gateway Interface), skirta išorinių programų paleidimui iš internetinio/žiniatinklio (angl. Web) serverio. Akivaizdu, kad CGI praplėtimais vadinamos tos programos, kurios naudojasi šią sąsają ir gauna per HTTP protokolą informaciją iš nutolusio vartotojo, apadoruoja šią informaciją ir gražina rezultatą atgal, tokiais būdais kaip: nuorodos į jau egzistuojanti HTML dokumentą, arba kitą objektą, pavyzdžiui toki kaip grafinis vaizdas arba HTML dokumentą, kuris buvo sukurtas dinamiškai. Informacijos perdavimas iš nutolusio vartotojo CGI praplėtimo programai dažniausiai vykdoma tokiu būdu.

HTML dokumente, kuris sukuriamas duomenų įvedimui, skirtas apdoroti įvedama informaciją. Tokiame dokumente dažniausiai būna tam tikri laukai, tokie kaip teksto įvedimas, pasirinkimai ir t.t. Kiekvienam tokiam elementui yra suteikiamas unikalus lauko vardas, ir taip pat reikia paminėti kad dažniausiai tokiose formose yra mygtukas, kuri reikia paspausti užpildžius visus formos laukus.

Kai vartotojas užpildo formą ir paspaudžia nurodyta mygtuką, duomenys perduodami CGI praplėtimui, kurio kelias nurodomas formos apraše. Šios užpildytus laukus CGI programa gauna per HTTP protokolą, tokiu pavidalu „lauko_vardas/reikšmė“.

Gavus ir apdorojus duomenys CGI praplėtimas sukuria HTML dokumentą ir įrašo šį dokumentą į standartini išvedimo įrenginį *stdout.* Po to šis dokumentas automatiškai gražinamas nutolusiam vartotojui.

Tai kaip CGI praplėtimai yra ne kas kito kaip programos, jos turi būti sutransliuotos atitinkamai operacinei sistemai, kurioje dirba jūsų internetinis serveris. Taip pat reikia paminėti kad reikia kurti Win32 konsolinę aplikaciją, tačiau nereikia maišyti ją su MS –DOS

programa, nes tai du skirtingi dalykai. Kuriant CGI praplėtimą Unix sistemai, dažniausiai yra naudojama Perl kalba.

Naudojantis parametru *METHOD* mes galime pasirinkti viena iš dviejų metodų informacijos perdavimui iš formos internetiniam serveriui.

Jeigu pasirenkame GET reikšmę, CGI programa duomenys iš formos gaus per užklausos kintamąjį QUERY_STRING. Jeigu pasirenkame POST reikšmę, CGI programa duomenys iš formos gaus per standartini įėjimo srautą.

ISAPI programų kūrimas. Visas ISAPI programas galima suskirstyti į dvi grupes: ISAPI filtrai (angl. filters) ir ISAPI praplėtimai (angl. extensions). ISAPI praplėtimai pagal savo pobūdį panašus į CGI programas, tačiau jos yra sukurtos ne kaip programos, o kaip dinaminio užkrovimo bibliotekos (angl. dynamic link library) DLL, kurios turi daugiau privalumų.

Taip pat kaip ir CGI programos taip ir ISAPI praplėtimai duomenys gauna iš interneto naršyklės (pavyzdžiui iš internetinės formos, kurią užpildo interneto vartotojas), apdoroja tos duomenys ir atsakymą gražina interneto naršyklei, kaip dinamiškai sugeneruota HTML dokumentą. Taipogi vietoj to, kad skaityti turinį iš kintamosios aplinkos ir standartinio įvedimo srauto, ISAPI praplėtimai duomenys gauna tam specialiai sukurtai funkcijai. Analogiškai vietoj įrašymo išvedamų duomenų į standartini išvedimo srautą ISAPI praplėtimai iškviečia atitinkamas specialias funkcijas.

ISAPI filtrai taip pat realizuojami kaip dinamiškai užkraunamos bibliotekos DLL, tačiau jie turi visai kitą paskirtį. ISAPI filtrai gali kontroliuoti visą duomenų srautą einanti per interneto serverį, HTTP protokolo lygiu. Todėl šios filtrus galima naudoti spręsti tokius uždavinius kaip duomenų kodavimas arba perkodavimas, informacijos suspaudimas. Šie filtrai naudingi sukurti savo filtrą vartotojų pasijungimo prie sistemos arba vartotojų autentifikacijai, o taipogi surinkti statinę serverio apkrovimo informacija.

Kaip ISAPI praplėtimai dirba? Kaip buvo aukščiau minėta ISAPI praplėtimai kuriami kaip dinamiškai užkraunamos bibliotekos DLL. Kreipimasis į šias bibliotekas vykdomas HTML dokumente analogiškai kaip ir CGI programos kreipimasis, t.y. iš formų arba nuorodų sukurtu naudojant tokias žymės (angl. tag) <FORM> ir <A>.

Kai vartotojas kreipiasi į ISAPI praplėtimą, atitinkama DLL biblioteka yra užkraunama į internetinio serverio adresyne erdvę ir tampa šio serverio susidedančia dalimi. Taip kaip ISAPI praplėtimai dirba internetinio serverio proceso ribose, o ne kaip atskiras procesas (taip veikia CGI programos), šis praplėtimas gali naudotis visais resursais kurie yra leidžiami šiam serveriui. Tai labai gerai įtakoja darbą (greitaveiką). Greitaveika išlieka gana aukštame lygyje ir tais atvejais kai šio praplėtimu naudojasi daugelis vartotojų.

ISAPI ir CGI privalumai ir trūkumai. Vienas iš ISAPI praplėtimų privalumų yra tas, pavyzdžiui, jeigu 20 vartotojų vienu metu paleis viena ir ta pačia CGI programą, tai serveryje susikurs 20 procesų – vienas kiekvienam vartotojui. Tai kaip sukūrimas naujo proceso atima gana daug sisteminių resursų iš serverio, tai atspindės greitaveikoje. Bet jeigu 20 vartotojų vienu metu kreipsis į tą patį ISAPI praplėtimą, į serverio proceso atminti bus užkrauta tik viena DLL bibliotekos kopija. Ši biblioteka dirbs daugia gijų režimų. Akivaizdu kad serverio sistemos procesų resursu nebus daug išnaudota.

Lyginant ISAPI praplėtimus ir CGI programas, negalima teigti kad ISAPI praplėtimai yra geresni ar pranašesni nei CGI programos, nes kiekvienos iš jų turi tam tikrų privalumų ir trūkumų.

Taip kaip ISAPI praplėtimai dirba serverio proceso ribose, šie praplėtimai turi būti gerai sukurti ir atstyguoti, nes įvykus ISAPI praplėtime klaidai, ši klaida lemia viso internetinio serverio avarini išjungimą, nes ISAPI praplėtimo dinamiškai užkraunama biblioteka yra užkraunama į serverio proceso ribas. Kas dėl CGI programų, taip kaip jos yra vykdomos kaip atskirais procesais savo adresų erdvėje, tai toks procesas negalės pakenkti interneto serverio darbui. Jeigu CGI programoje yra klaida, tai ši klaida tik užblokuos šios CGI programos procesą, bet neišjungs avariniu būdų viso interneto serverį.

CGI ir ISAPI praplėtimų aplinkos kintamieji yra tie patys, kas palengvina šių programų ir praplėtimų kūrimą.

Taip pat, kaip ir CGI programos ISAPI praplėtimai naudoja du metodus, t.y. GET ir POST, apie kurių skirtumus mes paminėjome aukščiau.

Dar vienas CGI programų privalumas, yra tas, kad šias programas galima rašyti keliomis kalbomis, pavyzdžiui Windows platformai rašoma konsoline kalba, o jei Linux platformai tai dažniausiai rašoma Perl kalba, tačiau ISAPI praplėtimai rašomi visom platformom viena kalba, t.y. C++ valdomo kodo klaba.

Kuriant šias CGI programas arba ISAPI praplėtimus taip pat buvo pastebėta, kad jos galima integruoti (įdiegti) ne tik į Microsoft Internet Information Server, o taip pat ir į Apache žiniatinklio serverį. O tai reiškia, kas mūsų CGI programa arba ISAPI praplėtimai gali dirbti ne tik ant Windows platformos, tačiau ir ant Linux ar Unix platformų, t.y. mūsų programinė įranga, jeigu ja galima taip pavadinti, tampa nepriklausoma nuo platformos.

Todėl buvo atsižvelgta į visus aukščiau paminėtus trukumus ir privalumus, ir nusprendėme kurti ISAPI praplėtimą, nes dirbant su duomenimis, yra svarbu laikas, per kuri laiko tarpą mes tos duomenys išgauname.

Tačiau išnagrinėjus ISAPI praplėtimus buvo pastebėtas vienas gana didelis trūkumas, kad šios praplėtimus reikia būtinai rašyti C++ valdomojo kodo kalba. Bet rašant šį ISAPI

praplėtimą yra komplikuota prieiti prie kompiuterio resursų, ar netgi duomenų bazės valdymo sistemos. Todėl buvo atsižvelgta į daugumą naujų technologijų, kurios padėtu supaprastinti ir pagreitinti šios sistemos veikimą. Buvo nuspręsta visą programos logiką parašyti .NET technologijos pagalba, kuri palengvintu ir pagreitintu prisijungimą ir duomenų išgavimą iš duomenų bazės valdymo sistemos. Taip pat pagreitintu pačio XML dokumento formavimą.

Bet didžiausias šios technologijos trukumas susideda iš to, kad ISAPI praplėtimus mes galime rašyti tik C++ kalba, o čia mes naudosime visai kitą programavimo kalbą, ir visai kitas technologijas. Todėl iškilo dilema, kaip susieti .NET technologijos pagalba parašyta programą su C++ programavimo kalba. Ir vėl buvo išnagrinėtos visos technologijos susijusios su šia problema. Ir buvo pasitelkta Microsoft COM objekto pagalba. Ši technologija padeda suskurti aplikaciją arba programą .NET aplinkoje, ją paversti objektu, prie kurio galima prieiti iš bet kokios kitos programavimo kalbos. Apie .NET ir COM objekto technologijas aptarsime sekančiose skyriuose. [19]

3.3. Microsoft .NET Framework technologija

Microsoft .NET Framework yra Microsoft Windows operacinės sistemos komponentas, sukurtas 2002 metais. Jis suteikia kitoms programoms galimybę naudotis daugybe jau paruoštu įvairių bibliotekų (pvz., duomenų bazių komponentus, formų komponentus...). Be to, šis komponentas ir tvarko programos kodą jos vykdymo metu, jei programa parašyta specialiai šiam paketui (sukompiliuota su CIL suderinamu kompiliatoriumi). Tai reiškia, kad programa vienodai gerai turėtų veikti įvairiose platformose; nėra būtinybės 64-ių bitų procesoriams skirtą CIL programą perkompiliuoti į 32-ų bitų skirtą procesoriams programą. Visa tai atliekama labai greitai ir automatiškai.

.NET Framework yra neatskiriamas "Windows“ komponentas, kuris palaiko kūrimą ir vykdymą naujos kartos programas ir XML interneto paslaugas. .NET Framework apima daugybę bibliotekų, iš anksto koduojamų (angl. prie-coded) sprendimų, kurios padeda spręsti paprastas programavimo problemas, taip pat virtualios mašinos, kurios valdo programų vykdymą, sukurtų tam tikrai aplinkai. .NET Framework yra skirtas įvykdyti šiuos tikslus:

- Užtikrinti nuoseklų objektiškai orientuoto programavimo aplinką, kur objekto kodas yra saugomas lokaliai, ir vykdomas lokaliai bet iškviečiant per interneto naršyklę, arba įvykdomas nuotolinių būdu.
- Teikti kodo vykdymo aplinką, kuri sumažins programinės įrangos diegimo ir versijavimo problemas (konfliktus).

- Teikti kodo vykdymo aplinką, kuri užtikrintu saugu kodo vykdymą, įskaitant kad kodas buvo sukurtas pusiau patikimos arba nežinomos šalies (angl. semi-trusted third party).
- Teikti kodo vykdymo aplinką, kuri šalintu veikimo problemų scenarijus arba interpretuotu terpę.
- Suteikti kūrėjui nuoseklios patirties, įvairių tipų programoms, pavyzdžiui Windows grindžiamas taikomąsias programas ir internete veikiančias taikomąsias programas. Sukurti visokeriopa komunikaciją pramonės standartams, siekiant užtikrinti, kad kodas parašytas .NET Framework, galėtu integruoti su kitais kodais.

.NET Framework sudaro du pagrindiniai komponentai: bendrą atliekama kalbą ir .NET Framework klasės biblioteką. .NET Framework pagrindas yra bendrai atliekama (angl. runtime) kalba. Galite galvoji apie aplinką, kaip agentas, kuris valdo kodą vykdymo laiką, teikiant pagrindines paslaugas, pavyzdžiui, atminties valdymo, pokalbio valdymą, ir nutolusi valdymą, o taip pat vykdyti griežtą saugumą ir kitų formų kodo tikslumą, kad skatinti saugumą. Tiesą sakant, koncepcijos kodo valdymas yra esminis principas. Atliekamas kodas yra žinoma kaip valdomas kodas, o kodas, kuris nėra orientuota į aplinką, yra žinoma kaip netvarkomas kodas. Klasės biblioteką- kitas pagrindinis komponentas. NET Framework, yra išsamūs, objektiškai orientuotas daugkartinio naudojimo tipus, kuriuos galite naudoti norėdami kurti taikomąsias programas, pradedant nuo tradicinių komandų eilutės arba grafinės vartotojo sąsajos (GUI), kurių vykdymo pagrindą pateikė ASP.NET, pvz. internetines formas ir XML interneto paslaugas.

Sąveika su COM objektų. Sąveikavimas tarp naujųjų ir senųjų taikomųjų programų yra būtinas, todėl .NET Framework yra numatyta prieigos funkcija, kad būtų įgyvendinto programos kurios vykdomos ne .NET aplinkoje. Tam yra sukurta COM (angl. Common Object Modeling) komponento technologija. [20]

Bendra kalbos infrastruktūra (angl. Common Language Infrastructure). Pagrindinis .NET sistemos branduolio aspektas yra neperžengti bendros kalbos infrastruktūros, arba CLI. Bendros kalbos infrastruktūrai suteikia neutralios kalbos platformą, programų kūrimui ir vykdymui, įskaitant funkcijas skirtas išskirti tvarkymo funkcijas, šiukšlių surinkimą, saugumo ir sąveikos.

Tarpiniai CIL kodai yra įsikūręs .NET sąrankos (angl. assemblies). Kaip pateikta specifikacijoje, agregatai yra saugomi kilnojamajame vykdomajame formate (angl. Portable Executable), kuriame yra visi Windows platformos exe ir dll failai.

COM Komponentų Objektų Modelis (angl. Component Object Model)

Komponentų objektų modelis yra programinės įrangos sąsajos standartas. Jis naudojamas, kad būtu galimas tarp procesų ryšis ir dinamiško objekto sukūrimas tarp skirtingu programavimo kalbų. COM terminas yra dažnai naudojamas ir programinės įrangos kūrimo pramonei, kuri apima OLE, ActiveX, COM, COM+, DCOM technologijas.

COM esmė kalbos atžvilgiu yra, tą, kad nepriklausomai kokia kalba šis objektas buvo parašytas, kokioje platformoje jis įgyvendintas, mes galime sukurti ryšį tarp mūsų kuriamos programos ir COM objekto.

COM platforma yra gerai suderinta su Microsoft .NET Framework technologija, ir netgi Microsoft sutelkia dėmesį kad kurti .NET programas panaudojant COM objektus, tai gana sudėtinga, bet naudojant šią technologija pagerėja ir pagreitėja taikomųjų uždavinių atlikimo laikas.

Sukuriant COM objektą, jis turi būti užregistruotas Windows registruose, ir taip pat ši COM objektą reikia įtraukti į sąrankas (angl. assembly). Tada šis objektas pasidaro matomas kitoms aplikacijoms ir programoms, t.y. taip kaip sukurta .NET aplikacija (programa) turi būti matoma ir ISAPI praplėtime, kuris yra kuriamas C++ kalba. Tai šią aplikaciją, kaip ir buvo pasakyta aukščiau, mes turime įtraukti į Windows registrus ir Windows sąrankas. Apie sąrankas papasakosime šiek tiek daugiau, tačiau apie šią technologiją Microsoft daug informacijos nesuteikia.

Sąranka (angl. Assembly) tai Microsoft Windows papildoma sistema, kuri dirba lygiagrečiai su Windows registrais, kurios svarbiausia paskirtis yra susieti .NET komponentus. Taip pat jeigu mes parašome .NET programą, ir norime ją paversti COM objektu, ir kad tas COM objektas butu matomas visose kitose programose, mes turime iš pradžių šį .NET komponentą užregistruoti sąrankoje, ir tik po to šis komponentas bus matomas, kitose aplikacijose.

Sąrankos vardas susideda iš keturių dalių:

1. Trumpo failo vardo. Windows tai vardas iš PE tik be plėtinio.
2. Kultūra. Tai RFC1766 regiono identifikatorius.
3. Versijos. Tai yra taškinis numeris, kurį sudaro keturios vertes.
4. Viešasis raktas. Tai yra 64 bitų maišos raktas, kuris atitinka privatu raktą, naudojama sąrankos pasijungimui. Tai dar vadinama stipriu vardu.

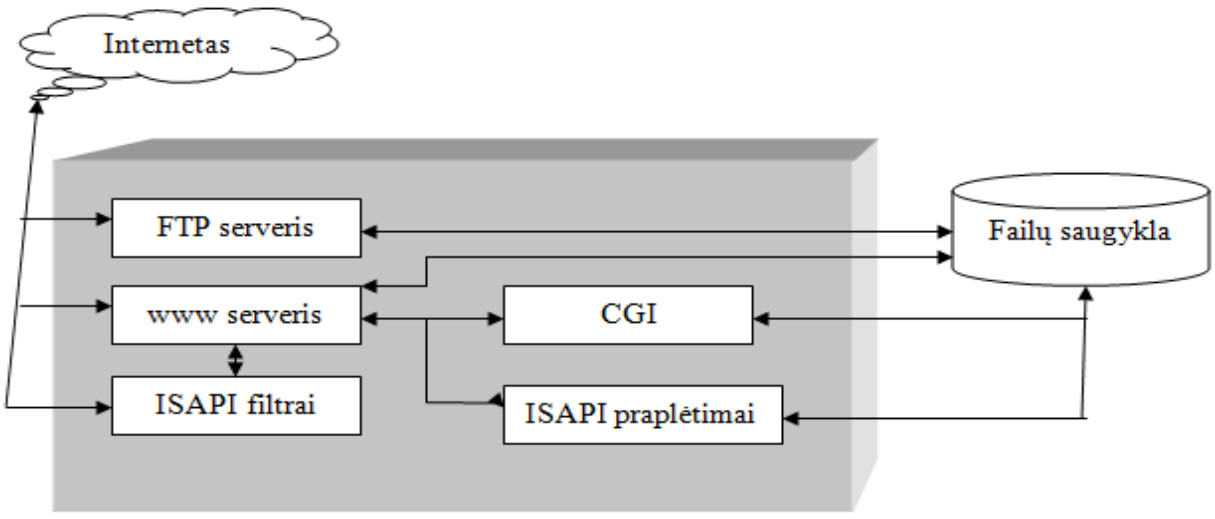
Tam kad pradėti kurti ISAPI praplėtimus, kurie kuriami C++ kalba, buvo išnagrinėta, kad iš šios kalbos priėjimas prie atitinkamų Windows ir duomenų bazės komponentų yra komplikuotas dalykas, o taip kaip buvo nuspręsta ISAPI praplėtimą kurti ant Windows

platformos, taip pat buvo nuspręsta pasinaudoti keliomis naujomis technologijomis, tokiomis kaip .NET Framework, kuris suteikia priėjima prie atitinkamų komponentų, ir kuri galima programuoti skirtingomis kalbomis, tokiomis kaip, VB.NET, C# ir kitos .NET kalbos. Bet susidurta su kita problema, nes ISAPI praplėtimus galime rašyti tik C++ kalboje, tačiau ir iš šios padėties buvo surasta išeitis. Buvo nuspręsta sukurti visą .NET komponentą VB.NET kalba ir šį komponentą užregistruoti Windows sąrankoje, o taip pat ir Windows registruose, kad šis komponentas pasidarytu matomas ISAPI praplėtimo programai, bet prieš tai dar šį komponentą reikia paversti COM (angl. Component Object Model) objektu, ir tada kuriant ISAPI praplėtimą belieka pasinaudoti keliomis specialiomis komandomis ir C++ kalboje inicializuoti (užkrauti) sukurta COM objektą. Sekantis žingsnis susideda iš to kad iš ISAPI praplėtimo siunčiamas fizinis XML dokumento kelias į kurį kreipiasi vartotojas iš naršyklės ir taip pat perduodami parametrai, jeigu tokie buvo surašyti naršyklės adreso eilutėje po klaustuko simboliu.

3.4. Žiniatinklio/internetinis serveris (angl. Internet Information Service)

Internet Information Services (IIS) - anksčiau vadinta "Internet Information Server - tai interneto paslaugų servisų rinkinys, sukurtas "Microsoft" korporacijos, skirtų naudoti su Microsoft Windows operacinėmis sistemomis. Žiniatinkio/interneto serveris pasaulyje užima antrąją vietą, tarp populiariausiu interneto serverių. Šiuo metu lyderio pozicijas užima Apache HTTP žiniatinklio serveris, kurio vienas iš privalumų yra tas, kad jis dirba ne tik ant Windows, bet ir ant Linux platformų.

Žemiau pateiksime žiniatinklio/internetinio serverio struktūrą, tačiau šią struktūra mes atvaizduosime iš savo nagrinėtos pusės, nes Microsoft korporacija tokiu duomenų neduoda.



3 pav. Žiniatinklio/internetinio serverio sandūra

Žiniatinklio/internetinis serveris susideda iš kelių pagrindinių dalių:

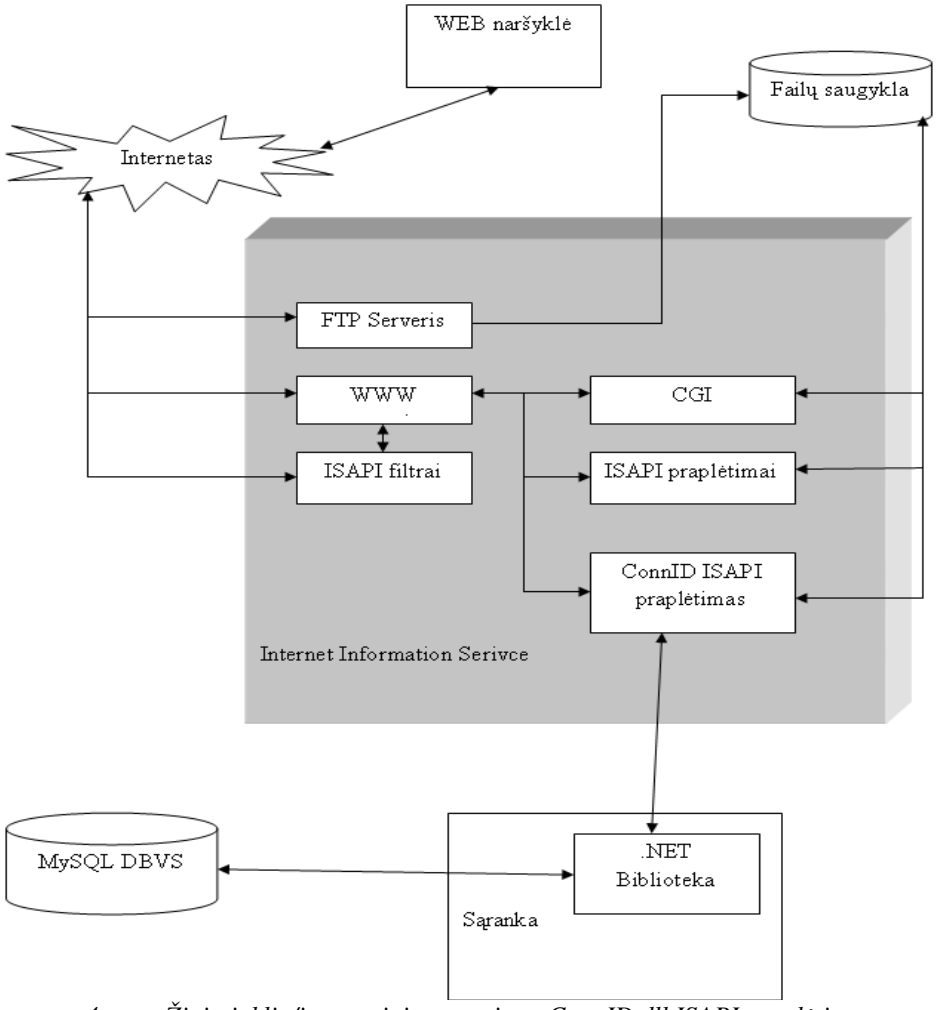
- FTP serverio (failo serveris)
- WWW serverio (žiniatinklio serveris)

ISAPI filtrai, kaip aukščiau minėjome, dažniausiai būna skirti apdoroti prisijungimo duomenys jeigu yra sukurta kitokia pasijungimo sistema. Arba ISAPI filtrą dar dažnai naudoja, tam kad kai pasijungia vartotojas, nuskaito jo IP adresą ir pagal tą adresą nustato iš kokios šalies vartotojas pasijungęs ir iškarto užkrauna puslapį, kuris parašytas suprantama, t.y. tos šalies kalba.

CGI programos, kaip ir ISAPI praplėtimai, buvo aprašyti aukščiau, todėl čia daug apie jos nekalbėsime.

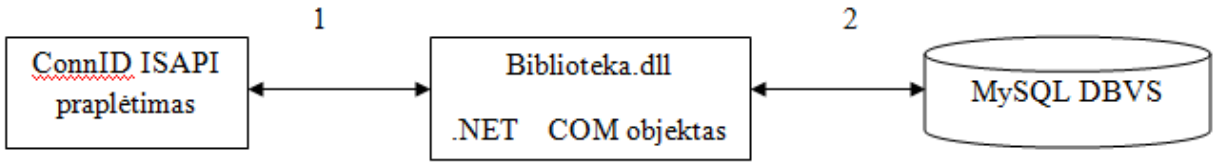
Žiniatinklio/internetinis serveris dirba tokiu pagrindu, kada iš kompiuterio vartotojas kreipiasi į žiniatinklio/internetinį serverį, iš pradžių patikrinama kokiu portu kreipiamasi, pagal nutylėjimą, jeigu kreipiamasi 80 portu, tada visą veikimą apdorojąwww serveris ir/arba ISAPI filtrai, tai priklauso nuo to ar yra naudojama kokia tai nors autentifikacija. Kai gaunama užklausą, www serveris tikrina, kas tai per užklausa, koks kreipimasis į failo plėtinį, mūsų atvejų mes kreipiamės į XML dokumentą, tai tada serveris pradeda tikrintis ar išvis toks failas egzistuoja serveryje, t.y. failų saugykloje, po to sekantis žingsnis serveris tikrina koks servisas yra atsakingas už šio dokumento apdorojimą. Užkraunamas reikalingas servisas į serverio proceso erdvę. Galiausiai vykdomas failo apdorojimas pagal servise užprogramuota logika. Jeigu į serverį kreipiamasi 21 portu, tai tada visą tolimesnį veikimą apdoroja ftp servisas. FTP servisas – pasižymi to kad per šį servisą mes galime nusiusti/atsisiusti failą į/iš serverio. Tačiau detaliau šio serviso mes nenagrinėsime. Nes tai nėra šio magistrinio darbo tikslas.

Dabar apibūdinsime kaip ši sistema veikia, ir su kiekvienu žingsniu šį veikimą detalizuosime. Pirmiausia pateiksime abstraktu šios sistemos veikimo principą, ir po to kiek vieną iš elementų bus aprašytas ir pavaizduotas detaliau.



4 pav. Žiniatinklio/internetinis serveris su ConnID.dll ISAPI praplėtimu

Visu pirma vartotojas turi patalpinti XML failą į žiniatinklio/internetini serverį, kuriame yra pagal atitinkama XML dokumento sintaksę parašyta užklausą į duomenų bazes lentelę. Sekančiu žingsniu vartotojas iš savo internetinės (angl. web) naršykles kreipiasi į serverį su užklausą, kurioje yra nurodytas XML failas. Žiniatinklio/internetinis serveris apdoroja užklausą, vykdomas patikrinimas į kuria serverio dalį servisą ši užklausa kreipiasi, ar tai www servisas, ar tai ftp servisas. Mūsų atveju pagal šią užklausą kreipiamasi į www servisą, toliau žiniatinklio/internetinis serveris patikrina kurį serverio funkcija apdoroja failą su tam tikru plėtiniu, mūsų atveju XML dokumentą apdoroja ConnID ISAPI praplėtimas, šis ir kiti programų susiejimai yra atliekami konfigūruojant žiniatinklio/internetinį serverį. Po to kai serveris „žino“ kad XML failą turi apdoroti ConnID ISAPI praplėtimas, serveris automatiškai šį praplėtimą užkrauna į savo darbinę aplinką (procesą). Toliau detalizuosime kaip veikia mūsų parašyta programa.

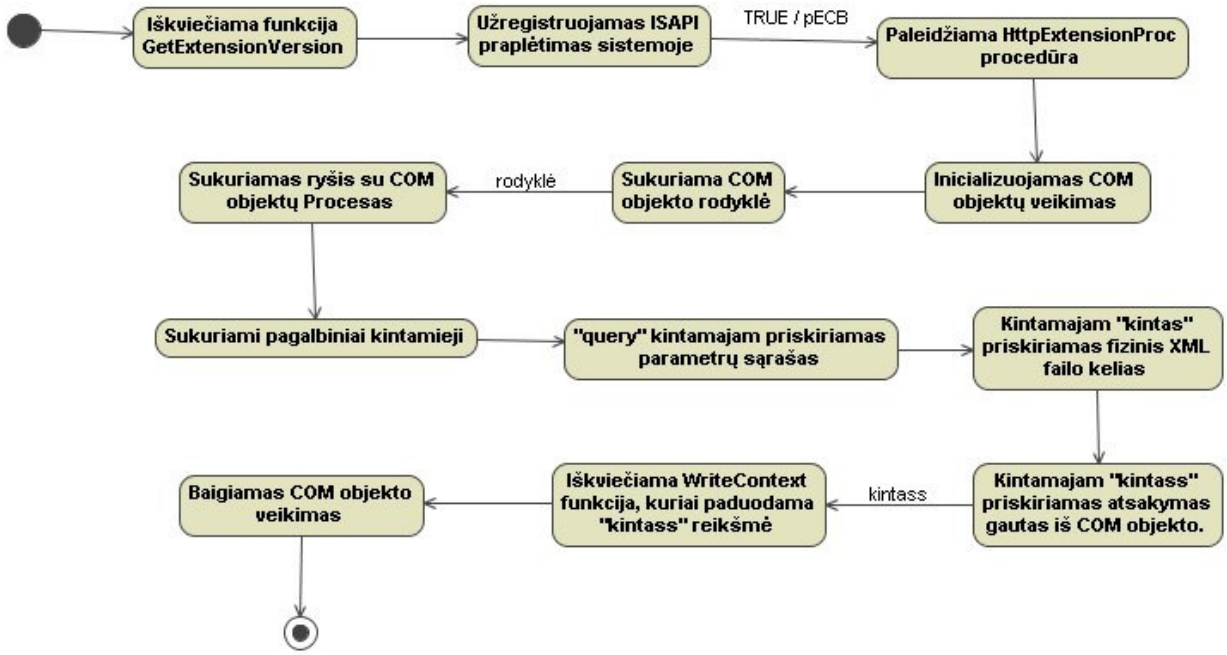


5 pav. Abstraktus programos veikimo principas

Kada į žiniatinklio/internetinio serverio aplinka yra užkraunamas ISAPI praplėtimas, kuris nuskaito iš vartotojo žiniatinklio/internetinės naršyklės adresą ir taip pat jei nurodyti parametrai, kurie yra aprašomi po klaustuko simboliu. Šie parametrai reikalingi tam, kad perduoti parametrus saugomosioms duomenų bazės procedūroms, pavyzdžiui iš duomenų bazes išrinkti įrašus iškvietus saugomąją procedūrą ir padavus jai šios parametrus. Tačiau šios parametrus galima perdavinėti ne tik per naršyklės užklausos eilutę, bet tos pačius parametrus galima aprašyti pačiame XML faile, į kurį kreipiasi vartotojas iš savo naršyklės. Kada ConnID ISAPI praplėtimas, t.y. dinamiškai užkraunama biblioteka, nuskaito virtualu kelią iš vartotojo naršyklės adreso eilutes ir šį kelią ConnID ISAPI praplėtimas konvertuoja į fizinį dokumento kelią pasinaudojus atitinkama funkcija, t.y. perduoda tikra fizini kelia kur randasi užklausiamasis XML dokumentas. Sekančiu žingsniu, šis fizinis kelias paduodamas Biblioteka.dll COM objektui, pasinaudojant COM objektu sąsajos ryšiu, Toliau sukurtasis COM objektas vykdo užprogramuota logiką, t.y. nuskaito XML failą, ieško atitinkamų žymių, kad nuskaityti parametrus, jeigu tokie yra aprašyti XML dokumente, tolimesnis COM objekto veikimas nuskaito kitas žymės, tokias kur yra aprašyta SQL užklausa į duomenų bazės valdymo sistemą, galiausiai COM objektas kreipiasi į duomenų bazės valdymo sistemą, pasinaudojus .NET aplinkai sukurtu sujungimo sąsajos ryšiu, siunčia duomenų bazei SQL užklausą ir jeigu reikia parametrus, kurie yra aprašyti XML dokumente, duomenų baze pagal jai pateikta SQL užklausą atlieka ten atitinkamus veiksmus ir gražina atgal duomenys pasinaudojus to pačiu .NET sąsajos ryšiu. Tada Biblioteka.dll COM objektas pradeda kurti ir formuoti XML dokumentą, ir jau pilnai suformuota XML dokumentą gražina vartotojui į naršyklę, taip po gi pasinaudojus COM objektų sąsajos ryšiu. [21]

Dabar aptarsime kiekviena aukščiau paminėta elementą atskirai. Kaip šie elementai veikia, kokia logika juose yra užprogramuota, pateiksime algoritmus.

Pradėkime nagrinėti pati pirmą objektą, t.y. ISAPI praplėtimą ConnID, kuris ir pradeda visą šios sistemos veikimo darbą.



6 pav. ConnID ISAPI praplėtimo veikimo algoritmas

ConnID ISAPI praplėtimo programa yra ganėtinai nesudėtinga, nes joje atliekami tik keli veiksmai. Šioje programoje, nėra labai sudėtingų skaičiavimų, nėra kreipimosi į duomenų bazės valdymo sistemą, šiame ISAPI praplėtime yra aprašomos tiktai dvi pagrindinės procedūros viena iš tokiu funkcijų yra skirta tam, kad nustatyti ISAPI praplėtimo versija ir užregistruoti šį praplėtimą sistemoje. Ši funkcija yra iškviečiama pati pirmiausia, kai tik yra užkraunamas ISAPI praplėtimas. Šios funkcijos veikimas yra standartinė dalys kuriant ISAPI praplėtimus. Ši, GetExtensionVersion procedūra nustato ISAPI praplėtimo versiją, ir šį ISAPI praplėtimą užregistruoja sistemoje. Sekanti galima teigti pati svarbiausia šio ISAPI praplėtimo procedūra yra HttpExtensionProc, šioje procedūroje yra aprašomas visas ISAPI praplėtimo veikimo principas (logika). Ši procedūra, šios sistemos atveju, inicializuoja COM objekto veikimą, t.y. inicializuoja COM objektų bendravimo sąsają. Sekančių žingsniu sukuriama rodyklę į patį COM objektą. Kada jau yra sukurta rodykle į COM objektą, yra inicializuotas COM objekto veikimas. Kada yra inicializuotas COM objekto veikimas, toliau logiška ir pradėti naudoti COM objekto funkcijas ir procedūras, tam dar yra sukuriami keli teksto tipo kintamieji, kurie skirti tam, kad išsaugoti atsakymus, kurios gražina tam tikros ISAPI praplėtimų standartinės funkcijos. Šiame ISAPI praplėtime pasinaudosime tik dviem standartinėm funkcijom, tokiomis kaip lpszQueryString, kuri gražina naršyklės adreso eilutes parametrus, kurie yra nurodomi po klaustuko simboliu, aišku jei šie parametrai buvo pateikti. Nes, pavyzdžiui, kviečiant išrinkimo SQL komandą, kuri išrinktu iš duomenų bazės lentelės visus įrašus nereikia perduoti jokiu parametrų. Sekanti funkcija kuri naudojama šiame ISAPI praplėtime yra lpszPathTranslated, ši funkcija nuskaito virtualu adresą iš vartotojo

internetinės naršyklės ir šį adresą konvertuoja į fizinį kelią, t.y. perduoda vienam iš tekstinių kintamųjų pilną kelią iki užklausiamo XML failo, kuris yra saugomas žiniatinklio/internetiniame serveryje. Tačiau atliekant šių funkcijų analizę pastebėta, kas šios funkcijos nieko nesiskiria nuo CGI programų naudojamų funkcijų, netgi šios funkcijos taip pat apsirašo. Ir paskutinis tekstinis kintamasis kuris naudojamas šiame ISAPI praplėtime yra skirtas tam, kad išsaugoti duomenys gautus iš COM objekto, t.y. Biblioteka.dll, kuris atlieką visą reikiamą logiką. Apie šio COM objekto veikimą aptarsime šiek tiek vėliau. Šiame kintamajame yra išsaugotas visas jau suformuotas XML dokumentas. Sekančiu žingsniu mes šį kintamąjį paduosime vienai ISAPI praplėtime parašytai funkcijai WriteContext, kuri gražiną kintamojo turinį vartotojui į naršyklės langą atsakymą, šiuo atveju vartotojui bus gražintas suformuotas XML dokumentas, priešingu atveju, jeigu buvo įvykus klaida, pvz. buvo XML dokumento struktūros klaidu, tai vartotojui bus į naršyklės langą gražintas klaidos pranešimas, ir klaidos kodas. Naudojant WriteContext funkciją, dar reikia paminėti, kad perduodama rodyklę, ši rodyklė perduodama tam, kad sistema žinotu kuriam procesui pateikti atsakymą. Taip pat šiai funkcijai reikia perduoti, kokio tipo bus išvedamas formatas, ar tai string, ar integer, ar kiti tipai, ir aišku kad paduodamas kintamasis, kuriame yra išsaugotas atsakymas, šio atveju XML suformuotas dokumentas.

Dar vienas svarbus momentas, kad visos ISAPI praplėtime naudojamos funkcijos yra tokio tipo, kurios gražina atsakymą „taip“ jeigu viskas įvyko be klaidų, bet jei įvyko klaida gražina reikšmę „ne“, ko pasėkoje yra sustabdomas ISAPI praplėtimo veikimas. Pavyzdžiui pagrindinėje funkcijoje, kur aprašomas ISAPI praplėtimo veikimas, yra gražinama reikšmė „HSE_STATUS_SUCCESS“, ši reikšmė yra lyginama pačiame žiniatinklio/internetiniame serveryje, t.y. patikrinama ar iš ISAPI praplėtimo yra gauta ši reikšmė, jei taip, tai tada ISAPI praplėtimas savo darbą atliko be klaidų, ir vartotojui bus pateiktas atsakymas.

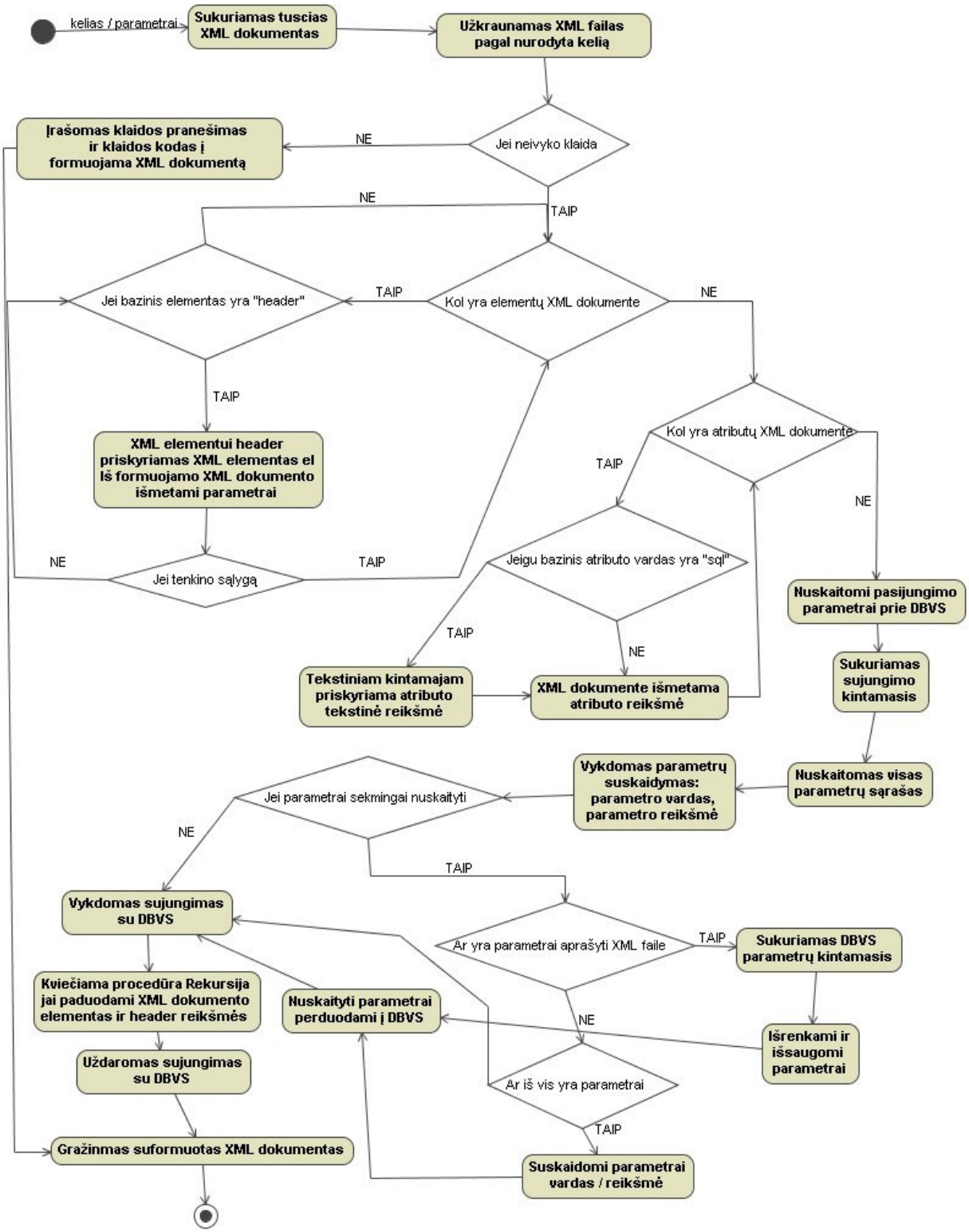
Kaip ir buvo paminėta aukščiau ISAPI praplėtimas mums reikalingas tam, kad išgauti duomenys iš vartotojo naršyklės, ir pasinaudojus atitinkamomis funkcijomis šiuos duomenys konvertuoti į reikiama formą, ir reikiamus duomenys paduoti COM objektui, kuris atliks tolimesnius reikiamus skaičiavimus. Apie šio COM objekto veikimą aptarsime žemiau ir pateiksime veikimo algoritmus.

COM objekto Biblioteka.dll veikimas

Visu pirma reikia paminėti, kad šis COM objektas skirtas nuskaityti SQL užklausą ir jeigu paduodami parametrai iš paduodamo XML dokumento, po to išgautą užklausą su parametrais nusiusti duomenų bazės valdymo sistemai, iš jos išrinkti duomenys, atitinkančius užklausai ir perduodamiems parametrams, suformuoti XML dokumentą į kurį reikia surašyti pagal XML standartą išrinktus iš duomenų bazės lentelės duomenys. Dar reikėtu paminėti, kad šis COM

objektas yra sudarytas iš dviejų procedūrų ir vienos funkcijos. Apie šių procedūrų ir funkcijų veikimo logika, pateiksime žemiau ir aptarsime kam kiekvienas šio COM objekto elementas yra skirtas, kokį veikimą atlieką.

Pradėkime aptarinėti nuo pagrindinės funkcijos, „data“. Žemiau pateiksime šios funkcijos veikimo algoritmą.



7 pav. COM objekto funkcijos „data“ veikimo algoritmas

Šį funkcija pradeda veikimą, tada kai yra ISAPI praplėtime inicializuotas COM objektas ir iš ISAPI praplėtimo per atitinkama COM objekto rodyklė yra kreipiamasi į šią funkciją, kuriai yra perduodamas fizinis kviečiamo XML failo kelias ir jeigu yra nurodyta parametrų sąrašas. Po to, kai jau yra vykdoma COM objekto „data“ funkcija, yra sukuriamas tuščias XML dokumentas kuriam priskiriamas XML užkrautojo failo turinys, kuris palaipsniui bus formuojamas, t.y. išmetamos nereikiamos reikšmės, įterpiami atsakymai ir galiausiai suformuotas XML dokumentas bus gražintas ISAPI praplėtimui, bet apie tai vėliau. Po to, kai šioje funkcijoje yra sukurtas tuščias XML dokumentas, sekantis šios funkcijos žingsnis, užkrauti pagal duotąjį fizini kviečiamo XML failo kelią, XML failą. Kada šis XML failas yra užkraunamas, šioje funkcijoje vykdomas patikrinimas ar taisyklinga yra užkrautojo failo struktūra, t.y. ar failas atitinka XML standartui, ar yra aprašytos visos reikiamos žymės. Jeigu vykdymo metu nebuvo įvykusi klaida, t.y. XML failas taisyklingas, pradedamas ciklas, kol XML dokumente yra elementų. Jeigu XML faile yra elementų, yra tikrinamas ar bazinis XML failo elementas nėra „header“, jeigu surandama, kad bazinis XML dokumento elementas yra „header“, sekančiu žingsniu yra vykdomas priskirimas, t.y. tarpiniam XML elementui yra priskiriamas „header“ elementas ir taip pat iš formuojamo XML dokumento yra išmetama „header“ žymė (angl. tag). Tolimesnis šios funkcijos žingsnis patikrinti ar buvo įvykdyta aukščiau paminėta sąlyga, jei ši sąlyga buvo įvykdyta, tada yra išeinama iš ciklo, priešingu atveju vėl yra vykdomas ciklas. Kada užkrautame XML dokumente pasibaigia XML elementai, tada yra vykdomas sekantis žingsnis, kuris vėl pradeda vykdyti ciklą, tol kol XML dokumente yra atributų. Kai vykdomas šis ciklas, yra tikrinama sąlyga ar XML dokumento bazinis atributas nėra „sql“, jeigu surandama, kad bazinis XML failo atributas yra „sql“, tada tekstiniam failui yra priskiriama atributo reikšmė, t.y. šio atveju yra priskiriama ROOT žymes XSL transformacijos kelias. Kai transformacijos kelias yra priskirtas kintamajam, tolimesnis šios procedūros veiksmas susideda iš to, kad iš formuojamo XML dokumento yra išmetama ROOT transformacijos šaka, ir paliekama tiktai ROOT šakninę žymę. Jeigu vykdant sąlyga nebuvo rasta atributų, tada iš formuojamo XML dokumento bus išmestos šakninio elemento atributo žymė, kuri nurodo, kad tai yra XML dokumentas. Šis visas veikimas bus atliekamas, tol kol XML dokumente bus atributu, t.y. kol bus tenkinama ciklo sąlyga, kada ciklo sąlyga nebus patenkinta, ši funkcija pradės inicializuoti pasijungimą prie duomenų bazės valdymo sistemos, t.y. bus nuskaityti parametrai, reikalingi kad pasijungti prie duomenų bazės valdymo sistemos. Kada šie parametrai bus nuskaityti, bus suformuotas pasijungimo kintamasis (angl. Connection String), kuris bus reikalingas vėliau, kad įvykdyti pasijungimą prie duomenų bazės valdymo sistemos. Sekantis funkcijos veikimo žingsnis susideda iš to, kad sukuriamas tekstinių kintamųjų masyvas, kuriam priskiriamos

suskaidytos parametrų reikšmės, t.y. kada yra aprašyti internetines naršyklės lange parametrai po klaustuko simbolio, yra nurodomas parametro vardas lygybės ženklas ir parametro reikšmė, tam kad nurodyti sekanti parametrą, reikia įrašyti „ampersand“ simbolį („&“) ir po to vėl aprašinėti parametro vardas lygybės simbolis ir parametro reikšmė. Tai šiame sukurtame masyve yra išsaugomos parametro pavadinimo ir reikšmės reikšmės. Tolimesnis šios funkcijos veikimas susideda iš to, kad tikrina ar buvo iš interneto naršyklės nuskaityti parametrai, jeigu parametrai buvo sėkmingai nuskaityti, tada dar yra vykdoma viena sąlyga ar ne buvo parametrai aprašyti paduodamajame XML dokumente, tam užtikrinti yra vykdomas ciklas, kuris vykdomas tol, kol XML dokumente yra atributų, ir yra vykdoma tikrinimo sąlyga, ar atributo bazinis vardas nėra lygus „name“, jeigu XML dokumento atributo vardas yra lygus „name“, tada sukuriamas parametras kuris priskiriamas duomenų bazės parametro kintamajam. Sekanti sąlyga užtikrina, ar buvo surastas parametras, jeigu parametras buvo surastas, sekančios sąlygos žingsnis užbaigti ciklą, ir pereiti nuskaitinėti sekanti parametrą, jeigu daugiau parametrų nebuvo rasta, užbaigti parametrų išrinkimo iš paduodamo XML dokumento ciklo. Tačiau, jei parametrai buvo perduoti tik per interneto naršyklės adreso eilutę, tada yra sukuriami dar keli papildomi kintamieji, kurie skirti išskaidyti parametrą į dvi dedamąsias dalis, t.y. parametro vardą ir parametro reikšmę. Kad šis veiksmas butu įvykdomas duomenų bazės valdymo sistemos komandines eilutes kintamajam yra priskiriamas taisyklingai suformuotas parametras. Tolimesnė sąlyga užtikrina ar tikrai buvo suformuoti parametrai, ar iš internetinės naršyklės eilutes, ar iš XML paduodamo failo, jei parametrai buvo aprašyti, tada į duomenų bazės valdymo sistemos komandinės eilutės kintamąjį paduodamas visas taisyklingai suformuotas parametrų sąrašas. Kada užbaigiamas ciklas, tada vykdomas sujungimas su duomenų bazės valdymo sistema, kviečiama procedūra „Rekursija“ kuriai yra paduodami kuriamo XML dokumento elementai ir parametrų reikšmės. Apie šios procedūros veikimą ir jos algoritmą aptarsime šiek tiek vėliau. Tolimesnis „data“ funkcijos veikimo žingsniai susideda iš to, kad yra uždaromas sujungimas su duomenų bazės valdymo sistema, tačiau reikėtu nepamiršti, kad pradžioje mes tikrinome ar paduotas (užkrautas) XML dokumentas yra struktūriškai taisyklingas, priešingu atveju, jeigu sąlyga yra netenkinama į formuojama XML dokumentą, yra paduodamas klaidos pranešimas ir klaidos kodas. Ir pats paskutinis šios funkcijos veikimo žingsnis yra tas, kad jau pilnai suformuotas XML dokumentas yra gražinamas šio atveju ISAPI praplėtimui, kuris ir inicializavo šios funkcijos veikimą.

Toliau aptarsime, kaip veikia ir kam skirta procedūra „Rekursija“. Ši procedūra skirta tam, kad nuskaitant XML paduodama dokumentą, vykdyti rekursivini žymių nuskaitymą. Pavyzdžiui, mes suformuojame XML dokumente, kur vienoje vietoje mes norime matyti iš

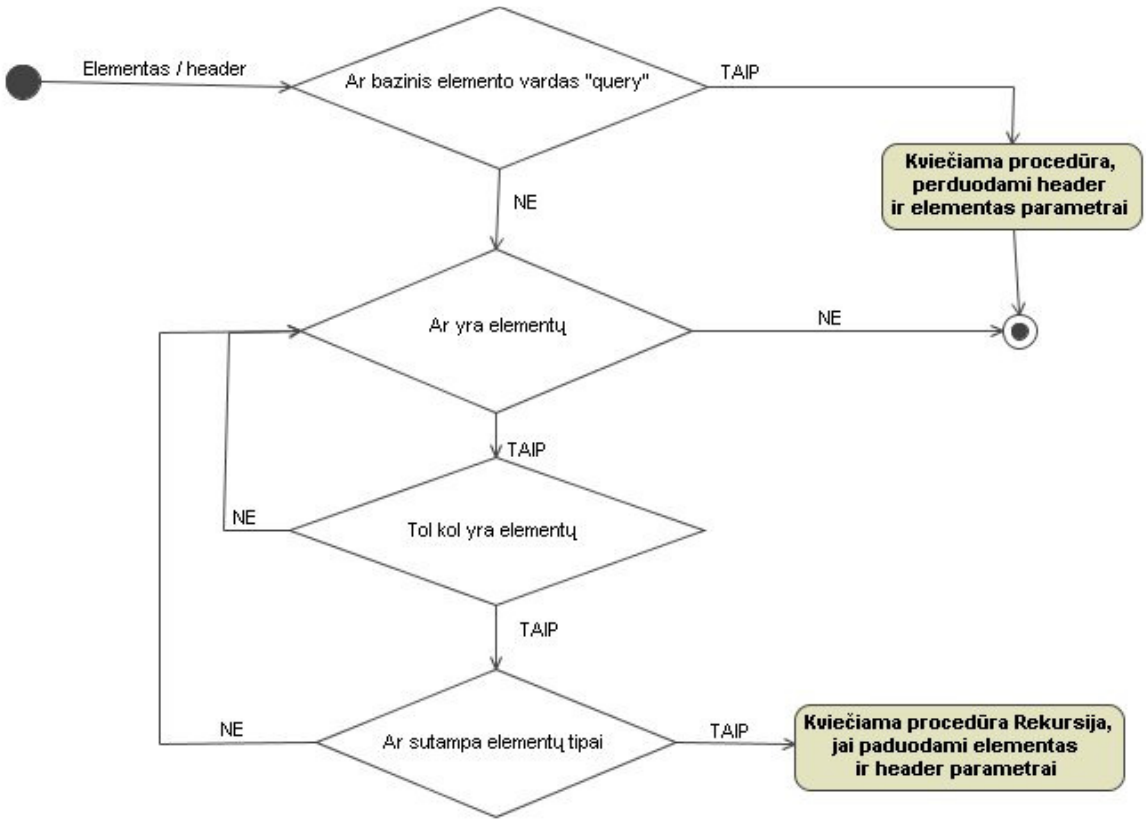
duomenų bazės lentelės išrinktus įrašus, kurie tenkina kažkokia SQL užklausą, o žemiau, visai kitame hierarchijos lygmenyje, norime matyti visai kita užklausą. Todėl ši procedūra vykdo žymių nuskaitymą gilinantis į vis gilesnes žymias. Šį apibūdinimą pateiksime XML failo struktūros pavyzdžiu.

Kaip iš žemiau pateikto paveiksliuko galite matyti, kad pirmame hierarchijos lygyje yra aprašytas ROOT šakninis elementas, sekančiame hierarchijos lygyje yra aprašyta XML dokumento žymė „<aaa>“, trečiame XML dokumento hierarchijos lygyje yra aprašyta „<bbb>“ žymė, ir ketvirtame XML dokumento hierarchijos lygyje yra aprašyta „<ccc>“ žymė. Todėl kai yra vykdoma „Rekursija“ procedūra, iš pradžių ši procedūra nuskaito SQL užklausą kuri yra aprašyta antrame hierarchijos lygyje, t.y. kur yra „<aaa>“ XML dokumento žymė, tolimesnio ciklo vykdymo metu yra nuskaitomi vis gilesni hierarchijos lygiai, t.y. bus vykdomas toks šios procedūros veikimas, nuskaitoma „<aaa>“ žymė, toliau yra nuskaitoma „<bbb>“ žymė ir galiausiai bus nuskaityta „<ccc>“ XML dokumento žymės.

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql" sql:xsl='qquery.xsl'>
  <sql:header>
    <sql:param name='sk'>101</sql:param>
    <sql:param name='Vard'>Bla2</sql:param>
    <sql:param name='Telef'>Qva2</sql:param>
  </sql:header>
    <aaa>
    <sql:query>
        CALL selectas
    </sql:query>
        <bbb>
        <sql:query>
            SELECT * FROM table1
        </sql:query>
            <ccc>
            <sql:query>
                SELECT Vard FROM table1
            </sql:query>
            </ccc>
        </bbb>
    </aaa>
</ROOT>
```

8 pav. XML dokumente rekursijos pavyzdys

Rekursijos procedūros algoritmą pateiksime žemiau.

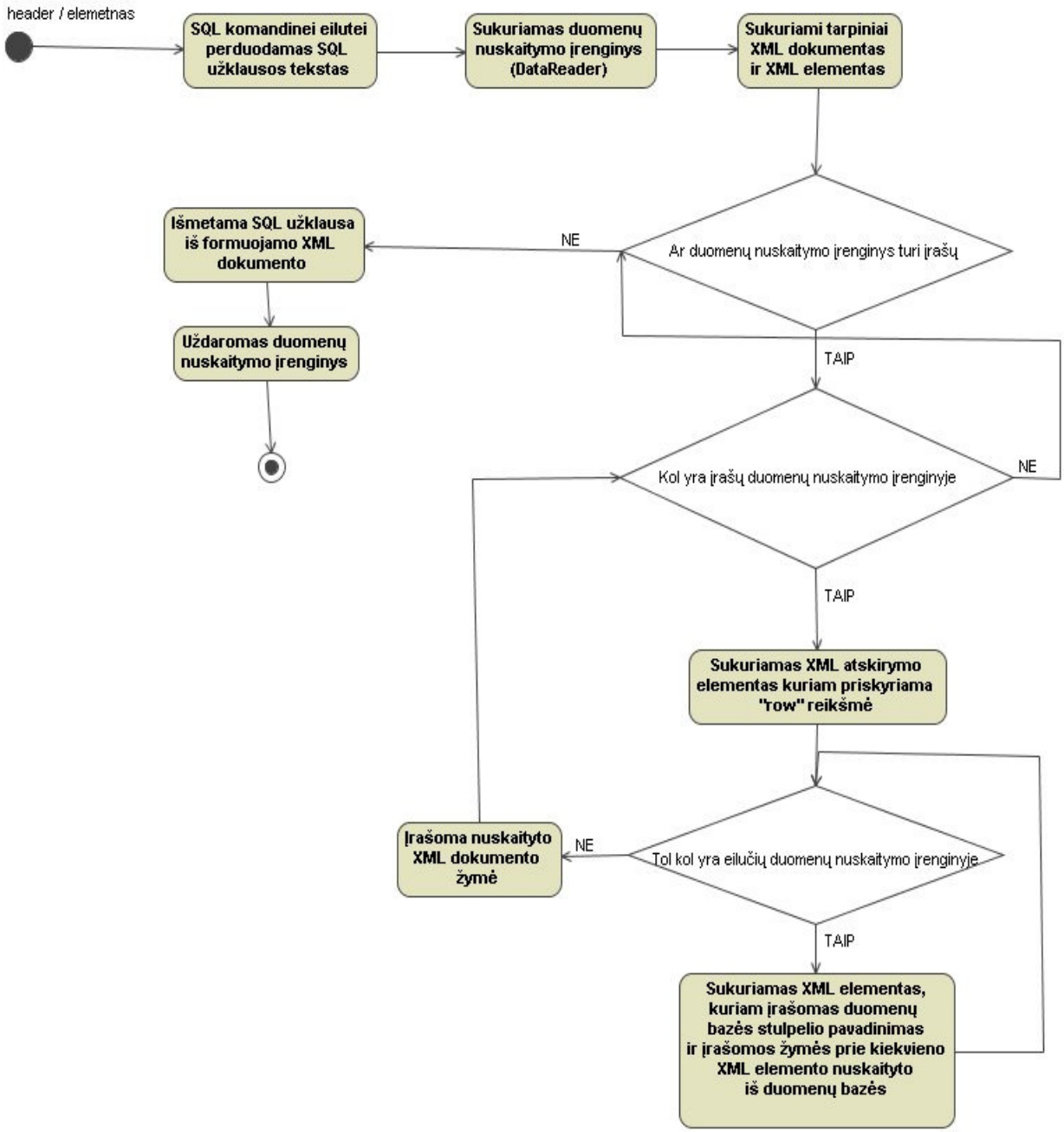


9 pav. Procedūros „Rekursija“ veikimo algoritmas

Procedūrai „Rekursija“ yra paduodami šie parametrai: XML dokumento elementai ir parametrų aprašymo srities žymė. Vykdant procedūrą yra tikrinama sąlyga, ar bazinis XML dokumento elemento vardas yra „query“, jei tenkinama ši sąlyga, tada kviečiama procedūra „proce“, kuriai perduodami šie parametrai: parametrų srities aprašymo žymė ir XML dokumento elementus. Apie procedūros „proce“ veikimą aptarsime vėliau ir pateiksime šios procedūros veikimo algoritmą. Taip pat reikia paminėti, kad tenkinant sąlygą ir įvykdžius kreipimąsi į „proce“ procedūra, „Rekursija“ procedūra užbaigia darbą, kas yra gerai matyti algoritme. Tačiau jai vykdant „Rekursija“ procedūrą yra netenkinama sąlyga, kad dar nėra surastas bazinis XML dokumento elementas, kurio vardas „query“, tada ši procedūra vykdo tolimesne sąlyga, t.y. sąlygą ar yra XML dokumente elementų, jei yra tenkinama ši sąlyga, tada inicijuojamas ciklas tol, kol yra XML elementų, jeigu šis ciklas tenkina sąlygą, tada dar yra įvykdomas vienas patikrinimas ar sutampa XML dokumento elementų tipai, jeigu ši sąlyga tenkina, t.y. XML dokumento elementai sutampa, tada vėl yra kviečiama procedūra „Rekursija“, kuriai jau perduodami nauji XML dokumento elemento parametrai ir parametrų aprašymo srities žymė. Tačiau, kai sąlyga, ar sutampa XML dokumento elementų tipai yra netenkinama, tada yra kreipiamasi vėl į sąlyga ar dar yra elementų XML dokumente. Tas pats yra vykdoma, kada yra netenkinama ciklo sąlyga, kol yra elementų XML dokumente, tada ir šioje vietoje yra kreipiamasi į sąlyga, tam kad patikrinti ar tikrai daugiau nėra elementų XML

dokumente. Kada yra netenkinama ir ši sąlyga, t.y. „ar yra elementų XML dokumente“, tada ši procedūra baigia savo darbą.

Dabar aptarsime kam yra skirta ir kaip veikia procedūra „proce“. Ši procedūra skirta, tam kad taisyklingai suformuoti XML dokumentą. T.y. šioje procedūroje yra vykdomas nuskaitymas iš duomenų nuskaitymo objektą (angl. Data Reader). Šioje procedūroje yra vykdomas nuskaitymas iš duomenų bazės valdymo sistemos lentelės (-ių) duomenų, ir šios lentelės stulpelių pavadinimų. Pagal kurios vėliau bus suformuotas taisyklingas XML dokumentas. Tačiau reikėtu dar paminėti, kad yra du galimi XML dokumento formavimo būdai: atributų formavimo būdas ir elementų formavimo būdas. Šioje programoje naudosime elementų formavimo būdą, nes naudojant šį formavimą, geriau matosi kur yra žymės ir kur yra duomenys. Žemiau pateiksime šios procedūros veikimo algoritmą, ir detaliau aptarsime, kas kaip veikia, kas po ko seka.



10 pav. Procedūros „proce“ veikimo algoritmas

Ši procedūra iškviečiama „Rekursija“ procedūros veikimo metu, ir šiai procedūrai yra perduodami šie parametrai: XML dokumento elementai ir XML dokumento parametrų aprašymo žymė. Kada yra inicializuojamas šios procedūros veikimas, duomenų bazės valdymo sistemos komandinės eilutės kintamajam yra perduodama SQL užklausą, kuri buvo išgauta iš paduodamo XML dokumento elementų. Sekantis šios procedūros veikimo žingsnis susideda iš to, kad yra sukuriamas duomenų bazės valdymo sistemos duomenų nuskaitymo objektas (angl. Data Reader), kuriam yra priskiriama duomenų bazės valdymo sistemos komandines eilutes komanda, kad butu įvykdytas nuskaitymas iš duomenų bazės lentelės, pagal atitinkama SQL užklausą. Kada šį komanda bus įvykdyta, tada duomenų nuskaitymo įrenginyje bus atvaizduota duomenų bazės lentelės struktūra ir šios lentelės duomenys, t.y.

kitaip tariant bus iš duomenų bazės valdymo sistemos paimta sugeneruota lentelė pagal pateikta SQL užklausą ir ši sugeneruota lentelė bus atvaizduota duomenų nuskaitymo objekte. Toliau visas programos veikimo darbas bus vykdomas naudojantis šiuo duomenų nuskaitymo objektu. Tolimesnis šios procedūros veikimas užtikrina, kad bus sukurtas tarpinis XML dokumentas, kuriam bus priskirtas jau modifikuotas XML dokumentas, kuris tik neturės duomenų gautų iš duomenų bazės lentelės, t.y. dar nebus pilnai su duomenimis sugeneruotas XML tarpinis dokumentas. Sekančiu žingsniu procedūra patikrins ar yra tenkinama sąlyga, ar duomenų nuskaitymo objektas turi įrašų, jei ši sąlyga bus įvykdyta, tuomet bus inicializuotas ciklas kuris turi tenkinti šią sąlyga: „Kol yra įrašų duomenų nuskaitymo objekte“, kuomet ši sąlyga bus tenkinama, tada bus sukurtas XML dokumento elementas, kuriam bus priskirta reikšmė „row“, ši reikšme, kaip ir pats XML dokumento elementas yra reikalingas, tam kad atskirti eilučių įrašus formuojamame XML dokumente, kitaip tariant, kad matytųsi, kur prasideda nauja eilutė. Sekančiame šios procedūros veikimo stadijoje bus inicializuotas dar vienas ciklas, kuris turi tenkinti sąlygą, tol kol yra eilučių duomenų nuskaitymo objekte, kol ši sąlyga bus tenkinama, bus sukuriamas XML dokumento elementas, kuriam bus priskirta žymė ir šios žymės reikšmė bus lygi lentelės stulpelio pavadinimui, tarp šios žymės pradžios ir pabaigos bus įrašyta reikšmė paimta iš duomenų bazės lentelės, o tiksliau tariant iš duomenų nuskaitymo objekto. Tol kol bus tenkinama ciklo sąlyga tol kol yra eilučių duomenų nuskaitymo įrenginyje, kada šio ciklo sąlyga nebus tenkinama, tada bus įrašoma žymė, kuri buvo nuskaityta iš vartotojo užklausiamo XML dokumento, ir toliau bus vėl grįžtama prie ciklo sąlygos, ar yra įrašu duomenų nuskaitymo objekte, ir kai šio ciklo sąlyga nebetinkins, bus vėl patikrinta ar yra įrašų duomenų nuskaitymo objekte, ir galiausiai kai nebetenkins ir ši sąlyga, tada bus iš formuojamo XML dokumento išmetamas SQL užklausos tekstas ir vietoj jo įrašomas atitinkamas šiai užklausai atsakymas gautas iš duomenų bazės valdymo sistemos. Po to bus uždarytas duomenų nuskaitymo objektas. Ir ši procedūra savo darbą baigia.

4. Eksperimentinė dalis

4.1. Komunikavimo sąsajos tarp duomenų bazės patikrinimas

Tam kad atlikti greitaveikos skaičiavimus, t.y. sužinoti per kiek laiko yra išrenkami, įrašomi, atnaujinami, ištrinami įrašai duomenų bazės valdymo sistemose, šis rodiklis yra svarbiausias kuriant tokio tipo programas ir programinę įrangą. Tačiau, kad atlikti sukurtos programos skaičiavimą, turime sužinoti kaip greitai sąveikauja pati komunikavimo sąsaja tarp programos ir duomenų bazės valdymo sistemos. Kaip anksčiau buvo minėta, dauguma duomenų bazių valdymo sistemų gamintojų patys sukuria tam tikrom aplinkom komunikavimo sąsajas. Taip kaip ši programa buvo kuriama .NET aplinkoje, testavimui buvo pasirinktos komunikavimo sąsajos skirtos šiai aplinkai.

MySQL duomenų bazės valdymo sistemai, gamintojai yra sukūrę .NET Connector komunikavimo sąsaja tarp programų vykdomų .NET aplinkoje ir jų duomenų bazės valdymo sistemos. Taip pat ir kiti duomenų bazės valdymo sistemų gamintojai yra sukūrę šias komunikavimo sąsajas, Microsoft SQL Server duomenų bazės valdymo sistemai yra sukurtas – Microsoft SQL .NET Connector, Oracle duomenų bazės valdymo sistemai yra sukurtas – Oracle Data Access .NET Component.

Įdiegus į sistemą reikiamas duomenų bazės valdymo sistemos komunikavimo sąsajas, ir pasirašius elementaria programą, kuri atliktu kelis veiksmus su duomenų bazėmis, t.y. atliktu duomenų įrašymą į duomenų bazės valdymo sistemą, duomenų išrinkimą, duomenų atnaujinimą, ir duomenų šalinimą iš duomenų bazės valdymo sistemos.

Tam kiekvienoje duomenų bazės valdymo sistemoje buvo sukurta viena lentelė kuri turėjo keturis laukus, t.y. ID – identifikacijos laukas, tekstinis laukas, kur pastoviai buvo įrašinėjamas tas pats tekstas, laikas – kur buvo imamas sistemos laikas ir talpinamas į šios lentelės lauką, ir paskutinis laukas buvo skaičius – kur atsitiktine tvarka buvo generuojamas skaičius ir įrašomas į duomenų bazės valdymo sistemą. Tam, kad atlikti duomenų bazės valdymo sistemų greitaveikos skaičiavimą, buvo inicializuotas ryšis su duomenų bazės valdymo sistema, sukurtas ciklas, kuriam paduodama įrašų skaičiaus reikšmę, ir paduodama standartine SQL komanda. Tačiau dar reikėtu paminėti, kad atliekant šią eksperimentinę dalį, buvo pastebėtas SQL užklausos komandos sintaksės skirtumas tarp skirtingų duomenų bazės valdymo sistemų. Įrašų įterpimui buvo paduodama ši komanda:

Oracle
```
"INSERT INTO table (" & " id, textas, laikas, skaicius" & ") VALUES (" & "
" & d & ", 'Tekstas', '" & Now().ToLongTimeString & "', '" & Rnd().ToString
& "'" & ")"
```
Microsoft SQL
```
"INSERT INTO table (id, textas, laikas, skaicius) VALUES (" & d2 & ",
'Tekstas', '" & Now().ToLongTimeString & "', '" & Rnd().ToString & "');"
```
MySQL
```
"INSERT INTO table (id, textas, laikas, skaicius) VALUES (" & d3 & ",
'Tekstas', '" & Now().ToLongTimeString & "', '" & Rnd().ToString & "');"
```

Šioje komandoje ID numeris skaičius buvo paimamas iš ciklo reikšmės ir talpinamas į vieną iš laukų, sekančiame lauke buvo talpinama tekstine reikšmė – „Tekstas“, kitame lauke buvo talpinama sistemos laiko reikšmė, ir paskutinį lauką buvo talpinamas atsitiktinę tvarka pasirinktas skaičius. Tačiau, išnagrinėjus SQL užklausas geriau, galima pastebėti, kas Microsoft SQL ir MySQL duomenų bazės valdymo sistemų SQL užklausos sintaksė yra identiškos.

Sekančiu žingsniu buvo imama kita komanda, pavyzdžiui, duomenim iš duomenų bazės valdymo sistemos lentelės išrinkimas, tam buvo naudojama ši komanda:

Oracle
```
"SELECT * FROM table WHERE id='" & Math.Round(Rnd() * d) & "'"
```
Microsoft SQL
```
"SELECT * FROM table WHERE id='" & Math.Round(Rnd() * d2) & "'"
```
MySQL
```
"SELECT * FROM table WHERE id='" & Math.Round(Rnd() * d3) & "'"
```

Šios užklausos metu buvo kviečiama duomenų išrinkimo SQL užklausa, kuri išrinkinėjo įrašus iš duomenų bazės valdymo sistemos lenteles, pagal atsitiktine tvarka sugeneruoto ir apvalinto iki sveikųjų dalių skaičiaus. Tačiau, kaip matosi iš aukščiau pateiktų užklausų, išrinkimo užklausų sintaksės visų trijų duomenų bazės valdymo sistemų gamintojų yra vienodos.

Po to buvo imama kita komanda, pavyzdžiui, duomenų bazės valdymo sistemos lentelės duomenims atnaujinti, tam buvo naudojama ši komanda:

Oracle
```
"UPDATE table SET textas = 'new_tekstas', laikas = '" &
Now().ToLongTimeString & "', skaicius = '1111' WHERE (id = " & " " & d & "
" & ")"
```
Microsoft SQL
```
"UPDATE table SET textas = 'new_tekstas', laikas = '" &
Now().ToLongTimeString & "', skaicius = '1111' WHERE (id = " & d2 & ")"
```
MySQL
```
"UPDATE table SET textas = 'new_tekstas', laikas = '" &
Now().ToLongTimeString & "', skaicius = '1111' WHERE (id = " & d3 & ")"
```

Šios užklausos metu buvo kviečiama įrašų atnaujinimo SQL užklausa (komanda), t.y. iš duomenų bazės valdymo sistemos buvo išrenkamas įrašas pagal ID reikšmę, kur ši reikšmė

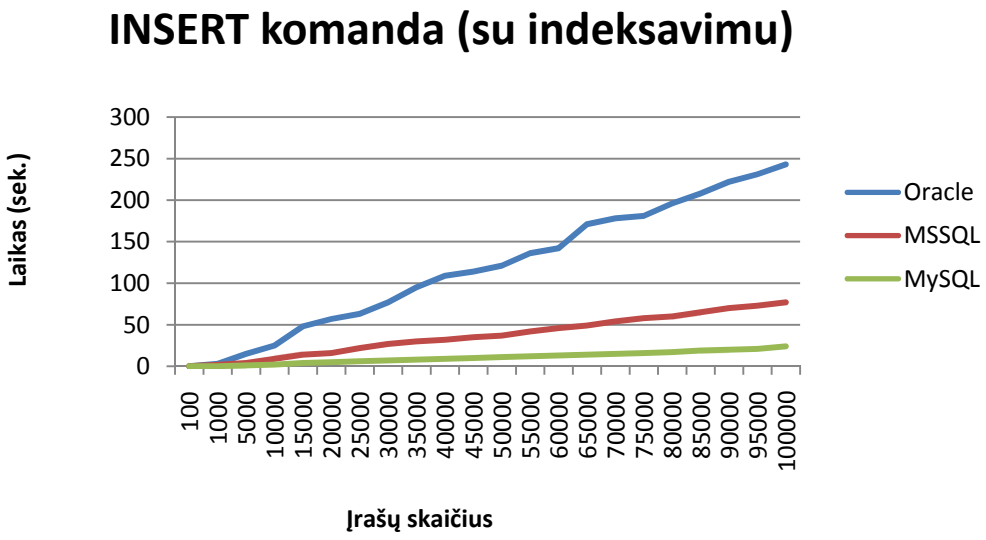
buvo generuojama tiksliau tariant imama iš vykdomojo ciklo, o po to vietoj tekstines reikšmės „tekstas“ buvo įrašoma reikšmė „new_tekstas“, vietoj sistemos buvusio laiko buvo įrašomas dabartinis sistemos laikas, o vietoj skaičiaus buvo tiesiog įrašomas skaičius „1111“. Kaip matosi iš aukščiau pateiktų užklausų, kad Microsoft SQL ir MySQL duomenų bazės valdymo sistemų SQL užklausos sintaksė yra vienoda.

Po to buvo paimta likusi komanda, duomenims iš duomenų bazės valdymo sistemos lentelės išmetimui, tam buvo naudojama ši komanda:

Oracle
```
"DELETE FROM BAND WHERE (id = " & " " & d & " " & ")"
```
Microsoft SQL
```
"DELETE FROM BAND WHERE (id = " & d2 & ")"
```
MySQL
```
"DELETE FROM BAND WHERE (id = " & d3 & ")"
```

Šios užklausos metu buvo kviečiama įrašų iš duomenų bazės valdymo sistemos išmetimo SQL užklausa (komanda), t.y. iš ciklo buvo imama ciklo reikšmė, ir pagal ID numerį iš duomenų bazės buvo išmetami duomenis. Kaip matosi iš aukščiau pateiktu Microsoft SQL ir MySQL užklausų SQL sintaksė yra vienoda.

Šios greitaveikos patikrinimas buvo vykdomas naudojant indeksuojama ir neindeksuojama lenteles, t.y. kada įrašai duomenų bazės valdymo sistemos lentelėje yra indeksuojami naudojant tam tikrus greitaveika užtikrinančius algoritmus ir be jų. Šiems įrašams sugeneruoti buvo naudojamas žingsnis, kas penkis tūkstančius įrašų, t.y. per kiek laiko buvo įrašytas 1 įrašas, 100 – įrašų, 1000 – įrašų, 5000 – įrašų, 10 000 – įrašų ir taip iki 100 000 įrašų. Buvo gauti sekantis duomenys, kurios pateiksime diagramomis.



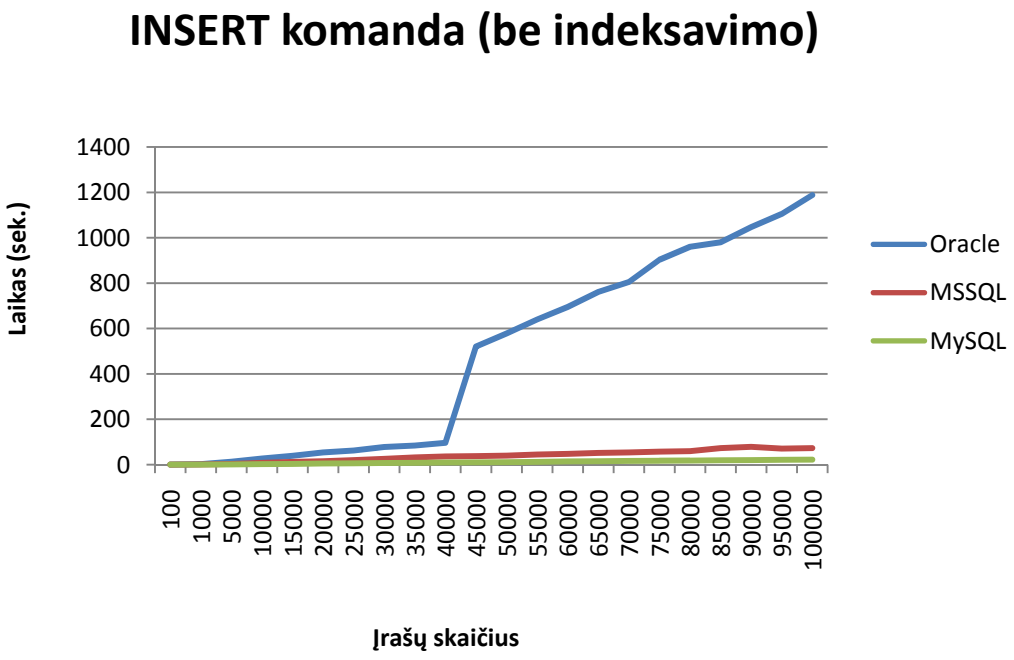
11 pav. SQL „insert“ komandos greitaveikos palyginimas indeksuojamoje lentelėje

Šiam eksperimentui, buvo naudojama SQL užklausos komanda, įrašams įterpti į duomenų bazės valdymo sistemą, kurioje duomenų bazės lentelė yra indeksuojama. ID rodiklis buvo užrašomas eiliškumo tvarka, t.y. buvo vykdomas ciklas, ir ciklo iteracijos skaičius buvo įrašomas į duomenų bazės valdymo sistemos lentelę.

Kaip matosi iš šio eksperimento, atliekant SQL „insert“ komandą, indeksuojamoje duomenų bazės valdymo sistemos lentelėje, geriausiai pasižymėjo MySQL duomenų bazės valdymo sistema, ne labai daug ko skiriasi ir Microsoft SQL duomenų bazės valdymo sistemos greitaveiką, tačiau Oracle duomenų bazės valdymo sistemai atliekant tą pačią komandą su tais pačiais duomenimis užtrūko daugiausiai laiko.

Nors, kaip teigia Oracle gamintojai, ir kiek buvo pravesta šios duomenų bazės valdymo sistemos greitaveikos bandymų, šie rodikliai yra kur kas geresni. Nes daugumoje šaltinių galima surasti tokius teiginius, kad Oracle duomenų bazės valdymo sistema yra pati geriausia, ir geriausiai apdoroja didelius duomenų kiekius. Tačiau čia lieka pasakyti šio rezultato vieną išvadą, kad gal šios blogus rezultatus įtakoja tai, kad yra naudojama laisvai platinama Oracle duomenų bazės valdymo sistemos versija, kuri specialiai yra padaryta taip, kad būtų blogesni rezultatai ir kad šios duomenų bazės valdymo sistemos vartotojai įsigytu mokama turinčia daugybę visokių privalumų Oracle duomenų bazės valdymo sistemą. Nes, pavyzdžiui, Microsoft SQL Express Edition versijoje, patys gamintojai teigia, kad ši versija skirta tik tam, kad pagaminti ir ištestuoti programą, ir kuri palaiko tiktai iki penkių pasijungimų.

Žemiau dar pateiksime greitaveikos diagramą, kaip veikė SQL „insert“ komanda su neindeksuojama duomenų bazės valdymo sistemos lentelė.



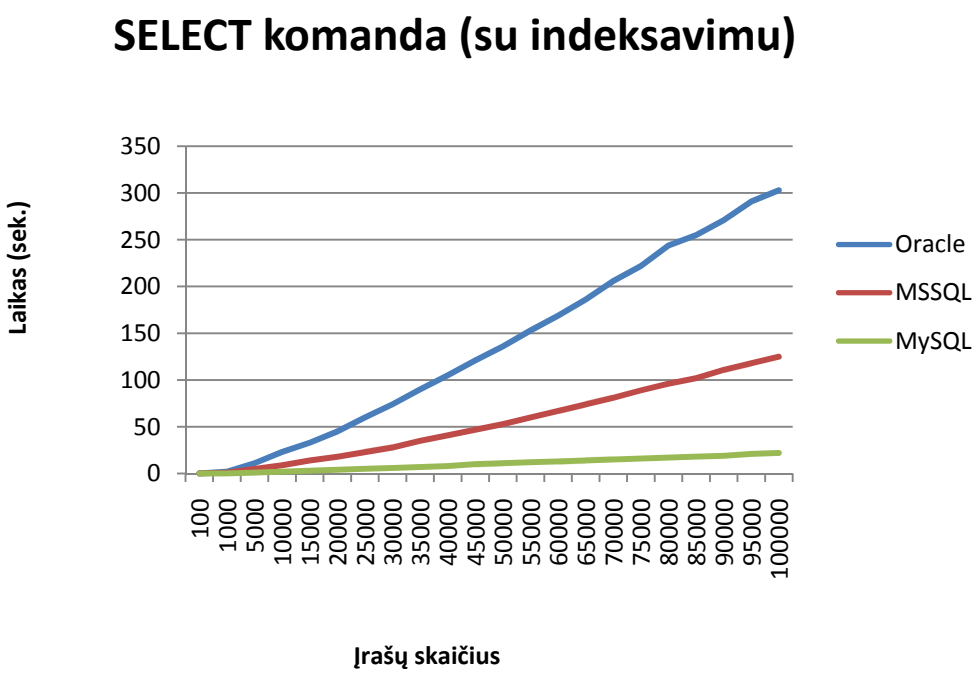
12 pav. SQL „insert“ komandos greitaveikos palyginimas ne indeksuojamoje lentelėje

Šiam eksperimentui, buvo naudojama SQL užklausos komanda, įrašams įterpti į duomenų bazės valdymo sistemą, kurioje duomenų bazės lentelė yra neindeksuojama. ID rodiklis buvo užrašomas eiliškumo tvarka, t.y. buvo vykdomas ciklas, ir ciklo iteracijos skaičius buvo įrašomas į duomenų bazės valdymo sistemos lentelę. Skirtumas nuo prieš tai buvusio eksperimento tik tas, kad yra naudojama neindeksuojama duomenų bazės valdymo sistemos lentelė. Šio eksperimentu norėta parodyti, koks yra skirtumas tarp indeksuojamos ir neindeksuojamos duomenų bazės valdymo sistemos lentelės.

Iš pateiktos diagramos matomas toks pat vaizdas, kad Oracle duomenų bazės valdymo sistema ilgiausiai trūko įterpti duomenys į duomenų bazės valdymo sistemos lentelę. Tačiau, kaip matoma šioje diagramoje, kad šis didžiulis laiko skirtumas atsirado tik po to kai buvo viršijama 40000 įrašų, nes kiek matosi iš aukščiau pateiktos diagramos iki 40000 įrašų, Oracle duomenų bazės valdymo sistema duomenys įterpia ganėtinai greitai. Todėl ir čia galima teigti tokias išvadas, kad tai yra specialus Oracle gamintojų nemokamos versijos apribojimas, kuris apsaugo jų gaminamas mokamas duomenų bazės valdymo sistemas, kad vartotojai pirktu pilna, t.y. mokama versiją šio gamintojo gaminamos duomenų bazės valdymo sistemos. Taip pat iš šios diagramos matyti, kad naudojant neindeksuojama duomenų bazės valdymo sistemos lentelė, duomenys į šią lentelę yra įrašomi greičiau, tai yra todėl, kad neindeksuojamose duomenų bazės valdymo sistemos lentelėse nereikia papildomai įrašinėti indeksus, kurie susiejami su atitinkamomis reikšmėmis pačioje duomenų bazės lentelėje. Bet

kiek matosi praktikoje, neindeksuojamų lentelių nieks beveik nenaudoja, nes tai įtakoja duomenų paieškos greitaveikos rezultatus, pačioje duomenų bazės lentelėje.

Apie paieškos greitaveikos rezultatus aptarsime žemiau. Pateiksime grafikus ir atliksime palyginimus.



13 pav. SQL „select“ komandos greitaveikos palyginimas indeksuojamoje lentelėje

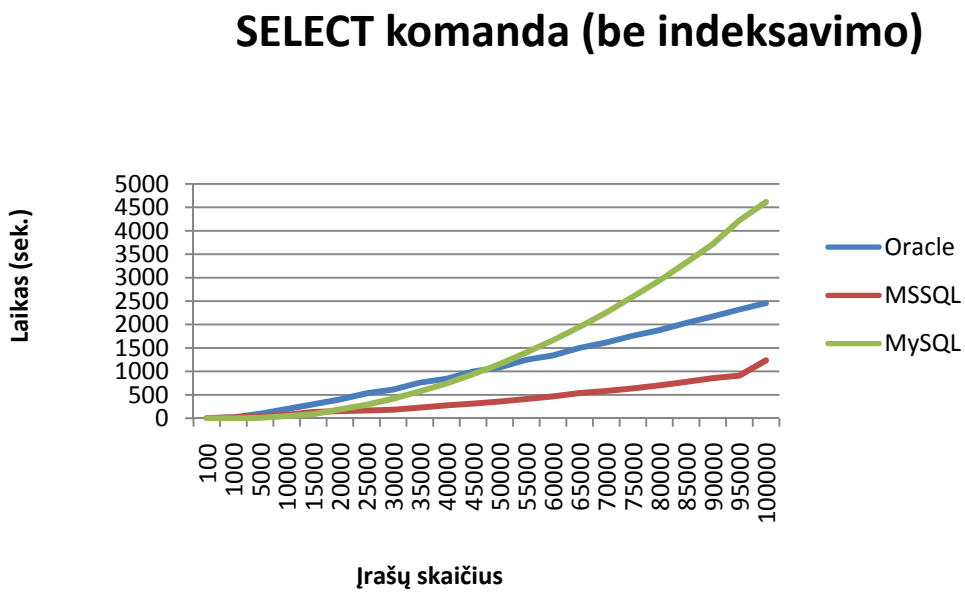
Šiam eksperimentui, buvo naudojama SQL užklausos komanda, įrašams ieškoti duomenų bazės valdymo sistemos lentelėje, kurioje duomenų bazės lentelė yra indeksuojama. Paieška buvo vykdoma nurodant ID rodiklio reikšmę. Vykdant šią užklausą ID rodyklės parametras buvo generuojamas atsitiktinė skaičių seka ir apvalinamas iki sveikųjų skaičiaus dalies. Šiam veikimui buvo vykdomas ciklas, ir kiek ciklo iteracijų buvo, tiek buvo atlikta paieškos užklausų į duomenų bazės valdymo sistemą.

Kaip matosi iš aukščiau pateiktos diagramos, vėl gi Oracle duomenų bazės valdymo sistema duomenys apdoroja lėčiausiai, t.y. išrenka duomenys iš indeksuojamos duomenų bazės lentelės lėčiausiai, ir tai yra todėl, kad Oracle gamintojai savo nemokamai platinamam produktui yra uždėję apribojimus, kad ši nemokamai platinama duomenų bazės valdymo sistema, negalėtu dirbti geriau, negu šio gamintojo platinama mokama duomenų bazės valdymo sistema, tai yra padaryti iš verslo tikslų. Taip pat iš šios diagramos matoma, kad Microsoft SQL duomenų bazės valdymo sistema veikia taipogi lėčiau negu MySQL duomenų bazės valdymo sistema, ir tai yra todėl, kad Microsoft gamintojai, taip pat kaip ir Oracle gamintojai apsaugo savo mokama produkcija, nes logiška manyti, jeigu šie gamintojai

platintu gerai ir greitai veikiančias nemokamas duomenų bazės valdymo sistemas, niekas iš jų nepirktu mokamu.

Tačiau kaip matosi iš diagramos, MySQL duomenų bazės valdymo sistema, su šią užduotimi susitvarkė geriausiai, ir tai yra todėl, kad MySQL duomenų bazės valdymo sistema yra laisvai platinama ir jos išeities kodas taipogi yra laisvai platinamas, todėl kiekvienas programuotojas galėtu tos apribojimus, nuimti, dėl šios priežasties ir nėra šiai duomenų bazės valdymo sistemai uždėta jokiu apribojimų. Tačiau dar reikėtu paminėti, kad MySQL gamintojai šio metu yra išleidę viena MySQL Enterprise mokama duomenų bazės valdymo sistemos versiją, kur gamintojai teigia, kad į šią versiją yra įtrauktas geresnis duomenų apdorojimo variklis, kuris duomenys apdoroja greičiau, ir iškarto į šią sistema yra įdiegti administravimo ir kiti analizės įrankiai.

Žemiau palyginimui pateiksime diagramą, kurioje buvo atlikti tie patys veiksmai su duomenų bazės valdymo sistemomis, kaip ir aukščiau, tik bus vienas skirtumas, kad šis bandymas bus atliktas su neindeksuojama duomenų bazės valdymo sistemos lentelė.

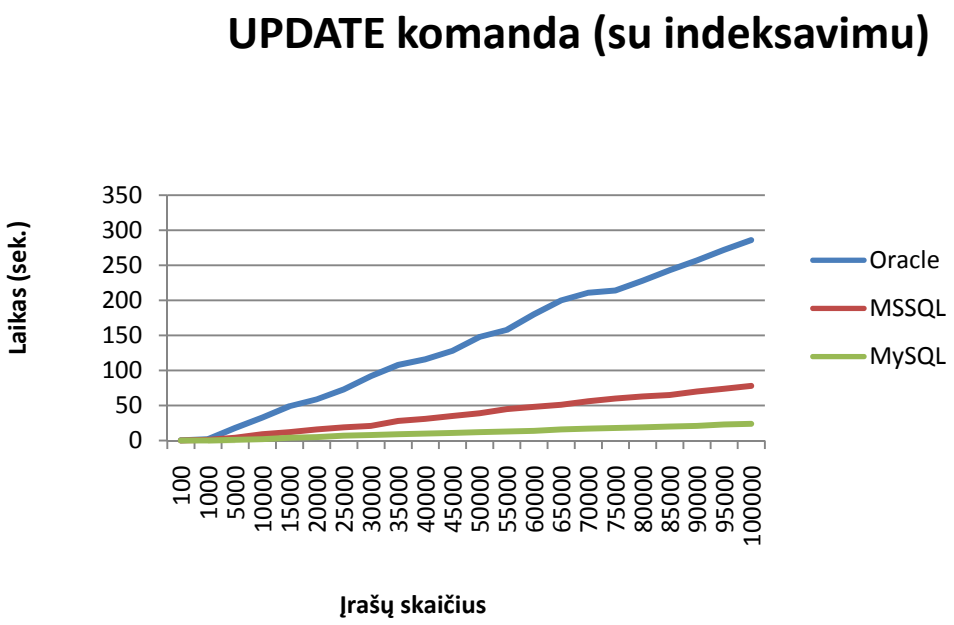


14 pav. SQL „select“ komandos greitaveikos palyginimas ne indeksuojamoje lentelėje

Šiam eksperimentui, buvo naudojama SQL užklausos komanda, įrašams ieškoti duomenų bazės valdymo sistemos lentelėje, kurioje duomenų bazės lentelė yra neindeksuojama. Paieška buvo vykdoma nurodant ID rodiklio reikšmę. Vykdant šią užklausą ID rodyklės parametras buvo generuojamas atsitiktinė skaičių seka ir apvalinamas iki sveikųjų skaičiaus dalies. Šiam veikimui buvo vykdomas ciklas, ir kiek ciklo iteracijų buvo, tiek buvo atlikta paieškos užklausų į duomenų bazės valdymo sistemą.

Kaip matosi iš pateiktos diagramos, šiame bandyme blogiausią greitaveikos rezultatą parodė MySQL duomenų bazės valdymo sistema. Šis greitaveikos rezultatas yra blogiausias dėl to, kad kada buvo kviečiama „select“ SQL užklausa su jai paduodamu atsitiktinai sugeneruotu ID numeriu, duomenų bazės valdymo sistemai reikėjo tikrinti eilės tvarka visus įrašus esančius ID stulpelyje, o dėl to kad ID skaičius buvo generuojamas atsitiktinę tvarka, šis greitaveikos laikas pasidarė labai didelis, tai yra todėl, kad, pavyzdžiui, buvo paduota užklausa, kur reikėjo išrinkti visus duomenų bazės valdymo sistemos lentelės duomenys, kur ID lauko reikšmė yra 5, o sekančia užklausa kur ID lauko reikšmė yra 55, dar sekančia užklausa, kur ID lauko reikšmė yra 25. Tai vykdant duomenų išrinkimą iš duomenų bazės valdymo sistemos lentelės, eilės tvarka buvo ieškoma, kur ID lauko reikšmė yra lygi 5, sekančios užklausos metu nuo pradžių vėl buvo ieškoma, kur ID lauko reikšmė yra lygi 55 ir t.t. O šis išrinkimo būdas labai atspindi greitaveiką. Ši greitaveika pasidarė labai didelė dėl to, kad šiam duomenų išrinkimui iš neindeksuojamos duomenų bazės valdymo sistemos lentelės nėra naudojamas joks algoritmas. Tačiau, kaip buvo paminėta anksčiau, mažai kas iš duomenų bazės valdymo sistemų programuotojų ir administratorių naudoja neindeksuojamas duomenų bazės valdymo sistemos lenteles, nes tai labai atsispindi duomenų išrinkimui iš duomenų bazės valdymo sistemos greitaveikai. Bet detaliau panagrinėju diagramą, iškyla klausimas, kodėl gi Oracle ir Microsoft SQL duomenų bazės valdymo sistemose ši greitaveika yra kur kas geresnė. Atsakymas į šį klausimą butu toks, kad šie gamintojai naudoja tam tikrus algoritmus duomenims išrinkti iš duomenų bazės valdymo sistemos neindeksuojamos lentelės. Tačiau gamintojai neteigia šios informacijos kokius algoritmus jie naudoja.

Toliau dar bus išnagrinėta vieną iš SQL užklausos komandų „Update“, ši komanda skirta duomenim surasti duomenų bazės valdymo sistemos lentelėje ir jos atnaujinti pagal atitinkamus parametrus. Žemiau pateiksime šios SQL komandos greitaveikos palyginimą su skirtingu gamintojų duomenų bazės valdymo sistemomis.



15 pav. SQL „update“ komandos greitaveikos palyginimas indeksuojamoje lentelėje

Šiam eksperimentui, buvo naudojama SQL užklausos komanda, įrašams atnaujinti duomenų bazės valdymo sistemoje, kurioje duomenų bazės lentelė yra indeksuojama. Duomenų įrašų atnaujinimas buvo vykdomas nurodant ID rodiklį. Šio rodiklio reikšmė buvo paduodama eiliškumo tvarka, t.y. buvo vykdomas ciklas, ir ciklo iteracijos skaičius buvo paduodamas ID reikšmei, pagal šią reikšmę ir buvo atnaujinami duomenys duomenų bazės valdymo sistemos lentelę.

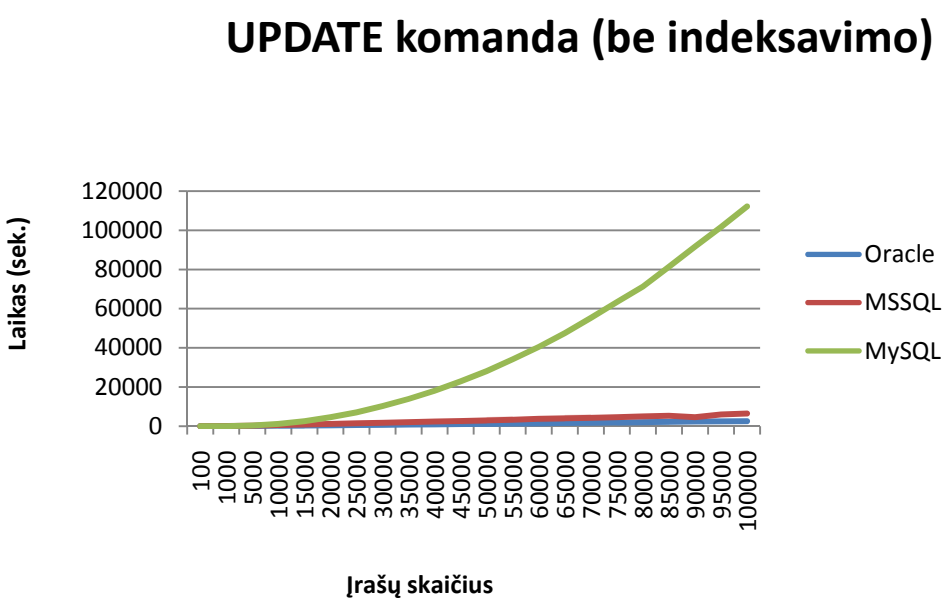
Galime teigti, kad atliekant „update“ SQL užklausos komandą, yra atliekama dvi komandos, t.y. viena komanda randa įrašą duomenų bazės valdymo sistemos lentelėje, o kita komanda šia užklausą atnaujina (perrašo). Kitaip tariant, galime įsivaizduoti, kad yra atliekama SQL užklausos komanda „select“ ir po to yra atliekama komanda – „delete/instert“.

Tačiau jei šią diagramą, palyginti su išrinkimo ir įterpimo diagramomis, kur šie veiksmai vyksta indeksuojamose duomenų bazės valdymo sistemos lentelėje, pirmą žvilgsni patrauks tai, kad Oracle duomenų bazės valdymo sistema vėl gi atlieka šią komandą lėčiausiai, nes tai įvyksta dėl to, kad yra apribotos šios nemokamai platinamos duomenų bazės valdymo sistemos galimybės. Ir taip, galime padaryti tokia išvadą, jeigu Oracle duomenų bazės valdymo sistemos duomenų išrinkimo ir duomenų įterpimo greitaveikos yra lėtos, todėl ir vykdant duomenų atnaujinimą ši greitaveika išlieka taipogi maža.

Taip pat šioje diagramoje galima pamatyti, kad Microsoft SQL ir MySQL duomenų bazės valdymo sistemų greitaveika labai panaši į šių duomenų bazės valdymo sistemų duomenų įterpimo greitaveiką.

Taip pat atidžiau panagrinėjus šias diagramas, buvo rastas vienas panašumas kol duomenų skaičius yra labai mažas, kažkur apie 1000 įrašų, šios tris bandomos duomenų bazės valdymo sistemos veikia labai panašiai, t.y. jų greitaveikos laikas beveik identiškas, iš šio teiginio galima padaryti tik tokia išvada, kad šios nemokamai platinamos duomenų bazės valdymo sistemos skirtos tik atlikti programų testavimo darbus, su dideliais duomenų kiekiais nei Microsoft SQL, nei Oracle duomenų bazės valdymo sistema normaliam greitam darbui neskirtos, tam reikia įsigyti šių gamintojų platinamas mokamas duomenų bazės valdymo sistemų versijas.

Toliau atliksime šių trijų duomenų bazės valdymo sistemų greitaveikos patikrinimą, kai yra vykdoma duomenų atnaujinimo komanda „update“ tik neindeksuojamoje lentelėje.



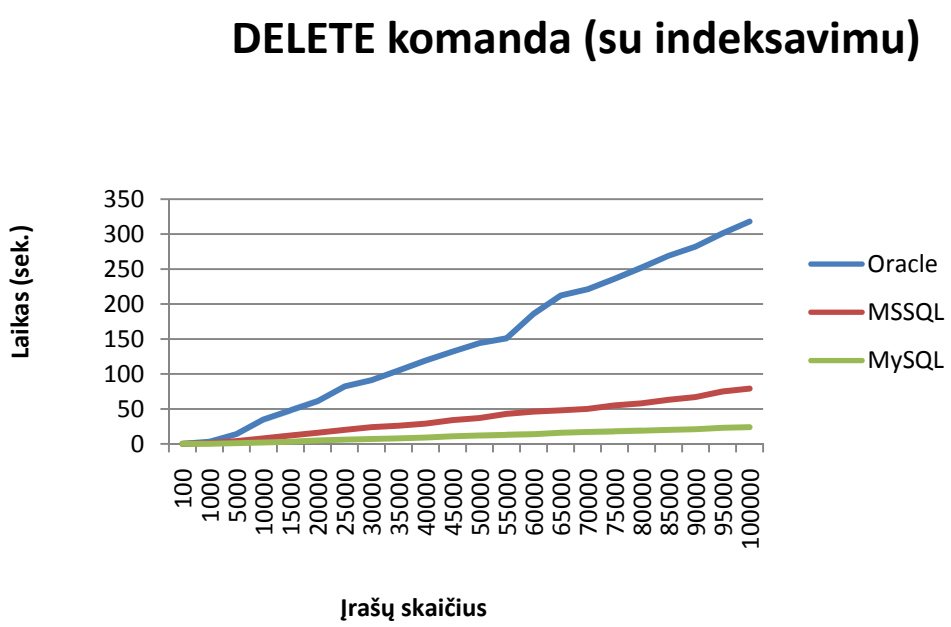
16 pav. SQL „update“ komandos greitaveikos palyginimas ne indeksuojamoje lentelėje

Šiam eksperimentui, buvo naudojama SQL užklausos komanda, įrašams atnaujinti duomenų bazės valdymo sistemoje, kurioje duomenų bazės lentelė yra neindeksuojama. Duomenų įrašų atnaujinimas buvo vykdomas nurodant ID rodiklį. Šio rodiklio reikšmė buvo paduodama eiliškumo tvarka, t.y. buvo vykdomas ciklas, ir ciklo iteracijos skaičius buvo paduodamas ID reikšmei, pagal šią reikšmę ir buvo atnaujinami duomenys duomenų bazės valdymo sistemos lentelę.

Kaip yra matoma aukščiau pateiktoje diagramoje, kada yra naudojama neindeksuojama duomenų bazės valdymo sistemos lentelė, greitaveika pasižymėjo Oracle ir Microsoft SQL duomenų bazės valdymo sistemos, tai yra todėl kad šie du gamintojai savo produktuose naudoja ypatingus algoritmus, kurie užtikrina tokia greitaveika neindeksuojamose duomenų bazės valdymo sistemos lentelėse. Tačiau, jei atkreipti dėmesį į

MySQL duomenų bazės valdymo sistemos greitaveika, kuri yra labai lėta, tai yra todėl, kad kada yra naudojamos neindeksuojamos duomenų bazės valdymo sistemos lentelės, daug laiko yra sugaištama tos duomenys surasti duomenų bazės lentelėje pagal atitinkama išrinkimo užklausą, t.y. kaip buvo minėta aukščiau, kada yra naudojamos neindeksuojamos duomenų bazės lentelės ir naudojamos išrinkimo ir duomenų įrašymo SQL užklausos (komandos), MySQL duomenų bazės valdymo sistema neturi savyje aprašyto greito duomenų apdorojimo algoritmo, ir todėl kad šios duomenys atnaujinti, reikiama įrašą reikia pirmiasia surasti, o kad tai atlikti reikia perbėgti per visus įrašus ir tik suradus reikiama įrašą, jį atnaujinti. Labai panašia maža greitaveiką yra pastiebiama ir išrinkimo ir įterpimo metu, kai šie veiksmai vykdomi su neindeksuojama duomenų bazės valdymo sistemos lentele ir naudojanti MySQL duomenų bazės valdymo sistema.

Toliau apžvelgsime likusiąją komandą, kuri ištrina įrašus iš duomenų bazės valdymo sistemos.



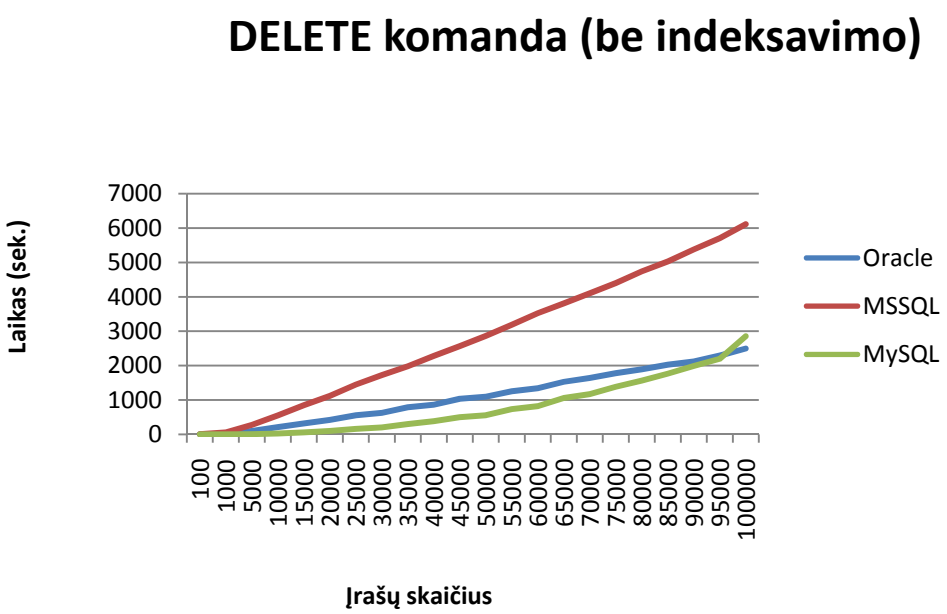
17 pav. SQL „delete“ komandos greitaveikos palyginimas indeksuojamoje lentelėje

Šio testavimo metu buvo atliekama SQL užklausos komanda – „delete“, kuri išmeta įrašus iš duomenų bazės valdymo sistemos lentelės, šis eksperimentas buvo atliekamas indeksuojamoje duomenų bazės lentelėje. Išmetimas buvo vykdomas pagal paduodamą ID reikšmę, t.y. šis veiksmas buvo vykdomas ciklišk ai, ir kiek buvo ciklo iteracijų, tiek buvo išmesta įrašų iš duomenų bazės valdymo sistemos lentelės.

Šioje diagramoje matyti, kad ir vėl Oracle duomenų bazės valdymo sistemos greitaveika yra labai lėta. Tai įvyko dėl to, kad Oracle gamintojai specialiai vykdo apribojimus savo nemokamai platinamai produkcijai.

Tačiau geriau panagrinėjus šią diagramą, pastiebiama, kad Microsoft SQL duomenų bazės valdymo sistemos greitaveika yra blogesnė nei MySQL duomenų bazės valdymo sistemos. Bet, kol įrašų skaičius yra ganėtinai mažas, visų duomenų bazės valdymo sistemų greitaveikos laikas beveik identiškas. Tačiau, kai įrašų skaičius didelis, MySQL duomenų bazės valdymo sistemos greitaveika yra geresnė.

Toliau dar išnagrinėsime greitaveiką, kaip šios duomenų bazės valdymo sistemos išmetinėja įrašus iš neindeksuojamos duomenų bazės valdymo sistemos lentelės.



18 pav. SQL „delete“ komandos greitaveikos palyginimas indeksuojamoje lentelėje

Šio testavimo metu buvo atliekama SQL užklausos komanda – „delete“, kuri išmeta įrašus iš duomenų bazės valdymo sistemos lentelės, šis eksperimentas buvo atliekamas indeksuojamoje duomenų bazės lentelėje. Išmetimas buvo vykdomas pagal paduodamą ID reikšmę, t.y. šis veiksmas buvo vykdomas ciklišKai, ir kiek buvo ciklo iteracijų, tiek buvo išmesta įrašų iš duomenų bazės valdymo sistemos lentelės.

Kaip matosi iš aukščiau pateiktos diagramos, Microsoft SQL duomenų bazės valdymo sistemos greitaveika yra pati blogiausia. Ir šios greitaveikos rezultatas yra blogas dėl to, kad yra naudojama neindeksuojama duomenų bazės valdymo sistemos lentelė, tačiau, taip pat šioje vietoje dar sukelia įtarimų ir tas, kad greičiausiai Microsoft SQL duomenų bazės valdymo sistemoje, kai yra išmetami duomenys iš neindeksuojamos duomenų bazės lentelės yra naudojamas blogas įrašų išmetimo algoritmas iš neindeksuojamos duomenų bazės

lentelės. Arba tai yra padaroma specialiai, kada yra naudojama laisvai platinama Microsoft duomenų bazės valdymo sistema.

Tačiau panagrinėjus diagramą, pastiebiama, kad MySQL ir Oracle duomenų bazės valdymo sistemos greitaveikos yra labai panašios. Bet vis dėl to, kol įrašų skaičius lentelėje yra nelabai didelis MySQL duomenų bazės valdymo sistemos greitaveika yra geresne nei Oracle. Bet čia vėl gi iškyla problema, gal šie Oracle duomenų bazės valdymo sistemos greitaveikos eksperimento rezultatai yra blogesni dėl to, kad yra naudojama laisvai platinama Oracle duomenų bazės valdymo sistema.

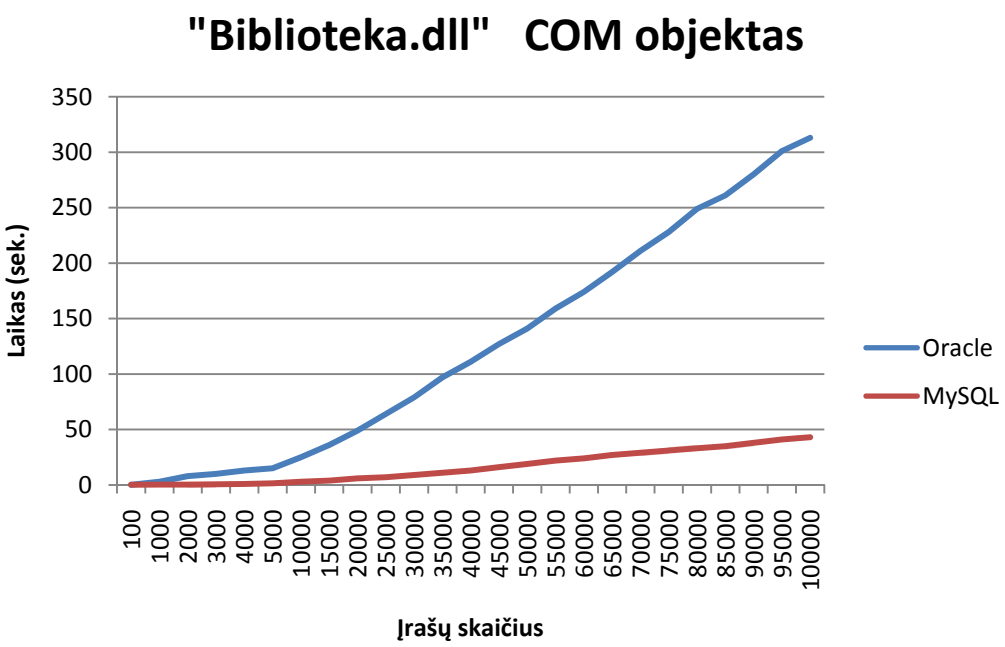
Dabar pateiksime, apibendrintai išvadas apie visas eksperimento detales.
Kiek iš šių greitaveikos eksperimentų pastebėta, geriausiai pasirodė laisvai platinama, atviro kodo MySQL duomenų bazės valdymo sistema. Tačiau ši greitaveika yra tik to atveju, jeigu yra naudojama neindeksuojama duomenų bazės valdymo sistemos lentelė. Todėl šią duomenų bazės valdymo sistema naudojasi vis daugiau vartotojų, nes ši sistema yra platinama nemokamai, greitaveikos laikas geras, palyginus su kitomis laisvai platinamomis duomenų bazės valdymo sistemomis. Ir tai yra pats didžiausias šios duomenų bazės valdymo sistemos privalumas.

4.2. Sukurtos programinės įrangos greitaveikos testavimas

Dabar reikėtu atlikti eksperimentą su sukurta programinę įranga, kad sužinoti ant kiek pablogėjo sumažėjo greitaveika MySQL ir Oracle duomenų bazės valdymo sistemų. Šiame eksperimente bus naudojama indeksuojama duomenų bazės valdymo sistemos lentelė, nes kiek matosi iš ankstesnių greitaveikos patikrinimų iš tokios lentelės duomenų išrinkimo laikas greičiausias, neskaitant apribojimų, kurie yra uždėti ant laisvai platinamų duomenų bazės valdymo sistemų.

Kaip anksčiau buvo aprašyta ši sukurta programinė įranga, t.y. ISAPI praplėtimas, naudojasi .NET technologijos pagrindu sukurtu COM objektu kuriame vykdoma visa programos logika, t.y. pasijungimas prie duomenų bazės valdymo sistemos, duomenų išrinkimas, XML dokumento formavimas ir šio suformuoto dokumento gražinimas, todėl reikėtu patikrinti pačio COM objekto greitaveiką.

Kad tai atlikti, buvo parašyta maža programėlė, kuri inicializuoja COM objekto veikimą, tam COM objektui paduoda XML failą, kuriame yra aprašyta SQL užklausos komanda. Ir galiausiai, kada COM objektas atlieka visus veiksmus pagal jo aprašyta algoritmą, gražina suformuota XML dokumentą sukurtai testiniai programėlei. Žemiau pateiksime diagramą, kur buvo atliktas COM objekto greitaveikos patikrinimas su MySQL ir Oracle duomenų bazės valdymo sistemomis.



19 pav. COM objekto greitaveikos patikrinimas su indeksuojama lentele.

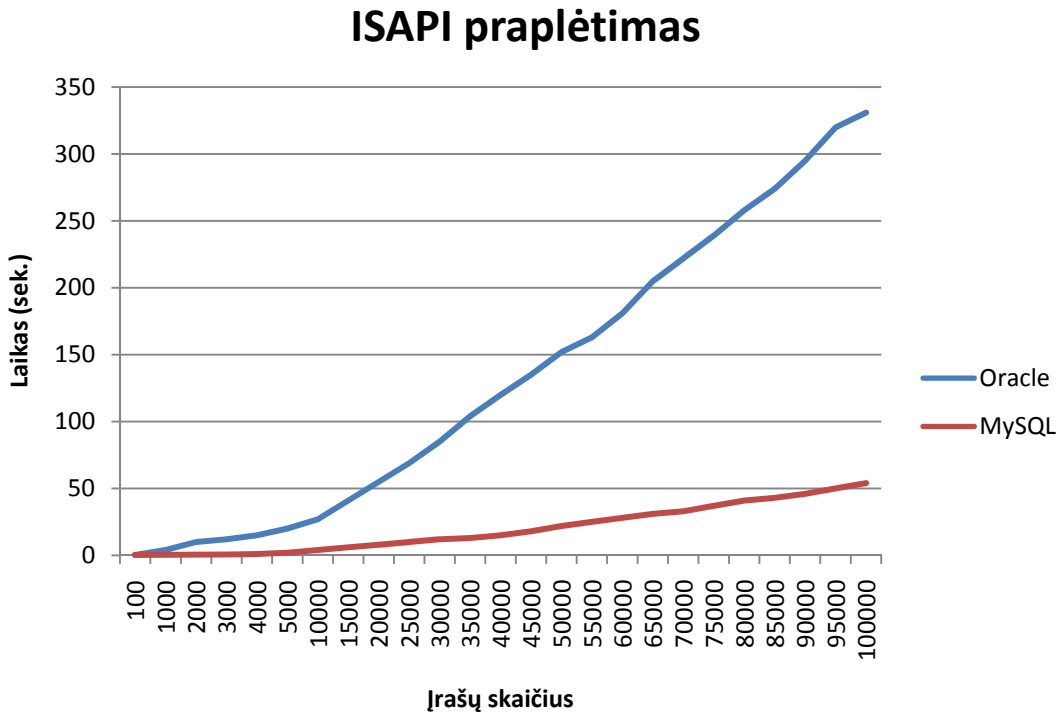
Šiam eksperimentui atlikti buvo naudojama indeksuojama duomenų bazės valdymo sistemos lentelė, į kuria buvo įrašomi duomenų įrašai. Taip pat buvo sukurtas XML dokumentas (failas) kuriame buvo aprašyta SQL užklausą duomenim išrinkti, t.y. SELECT SQL komanda, kuri išrenka visus įrašus iš duomenų bazės valdymo sistemos lentelės. Taip pat buvo sukurta maža konsolinė programa, kuri inicializuoja COM objekto veikimą ir perduoda jam anksčiau paminėto XML dokumento kelią. Kai XML dokumento kelias yra perduotas COM objektui, COM objektas pradeda darbą, t.y. pasijungę prie duomenų bazės valdymo sistemos, šiai sistemai siunčia SQL užklausą duomenim iš lentelės išrinkti, gautus duomenis suformuoja į XML formą ir jau suformuota XML dokumentą gražina vartotojui, t.y. mūsų atveju šiai konsolinei programai. Taip pat pabaigoje yra išvedamas laikas, kiek užtrūko COM objekto veikimas, t.y. gaunamas COM objekto greitaveikos laikas.

Kaip matosi iš pateiktos aukščiau diagramos, čia geriausia greitaveika pasižymėjo MySQL duomenų bazės valdymo sistema. Tai įvyko todėl, kad anksčiau atliktame eksperimente, kur buvo testuojama SELECT komandos greitaveika, indeksuojamoje duomenų bazės valdymo sistemos lentelėje, kur buvo prieita išvados, kad Oracle duomenų bazės valdymo sistemos greitaveika yra maža (lėta), todėl kad Oracle duomenų bazės valdymo sistemos gamintojai specialiai uždėjo apribojimą laisvai platinamai Oracle duomenų bazės valdymo sistemai. Todėl ir šiame eksperimente šis Oracle duomenų bazės valdymo sistemos greitaveikos laikas yra blogas.

Tačiau atidžiau įsižiūrėjus į diagramos grafiką, pastebima tai, kad greitaveikos laikas tarp duomenų išrinkimo, kai buvo atliktas eksperimentas testuojant duomenų bazės valdymo

sistemos pajungimo sąsają ir šio eksperimento testuojant COM objekto greitaveiką pastebėta tai, kad greitaveikos laikas šiek tiek padidėjo, tai įvyko todėl, kad COM objektas ne tik atlieka SQL užklausos duomenų išrinkimo iš duomenų bazės valdymo sistemos lentelės, bet dar ir formuoja pati XML dokumentą, todėl COM objekto greitaveika yra blogesnė nei duomenų bazės valdymo sistemos sąsajos greitaveika. To ir reikėjo tikėtis, nes COM objektas atlieka daugiau veiksmų, kuriems atlikti reikia daugiau laiko.

Žemiau pateiksime dar vieno eksperimento rezultatus. Šis eksperimentas buvo atliktas, tam, kad pažiūrėti kiek pasikeis greitaveika, kada testavimas bus atliktas per ISAPI praplėtimą.



20 pav. ISAPI praplėtimo greitaveikos patikrinimas su indeksuojama lentele

Šiam eksperimentui atlikti buvo naudojamas anksčiau testotasis COM objektas ir ISAPI praplėtimas. Tačiau šio eksperimento metu XML dokumentas buvo paduodamas naudojantis žiniatinklio/internetinio serverio pagalba, t.y. buvo per internetinę naršyklę iškviečiamas ISAPI praplėtimas, kuris turėjo apdoroti paduodama XML dokumentą ir gražinti į naršyklės langą suformuota XML dokumentą. Šio testavimo metu buvo skaičiuojamas greitaveikos laikas taip, buvo įrašoma užklausą, t.y. kreipinys į XML dokumentą, kuris randasi internetiniame serveryje, į naršyklės langą, ir tik įvykdžius šią užklausa pradedamas skaičiuoti laikas. Per šį laiką sistema turėjo užkrauti ISAPI praplėtimą į savo darbo procesą, po to ISAPI praplėtimas turėjo inicializuoti COM objekto veikimą, ir galiausiai gražinti suformuota XML dokumento tekstą, vartotojui į naršyklės langą. Tačiau šis testavimas buvo

atliekamas du kartus, nes pirmo užkrovimo-paleidimo laikas visados buvo didesnis, nes per ši laiką turėjo būti užkraunamas ISAPI praplėtimas į sistemos procesą, o tai šiek tiek užtrūkdavo, todėl antro kreipimosi metu jau ISAPI praplėtimas buvo užkrautas į sistemos procesą ir greitaveikos laikas šiek tiek buvo geresnis, todėl kreipimasis buvo atliekamas du kartu, ir iš gautu dviejų laikų buvo imamas vidurkis.

Kaip matosi iš diagramos, vėl gi geriausiai pasirodė MySQL duomenų bazės valdymo sistema, bet tai yra todėl kad greičiausiai kaip buvo aptarta anksčiau, kad Oracle duomenų bazės valdymo sistemos greitaveikos laikas yra blogesnis dėl to, kad yra naudojama nemokamai platinama Oracle duomenų bazės valdymo sistemos versija, kuriai yra uždėti tam tikri gamintojo apribojimai.

Tačiau geriau įsižiūrėjus į diagramas mes pastiebėme, kad greitaveikos laikas yra šiek tiek blogesnis, nes COM objekto eksperimento metu parodytas greitaveikos laikas. Šį reiškinį galima lengvai paaiškinti ir pakomentuoti. Tai įvyksta todėl, kad yra skaičiuojamas pats ISAPI praplėtimo užkrovimo laikas ir dar plius yra skaičiuojamas tas laikas, kuris reikalingas atvaizduoti suformuotą XML dokumento turinį naršyklės lange. Taip pat galime pastebėti, kad ko didesnis duomenų kiekis to daugiau laiko yra užtrunkama, tai įvyksta todėl, kad duomenų bazės valdymo sistemos lentelėje yra daugiau įrašų, daugiau užtrunkama laiko jos išrinkti, taip pat padidėja ir pačio XML dokumento formavimo laikas, nes yra surašoma daugiau žymių ir atvaizduojama daugiau duomenų, ir dar vienas įtakojantis faktorius yra tas, kad ko didesnis yra suformuotas XML dokumentas, to daugiau laiko užtrunkama šį dokumentą atvaizduoti internetinės naršyklės lange.

5. Gauti rezultatai ir išvados

Darbo rezultatai:

- ✓ Suprojektuota vieninga programinė įranga, leidžianti duomenų mainus tarp MySQL ir Oracle duomenų bazių valdymo sistemų ir žiniatinklio/internetinio serverio, išlaikant tą pačią XML dokumento struktūrą, kaip ir OpenXML.
- ✓ Atliktos duomenų bazių valdymo sistemų sąsajos greitaveikos palyginimai, atliekant skirtingas SQL užklausas.
- ✓ Atliktas sukurtos programinės įrangos greitaveikos palyginimas, tarp laisvai platinamų duomenų bazių valdymo sistemų Oracle ir MySQL.

Išvados:

- Atviro kodo laisvai platinamos duomenų bazės valdymo sistemos, tokios kaip MySQL, PostgreSQL neturi galimybės duomenim išgauti per XML dokumentus aprašant juose SQL užklausas.
- Tarp laisvai platinamų duomenų bazių valdymo sistemų duomenų mainams per XML dokumentą palaiko tiktai Microsoft SQL Express duomenų bazės valdymo sistema.
- Didžiausią našumą turi atviro kodo MySQL duomenų bazės valdymo sistema tarp duomenų mainams per XML dokumentus.
- Labiausiai paplitusioms duomenų bazių valdymo sistemoms yra sukurtos komunikavimo sąsajos .NET aplinkai.
- Nebuvo sukurtos vieningos programinės įrangos, duomenų mainams tarp DBVS ir XML dokumento, kuriame yra aprašytos SQL užklausos.
- Suprojektuota programinė įranga gali būti taikoma įvairioms duomenų bazių valdymo sistemoms keičiant kreipimosi į DBVS programinį komponentą.

6. Literatūros sąrašas

1. Atviras kodas Lietuvai. Iš Atviras Kodas Lietuvai [interaktyvus]. [žiūrėta 2008-01-24]. Prieiga per internetą: http://www.akl.lt/ak/licencijos
2. Licencijų naudojimo statistikos duomenys. Iš Freshmeat [interaktyvus]. [žiūrėta 2008-01-30] Prieiga per internetą: http://freshmeat.net/stats/#license
3. Į servisą orientuota architektūra. Iš IBM [interaktyvus]. [žiūrėta 2008-02-18]. Prieiga per internetą: http://www-05.ibm.com/lt/ondemand/insights/soa.html
4. Paul C. Brown. Implementing SOA : Total Architecture in Practice. JAV, 2008. 736 p.
5. Timo Böhme, Erhard Rahm. Evaluation of XML Data Management Systems with XMach. Mokslinis straipsnis. Vokietija, 2008. Prieiga per internetą: http://www.springerlink.com/content/1nvgeya68k9uqw6w/fulltext.pdf
6. Сергеев Александр Петрович. HTML & XML. Профессиональная работа. Maskva, 2004. 880 p.
7. O'Reilly. XML Schema. JAV, 2005.396 p.
8. Devan Shepherd. Sams Teach Yourself XML in 21 Days. Norvegija, 2002. 598 p.
9. Grem Malkolm. Програмирование MsSQL Server 2000 с использованием XML. Maskva, 2002. 318 p.
10. XML standartas Microsoft produktuose. Iš Microsoft [interaktyvus]. [žiūrėta 2008-05-19]. Prieiga internete: http://office.microsoft.com/lt-lt/help/HA100340221063.aspx
11. Skyrius R., Mikalauskienė A., Zalieckaitė L. Informacijos ir komunikacijos technologijos. Vilnius, 2008. 360 p.
12. Sekliuckis V., Garšva G., Gudas S. Informacijos sistemos ir duomenų bazės. Kaunas, 2006. 349 p.
13. Baronas R. Duomenų bazių sistemos. Vilnius, 2005. 184 p.
14. Vikram Vaswani. MySQL: The Complete Reference". JAV, 2006. 544 p.
15. Marc DeLisle. Creating your MySQL Database. Norvegija, 2006. 105 p.
16. Ben-gan I., Sarka D., Wolter R. Inside Microsoft SQL Server 2005. T-SQL Programming. JAV, 2007. 532 p.
17. Кевин Луни, Боб Брила. Oracle 10g. Настольная книга администратора баз данных. Maskva, 2008. 752 p.
18. Duomenų bazės valdymo sistemų specifikacijos. Iš Road Test [interaktyvus]. [žiūrėta 2008-04-19]. Prieiga per internetą: http://www.builderau.com.au/architect/database/soa/Road-test-Four-databases-tested/0,339024547,339224962-7,00.htm

19. Александр Вячеславович Фролов, Григорий Вячеславович Фролов. Создание приложений с базами данных для Интернета и интрасетей: практическое руководство. Maskva, 2000. 698 p.
20. Denise Gosnell, Matthew Reynolds, Bill Forgey. Visual Basic .NET Databases. JAV, 2005. 689 p.
21. Michael Rawlins. Using XML with Legacy Business Applications. JAV, 2003. 624 p.

Research in XML documents support in open source data base management systems.

Summary

In this study, has been created software, that support XML documents interfacing with open source data base management systems.

This system enables in XML document write SQL query or queries. When this XML document (file) will be requested from internet browser, this system generate a new XML document in which instead placed SQL query or queries, would be inserted generated query answer from data base written in XML format.

This system has been created similar as Microsoft product SQLXML. Only our system has a feature, that it interfacing with open source data base management systems, such as MySQL.