KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Virginijus Dirgėlas

Psichofiziologinės reabilitacijos pacientų informacinės sistemos kūrimas ir tyrimas

Magistro darbas

Vadovas
doc. dr. V. Jusas

KAUNAS, 2005

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. E. Bareiša
2005-05-23

Psichofiziologinės reabilitacijos pacientų informacinės sistemos kūrimas ir tyrimas

Informatikos mokslo magistro baigiamasis darbas

Kalbos konsultantė
Lietuvių k. katedros lekt.
dr. J. Mikelionienė
2005-05-23

Recenzentas
doc. dr. A. Lenkevičius
2005-05-23

Vadovas
doc. dr. V. Jusas
2005-05-30

Atliko
IFM 9/2 gr. stud.
V. Dirgėlas
2005-05-23

KAUNAS, 2005

SUMMARY

During development of information system, we have problems to choose appropriate architecture, technologies and methods. The conception of information system, designing methods, problems, related with information system development is given in this work. There are mentioned the main security requirement for hospital information system.

We analyze created system “Information system of psychophysiology rehabilitation patients”. There are mentioned the main cases why client/server architecture, new technology adaptation were selected. The comparison of methods for data interface realization: PL/SQL language written and stored procedures and query generation is given. The selected method rating - how it speed system developing and how it influence system performance are evaluated. There are given experiments, which decrease system performance by selected methods.

TURINYS

Lentelių sąrašas

Paveikslų sąrašas

1. Įvadas

Pastaruoju metu augant programinės įrangos paklausai daugėja ir jos kūrėjų. Todėl rinkoje atsiranda didelė konkurencija. Tenka nemažai spręsti, kaip paskatinti klientą, kad jis pasirinktų mūsų sukurtą programinę įrangą. Taip pat ir klientui yra sunkus uždavinys pasirinkti programinės įrangos kūrėją. Vis greitėjant tempams, kiekvienas klientas nori gauti kokybišką produktą kuo greičiau, užsitikrinti, kad įsigyta programinė įranga bus prižiūrima ir tobulinama. Kuo greičiau įgyvendinami norimi nauji pakeitimai, taisomos klaidos. Visa tai pasiekti reikia pasirinkti metodą, galintį užtikrinti spartesnį PĮ kūrimą ir priežiūrą. Metodai priklauso nuo konkrečios PĮ, kurią reikia sukurti.

Kuriant informacines sistemas, svarbu suprasti, kas yra informacinė sistema, kokie jos pagrindiniai kriterijai, reikalavimai, savybės. Norint, kad informacinė sistema išliktų ir būtų naudojama, reikia, kad ji dirbtų sparčiai ir būtų atnaujinama. Informacijos pokyčiai, su ja susiję reikalavimų pokyčiai reikalauja informacinių sistemų tobulinimo, atnaujinimo. Todėl atsiranda problema, kaip jas sukurti greičiau ir mažesniais kaštais, atnaujinti, pritaikyti pasikeitusiems reikalavimams. Tenka gerai apsvarstyti būsimą architektūrą ir technologijas, užtikrinančias lengvą sistemos priežiūrą.

1.1. Tikslas

Informacinių sistemų pagrindinis elementas – duomenys. Dažniausi su jais susiję veiksmai:

- įrašymas,
- išrinkimas,
- atnaujinimas.

Kai informacinės sistemos būna mažos ir nesikeičia duomenys, problema pasirinkti metodus, architektūra, technologijas nėra aktuali. Susiduriant su didelėmis sistemomis, kai vieną objektą apibūdina daugybė atributų, problema pasidaro aktuali. Reikia iš karto nuspręsti kaip bus apdorojami objektai, kokį metodą pasirinkti, kad jis sumažintų programuotojo rankinį darbą įvedant atributų vardus, nurodant jų reikšmes.

Darbe bus bandoma pagrįsti kuriamos informacinės sistemos architektūros, technologijų pasirinkimą. Taip pat bus analizuojami metodas, kurie leistų išspręsti problemą su duomenų

dažniausiais veiksmais, atliekant išsaugotų procedūrų pakeitimą sugeneruotomis užklausomis ir palyginti su standartiniu kūrimo metodu naudojant PL/SQL kalba sukurtomis užklausomis.

1.2. Santrauka

Kuriant informacines sistemas, susiduriama su problema kaip pasirinkti tinkamą architektūrą, technologijas, metodus. Šiame darbe pateikiama informacinės sistemos samprata. Išvardijami projektavimo metodai, problemos. Aptariami pagrindiniai saugumo reikalavimai ligoninės informacinėms sistemoms.

Analizuojamas sukurtos sistemos “Psichofiziologinės reabilitacijos pacientų informacinė sistema” kūrimas. Pagrindžiamas kliento-serverio architektūros pasirinkimas, naujų technologijų pritaikymas. Stengiamasi pateikti metodus, kurie sumažintų informacinių sistemų kūrimo ir priežiūros kaštus. Pateikiamas metodų palyginimas duomenų sąsajos realizacijai: PL/SQL kalboje parašytų išsaugotų procedūrų ir užklausų generavimo. Pateikiamas pasirinkto metodo įvertinimas - kaip paspartina sistemos kūrimą ir įtakoja sistemos našumą. Pateikiami eksperimentiniai bandymai, parodantys kaip sumažina sistemos našumą pasirinktais metodais.

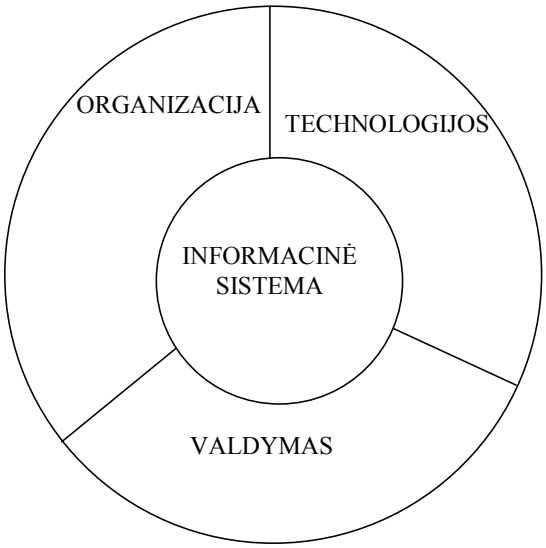
2. Analitinė dalis

2.1. Informacinė sistema

Apie informacinės sistemas galima rasti ne vieną apibrėžimą. Dažniausiai ji apibrėžiama įspraudžiant į specifinius rėmus, kur informacinė sistema yra naudojama. Štai keletas informacinės sistemos pavyzdžių:

- tai procesas, kuris paverčia duomenis į informaciją ir informaciją paverčia į žinias;
- organizuoti komponentai sukurti rinkti, apdoroti, persiųsti ir laikyti duomenis tam, kad duotų informacijos esant pareikalavimui.

Taip pat jei pažvelgsime į konkretesnį subjektą kaip kažkokią organizaciją, tai informacinę sistemą galima atvaizduoti grafiškai, tai pateikiama 2.1. paveiksle.



2.1. pav. Informacinė sistema

Jei pasigilintume į tai ką sako teorija apie sistemas, galima rasti tokių sistemos savybių:

- informacinė sistema egzistuoja kažkokioje aplinkoje;
- informacinę sistemą skiria kažkokia riba nuo tos aplinkos, kurioje ji yra;
- ji gauna kažkokius įėjimus iš aplinkos ir duoda išėjimus į aplinką;
- sistema turi sąsajas, ji gali bendrauti su kitomis sistemomis;
- sistemą gali sudaryti posistemiai, kurie savo ruožtu taip pat gali būti sudarytos iš posistemių;

- sistemos, kurios išlieka turi valdymo mechanizmą;
- daugelis sistemų turi specialius posistemius, kurių tikslas kontroliuoti visą sistemą.

Norint atsakyti, kas yra informacinės sistemos analizė, galima pasakyti, kad tai rinkinys notacijų, metodologijų ir įrankių, naudojamų surinkti detalią informaciją apie problemas informacinei sistemai projektuoti ir sukurti. Sistemos analizė turi užtikrinti, kad pateikta informacinė sistema užtikrintų vartotojo poreikius, būtų pristatyta laiku ir jos atnaujinimas turėtų būti nebrangus.

Problemos, nustatant teisingą sistemos analizę, atsiranda dėl blogai apibrėžtų aplinkybių, dviprasmybių, nesuderinimų.

Informacinės sistemos analizės etapo rezultatas - tai reikalavimai. Jiems priklauso funkciniai ir nefunkciniai reikalavimai.

Kalbant apie sistemos projektavimą, galima pasakyti, kad tai specifikacijos realizuoti informacinę sistemą, į ją įeina: techninė įranga, kurioje bus įdiegta projektuojama sistema; programinė įranga, kur sistema veiks: operacinė sistema, duomenų bazės valdymo sistema, programavimo kalba ir t.t. Pateiktos programinės įrangos architektūra ir sąsajos tarp sistemos modulių.

Projektavimas nusako, kokios funkcijos naudojamos kiekviename modulyje ir ką kiekvienas modulis daro. Duomenų bazė, tai dalis informacinės sistemos, kuri bus laikoma duomenų bazės valdymo sistemoje arba failuose [1].

2.2. Informacinių sistemų projektavimas

2.2.1. Projektavimo metodai

Pastaraisiais metais pastebimai išaugo metodų ir palaikymo priemonių informacinės sistemos projektavimo procesui. Šie metodai suteikia paprastą grafinę notaciją atvaizduoti sistemos elementus ir taip projektuotojams palengvina spręsti projektavimo problemas. Naudojantis šiais metodais, galima gauti projektavimo scenarijų, kuris leidžia projektuotojui numatyti tiksliai aprašytus žingsnius. Tačiau projektavimo problemų mastas didėja ir paaiškėja nauji patikimumo reikalavimai net ir tada kai metodas yra aiškus ir tikslus. Vis sunkiau valdyti didėjantį struktūrinį projektuojamos sistemos sudėtingumą.

Logiška išvada paaiškėja iš šių problemų analizės, kad reikalingas būdas automatiškai atlikti daugiau validacijų ir taip projektuotojas duos projektą, kuris labiau tenkina numatytus

reikalavimus. Tai gali būti atlikta greičiau ir didesniu patikimumu. Daugelis tyrinėtojų pastebėjo, kad formalieji metodai yra būtini, norint pasiekti patikimumą. Tačiau ši viena išvada nepasiteisina praktiškai, nes pakankamai neįtikina tikrų projektuotojų, kurie tai suvokia kaip naštą, o ne kaip privalumą.

Todėl egzistuoja didelis poreikis priemonių, kurios pagrįstos formaliais metodais, kas rodo, kad šis suvokimas neteisingas, todėl pasitelkti formalius metodus yra tikslinga [2,3].

2.2.2. Projektavimo problemos

Nors įrankiai ir priemonės projektuoti programinę įrangą gerėja, projektai susiję su informacine sistema dažnai būna nesėkmingi. Pažiūrėję į statistiką, kuri sako kad 50% pabaigtų projektų viršijo biudžetą 60-190% ir tik 25% projektų buvo pabaigti laiku. Pasinaudojus kitais šaltiniais, sakoma, kad tik 16% projektų baigiama laiku ir neviršija biudžeto.

Dėl to viena agentūra užsiėmė patyrinėti ir suprasti informacinių sistemų projektų nesėkmes ir pasiūlyti patobulinimus šioje srityje. Pirmiausia galima pasakyti, kad informacinių sistemų projektai būna sėkmingi kai jie baigiami laiku, neviršija skirto biudžeto, pasiekia numatytą funkcionalumą ir yra aukštos kokybės. Kontrakto tipas įtakoja projekto sėkmę ir pagal jį galima stebėti projekto progresą. Pilnas projekto stebėjimas atneša geresnę projekto sėkmę.

Atlikus tyrimą buvo padarytos išvados ir paskelbtos priežastys, dėl kurių įvyksta nesėkmės kuriant informacinės sistemos projektus. Jos išvardintos keturios: pirmiausia, sudarytas kontraktas nebūna pakankamai detalizuotas ir tikslus. Organizacijos programuotajai nesiekia numatyto tikslo sutartyje ar nesiekia paskatinimo už darbą. Antra, projekto kūrimo stebėjimas nėra efektyvus: organizacija nepakankamai įtraukia į savo darbą įrankius ar metodus arba įrankiai ir metodai naudojami anksčiau dirbant su projektas sunkiai padėdavo su iškilusiomis projekto problemomis. Trečia: organizacija patiria nesėkmę, sprendžiant pagrindinių tikslų konfliktus, informacijos slėpimą ir programuotųjų išsisukinėjimus. Ketvirta: organizacijos nepasinaudoja efektyviais metodais programuojamų užduočių sprendimams. Gal vienu metu kažkokie metodai žymiai efektyvesni ir užduotys lengviau padaromos.

Kad informacinių sistemų kūrimo palengvinanti ir pasiekti geriausia rezultatą rekomenduojama atkreipti į šiuos aspektus:

- kontrakto tipas;
- projekto stebėjimas ir atliktų užduočių fiksavimas;
- pagrindinių tikslų konfliktų sprendimas;
- darbuotojų išsisukinėjimai;

- privačiai darbuotojų laikoma informacija;

programavimo užduočių alternatyvus sprendimų variantai[4].

2.3. Problema informacinių sistemų kūrime

Šiame darbe atkreipsime didžiausią dėmesį į problemas, kurios susijusios su kuriamos informacinės sistemos duomenimis. Galima teigti, kad pagrindinis informacinės sistemos elementas – teisingi duomenys. Nuo konkrečios informacinės sistemos priklauso jų gausa, dažnas keitimasis, naujų duomenų tipų atsiradimas. Problema padidėja kai susiduriama su objektais, kuriuos apibūdina daugybė skirtingų parametrų. Atsiranda didžiulė rizika kūrimo etape sumaišyti laukus, reikšmių priskyrimą, sąsajų suderinamumą tarp duomenų ir veiklos paslaugų. Todėl reikia pasirinkti tokią sistemos architektūrą, kuri leistų automatizuotai manipuliuoti duomenimis ir reikėtų kuo mažiau programuotojo rankinio darbo apdorojant juos.

Informacinės sistemos, kurios pasižymi daugiaatribučiais objektais, kelia problemų ne tik programinės įrangos kūrėjams, bet ir pačiam klientui – vartotojui. Didžiulės duomenų įvedimo formos turi būti patogios ir vartotojui lanksčios. Vartotojui turi būti suteikta galimybė užpildyti nepabaigtą formą, jei formoje yra pakankamai didelis kiekis laukų, o tai didina riziką, kad duomenys ateityje bus nevisiškai užpildyti.

2.3.1. Architektūros pasirinkimas

Pasirenkant architektūrą, reikia gerai apgalvoti, kaip kuriamas projektas bus vystomas po projekto sukūrimo. Kaip lengviau valdyti ir įgyvendinti atsiradusius reikalavimų pasikeitimus projekto įgyvendinimo stadijoje? Mūsų nagrinėjamos sistemos “Psichofiziologinės reabilitacijos pacientų informacinės sistemos” architektūra parinkta atsižvelgiant į būsimus pakeitimus. Pirmiausia parinkta kliento-serverio architektūra leidžia turėti visą informacinę sistemą (duomenis, programos kodą) vienoje vietoje. Tai suteikia naudos, kai atliekami pakeitimai, kurie įtakoja visus šios informacinės sistemos vartotojus. Parinkus vartotojo klientą – interneto naršyklę, išvengiame pakeitimų grafinėje vartotojo sąsajoje pas kiekvieną vartotoją ir užtikriname, kad visi vartotojai naudosis vienoda sistemos versija. Taip pat pasirinkę interneto naršyklę nebereikia kurti vartotojo sąsajos, vartotojai dažniausiai yra įpratę naudotis naršykle.

Centralizuotas duomenų saugojimas pasirinktoje duomenų bazės valdymo sistemoje užtikrina duomenų integralumą, išvengiama jų dubliavimo. Pasirenkant duomenų bazės sistemą,

reikia atsižvelgti į konkrečios sistemos reikalavimus. Svarbu atkreipti dėmesį kiek numatoma duomenų, kiek bus saugoma įrašų, koks reikalingas našumas, kokios galimybės replikuoti duomenis, kad padidintumėme DBVS našumą. Reikia atskirti sistemos visą veiklos logiką nuo duomenų logikos, kad būtų lengvesnis pernešimas į kita duomenų bazės valdymo sistemą esant poreikiui ateityje.

"Psichofiziologinės reabilitacijos pacientų informacinės sistemos" pagrindinis elementas – mėginai. Jų yra 21 ir beveik kiekvieną mėginį apibūdina didelis kiekis atributų (vidutiniškai apie 70). Reikia numatyti, kad mėginių skaičius gali pasipildyti, o taip pat pasipildyti mėginio atributai. Tam tikslui architektūra turėtų užtikrinti nepriklausomumą nuo mėginių skaičiaus ir mėginio atributų skaičiaus. Keičiantis tik mėginių laukams, kurie jų neidentifikuoja, reikia užtikrinti, kad nereikėtų įsikišti programuotojui į visus architektūros paketus.

2.3.2. Technologijų pasirinkimas

Pagal kiekvieno projekto tipą ir poreikius, pasirenkama tinkamiausia programavimo technologija. Pasirenkant ją, reiktų atkreipti dėmesį ar mes turime pakankamai specialistų, kurie galėtų šia technologija realizuoti projektą, ar po projekto sukūrimo bus galima lengvai rasti programuotojų, išmanančių šias pasirinktas technologijas. Įvertinti ar patenkins našumo reikalavimus, ar bus galima išplėsti sistemą. Ar suderinama su kitomis technologijomis? Ar galima integruoti komponentus iš kitų kūrėjų?

Mūsų analizuojame projekte parinktos technologijos, kurios atitinka išsikeltus tikslus. Duomenų bazės valdymo sistemą naudosime Oracle 9.1. Užsakovo reikalavimuose buvo apribotas sprendimas, būtent šia duomenų bazės valdymo sistema, kuri pasižymi našumu esant dideliems duomenų kiekiams, išplečiamumu ir nuolatiniu gamintojo palaikymu.

Sistemai pasiekti bus naudojama interneto prieiga per HTTP protokolą, kurį aptarnaus Apache programa. Pasirinkus šį protokolą yra išvengiama papildomų problemų, kurios gali iškilti vartotojams. Dažniausiai ugniasienės sukonfigūruotos taip, kad HTTP protokolas ir kiti žinomi yra praleidžiami, o visą kita blokuojama.

Visai logikai užtikrinti pasirinkta plačiai paplitusi programavimo kalba PHP [5,6]. Šios kalbos pasirinkimą lemia jos nepriklausomumas nuo techninės įrangos, taip pat nuo operacinės sistemos, todėl pasikeitus operacinei sistemai ar techninei įrangai ateityje neiškils problemos dėl atnaujinimo. Taip pat pakankamai spartus ir lengvas darbas su įvairiomis duomenų bazės valdymo sistemomis leis taip pat lengvą DBVS pakeitimą.

2.3.3. Duomenų struktūros pasirinkimas

Projektuojant duomenų struktūrą, išskyrus reikiamas esybes, jas charakterizuojančius atributus, bei nustačius tarpusavio ryšius, būtina sukurti reikiamus raktus, indeksus. Būtina užtikrinti kiek galima geresnį duomenų panaudojamumą bei vientisumą. Tam reikia atlikti duomenų struktūros normalizavimą – „sutvarkymą“ pagal tam tikras taisykles. Normalizavimas – procesas, taikomas projektuojant reliacines duomenų bazes. Jo tikslai:

1. Duomenų bazėje turi būti saugomi visi reikalingi duomenys.
2. Duomenų perteklius turi būti kiek galima mažesnis.
3. Visos reikšmės rašomos į joms skirto duomenų tipo laukus.
4. Duomenų atnaujinimas turi būti vykdomas efektyviai.
5. Vengiama netyčinio duomenų praradimo pavojaus.

Paprastai yra normalizuojama mažiausiai iki 3 normalinės formos. Tokiu būdu duomenų bazė tampa apsaugota nuo „galimų duomenų sugadinimų, vadinamųjų „įterpimo anomalijų“, „šalinimo anomalijų“ ir „atnaujinimo anomalijų”. Šio tipo klaidos dažnos nepakankamai normalizuotose bazėse, kuriose duomenys gali būti vienaip ar kitaip sudubliuoti (pertekliniai). Pavyzdžiui, duomenų įrašymo atveju gali pasitaikyti taip, jog duomenis reikės įrašyti daugiau nei į vieną vietą, šalinimo metu gali būti pašalinti ne tik reikiami, bet ir tėviniai duomenys, o atnaujinimo veiksmas apima tiek įrašymą, tiek šalinimą, todėl galimos abiejų tipų anomalijų kombinacijos.

Analizuojamame projekte duomenų struktūra buvo pasirinkta taip, kad būtų sumažinta ryšių tarp lentelių skaičius. To siekiama tam, kad pagreitintumėme duomenų išrinkimą. Kiekvienas mėginys saugomas atskiroje lentelėje su jį apibudinančiais atributais ir jį identifikuojančiais laukais. Detalesnę struktūra bus aprašyta ir pateikta skyriuje 3.7. Duomenų vaizdas.

2.3.4. Duomenų sąsaja

Didžiausia problema atsiranda norint apdoroti didelius kiekius duomenų. Tai yra kiekvieną tyrimą sudaro mėginiai, kurie turi daug reikšmių, kurias reikia išsaugoti, keisti ir atvaizduoti. Todėl svarbiausia vieta jų surašymas į duomenų bazę. Čia ir iškyla problema kaip realizuoti duomenų bazės sąsają tarp duomenų bazės valdymo sistemos ir veiklos logikos, kuri atiduoda duomenis vartotojui ar įveda ją. Vienas iš realizacijos būdų tai yra PL/SQL kalba [7] kurti kiekvienam mėginiui procedūrą arba generuoti užklausas ir jas siųsti į duomenų bazės

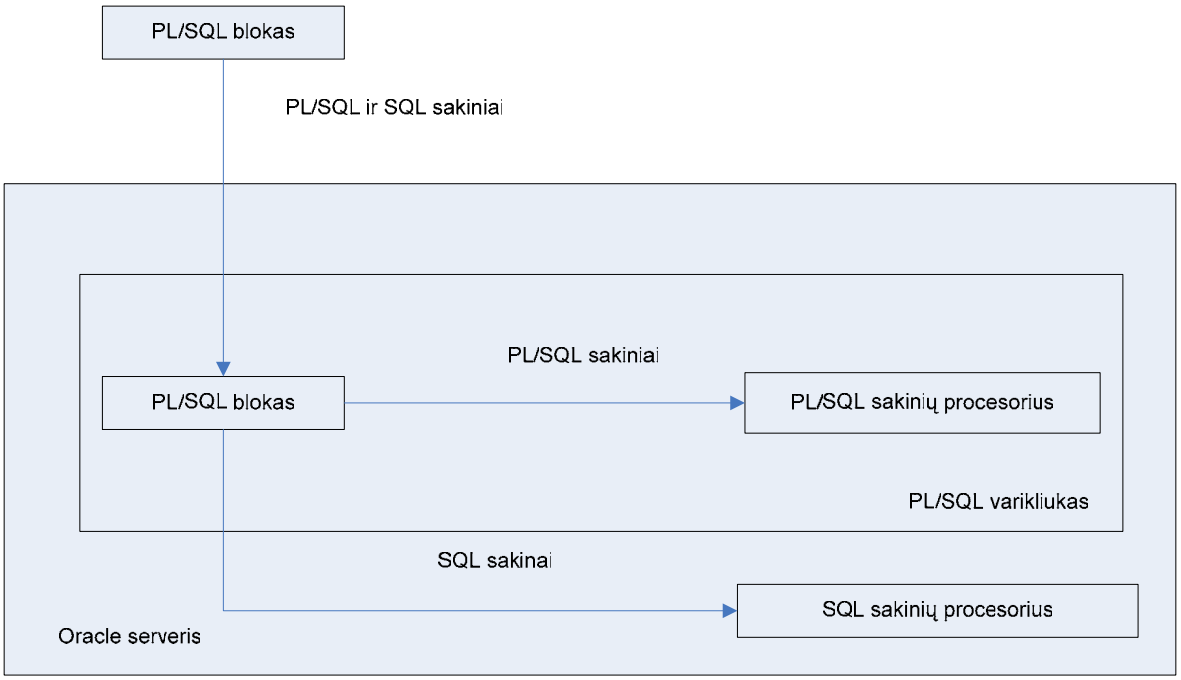
valdymo sistemą. Bet naudojimas jos tiesiogiai priklauso nuo mėginių skaičiaus ir mėginį apibudinantį atributų skaičiaus. O tai nėra labai gerai, nes prieštarauja mūsų norimos architektūros tikslams būti kuo mažiau priklausomiems nuo mėginių skaičiaus.

PL/SQL

Daugelis Oracle sistemų yra sukurtų naudojant kliento-serverio architektūrą. Oracle duomenų bazė įdiegta darbinėje stotyje. Programa, kurie kviečia užklausas į šią duomenų bazę būna kliento kompiuteryje. Šios programos gali būti parašytos naudojant C, Java ar PL/SQL.

Kadangi PL/SQL yra kaip ir visos kitos programavimo kalbos, ji turi savo sintaksę, taisykles, kurios nustato kaip programos sakiniai veikia tarpusavyje. Svarbu suprasti, kad PL/SQL nėra atskira programavimo kalba, ji yra dalis Oracle DBVS ir gali egzistuoji kliento kompiuteryje ir darbinėje stotyje. Todėl lengvai galime pernešti modulius iš kliento kompiuterio į darbinę stotį ir atvirkščiai.

Abiejose aplinkose bet kuris PL/SQL blokas ar paprogramė yra apdorojama PL/SQL varikio, kuris yra specialus kiekvieno Oracle produkto komponentas. PL/SQL varikis apdoroja ir vykdo PL/SQL sakinius, siunčia visus SQL sakinius SQL sakinių procesoriui. SQL sakinių procesorius visada būna Oracle serveryje. 2.2. paveiksle vaizduojamas PL/SQL variklio išdėstymas Oracle serveryje.



2.2. pav. PL/SQL variklis ir Oracle serveris.

Kai PL/SQL blokas yra patalpintas serveryje, visas PL/SQL blokas yra persiunčiamas į PL/SQL variklį. Šį gautą bloką apdoroja kaip pavaizduota 2.2 paveiksle.

Kada PL/SQL variklis yra patalpintas kliento kompiuteryje, visas PL/SQL vykdymas yra atliekamas kliento kompiuteryje. Visi SQL sakiniai, kurie egzistuoja šiuose blokuose yra įdedami į PL/SQL bloką ir siunčiami į Oracle serverį tolimesniam apdorojimui. Kada PL/SQL blokas neturi SQL sakinių, visas blokas apdorojamas kliento kompiuteryje[7].

Metaprogramavimas

Pasiremiant metaprogramavimo principais bus generuojamos užklausos. Todėl šiame skyrelyje susipažinsime su pagrindiniais metaprogramvimo principais ir privalumais.

Metaprogramavimu vadinamas toks programavimo būdas, kai tikslia kalba užrašome bazinį (konkretų, ar mažai apibendrintą) srities funkcionalumą, o metakalba išreiškiame bendrinį funkcionalumą tam, kad išplėstume programos (komponento) atkartojamumo laipsnį ir padidintume jos pritaikomumą.

Metakalba vadinama aukštesnio lygio kalba, kurios paskirtis yra modifikuoti žemesnio lygio kalba (dar vadinama tikslo kalba) parašytas programas.

Tikslo kalba – kalba, skirta srities funkcionalumui išreikšti arba algoritmams aprašyti; gali būti arba algoritminė programavimo kalba, arba bendros paskirties kalba. Šiame darbe nagrinėjamam užklausų generavimui tikslo bus SQL kalbą.

Kuriant užklausų generatorių pasinaudosime metaprogramavimo pagrindiniu mechanizmu parametrizavimu.

Parametrizavimas – tai pagrindinis mechanizmas, kurį taikant yra įgyvendinami metaprogramavimo principai ir koncepcija. Parametrizavimo metu atskiros srities koncepcijos yra specifikuojamos bendriniais parametrais ir realizuojamos naudojant įvairias metakalbos abstrakcijas [9].

Užklausų generavimas

Užklausų generavimo metodas remiasi metaprogramavimo principais. Tikslas parašyti tokius metodus, kuriais būtų galima sugeneruoti užklausas tinkančias panaudoti visame kuriamame projekte, kad išvengti konkrečių išsaugotų procedūrų ar užklausų. Šiam tikslui bus naudojama kaip meta kalba PHP programavimo kalba. Ši kalba naudojama kaip pagrindinė kalba šiame projekte, kuri atsakinga už veiklos funkcijas ir atvaizdavimą, bei užtikrina sąsaja su

duomenų bazės valdymo sistema. Šiuo atveju tikslo kalba bus SQL (struktūrizuota užklausų kalba).

Dėl pakankamai lanksčios PHP kalbos, kuri netipizuota leidžia sukurti neapibrėžtus parametrus. Užklausų generavimo metodai turės parametrus masyvus, į kuriuos galima paduoti neapibrėžtą elementų skaičius.

Kad išvengti papildomo parametrų masyvo, laukų pavadinimams apibudinantį objektą ir šių laukų reikšmės, naudosime vieną masyvą, kurio indekso pavadinimas atstoja objekto atributo pavadinimą, o masyvo reikšmė pagal tą indeksą objekto atributo reikšmę:

$array['code'] – reiškia, kad mes įrašysime objektui atributą *code* reikšmę kuri saugoma *$array['code']* kintamajame.

Įvedus dar papildomus pažymėjimus masyvo indeksų pavadinimams leidžia išskaidyti duomenis pagal tai ar duomenis skirti įrašyti į duomenų lenteles ar pagal juos išrinkti duomenis:

$array['u_code']

$array['u_tr_nr']

$array['i_field1']

$array['i_field2']

Sakysime masyvo indeksai kurie prasideda *i_* bus naudojami įrašyti reikšmes tiems laukams, kuriuos atitinka šie indeksai, o *u_* laukai gali žymėti kuriam objektui reikalinga atnaujinti informaciją pagal juos atitinkančius laukus.

Kaip bebūtų reikės sukurti kiekvienam užklausų tipui: įrašymui, atnaujinimui, ištrynimui ir išrinkimui atskirus metodus, kuriuo generuoja šiems veiksmas atlikti užklausas.

2.4. Informacinės sistemos saugumas

Kada organizacija tampa labai priklausoma nuo informacinės sistemos strateginių privalumų ir veiklos, informacinės sistemos saugumas pasidaro didelė ir labai svarbi problema. Šiandiena, susijusios su elektroninio verslo aplinka, apsaugos problemos yra pirmaeilės. Vadovybė turėtų investuoti į informacinės sistemos apsaugą, kad būtų užkirstas piktnaudžiavimas.

2.4.1. Duomenų saugumo reikalavimai

Sveikatos institucijos teikdamos savo paslaugas kaupia apie pacientus informaciją popieriniame formate. Tačiau tai suteikia nepatogumų ir jau daugelį sveikatos institucijų pradeda pereiti prie elektroninio pacientų informacijos kaupimo. Elektroninis pacientų informacijos kaupimas suteikia privalumų: be gero įskaitomumo jie suteikia priėjimą prie paciento duomenų keletui gydytojų vienu metu skirtingose vietose, taip pat jie pašalina bereikalingų dokumentų laikymą. Toks būdas gali optimizuoti tikslumą, užbaigtumą, kainą.

Kad suteikti pacientams aukštos kvalifikacijos sveikatos paslaugas dažniausiai į tai įtraukiama daugiau nei viena institucija. Todėl tenka dalintis pacientų informacija. Elektroninis pacientų informacijos saugojimas kaip tinka šiam atvejui. Tada susijusios sveikatos apsaugos įstaigos gali prieiti prie šių įrašų internetu ar tinklo pagalba. Tačiau svarbiausia užtikrinti duomenų saugumą ir duomenų apsaugojimą. Internetas suteikia lengva duomenų pasikeitimą tarp institucijų, bet tai padidina pavojų, kad svarbi pacientų sveikatos informacija gali būti pakeista arba pasinaudota ja. Todėl toliau bus paminėta būtent apie pacientų duomenų saugumą.

Pacientų sveikatos informacijos apsauga ir saugumas nėra vien paciento interesas, bet daugelyje išsivysčiusių šalių to reikalauja įstatymai. Todėl galima pasakyti, kad duomenų saugumas ir jų apsauga turi garantuoti tokias 5 esminius siekius:

- konfidencialumas - tai reiškia, kad paciento duomenys nebus prieinami arba paimti asmenų, kuriems ši informacija nesuteikiama;
- integralumas – užtikrina, kad paciento duomenys nebus ištrinti ar pakeisti asmenų, kuriems tai nesuteikta
- autentiškumas – duomenys prieinamu tik įrodžius asmens tapatybę, kuriam galima pasiekti duomenis
- atsakingumas – visi veiksmai su paciento duomenimis turi būti fiksuoti ypatingai pakeitimai paciento duomenims.
- naudingumas – paciento duomenys pasiekiami tik autorizuotiems vartotojams.

Metodai pasiekti šiuos saugumo reikalavimus.

Daugelis metodų užtikrinantis pacientų duomenų saugumą ir apsaugą yra pagrįsti kriptografijos procedūromis. Šifravimas yra naudojamas užkoduoti paprastai parašytas žinutes, kurios paprastiems vartotojams perskaitomos kaip užšifruotas tekstas, o įskaitomas tik autorizuotiems asmenims. Yra du pagrindiniai šifravimo algoritmai:

Simetriškas algoritmas – naudojamas tik vienas raktas užšifruoti ir atkoduoti pranešimus. Šio algoritmo privalumas greitas žinučių užšifravimas ir atkodavimas. Naudojant šį metodą saugumo problema atsiranda perduodant raktą, kuris gali atkoduoti ar užkoduoti žinutes. Vienas iš šių algoritmų yra DES

Asimetriškas algoritmas – naudojami 2 susiję raktai užkoduoti ir atkoduoti žinutes. Naudojami privatus ir viešas raktai, kur viešas raktas yra visiems žinomas ir juo galima užkoduoti pranešimus ir atkoduoti galima tik naudojant privatu savo raktą. Šiuo būdu nėra problemos raktų pasidalijime, bet jis lėtesnis negu simetriškas algoritmas. Vienas iš šių algoritmų – RSA.

Taigi pasinaudojant abiejų būdu privalumais ir trūkumas ir norit pasiekti auštą saugumą naudojami abiejų metodų kombinacijos.

Kad būtų galima naudotis pacientų duomenimis tarp dviejų bendradarbiaujančių sveikatos institucijų reikalingas saugus ryšys tarp jų. Tam galima naudoti internetą, bet kaip jau buvo minėta tai nėra pakankamai saugu, todėl vidiniame tinkle statoma ugniasienė. Taip pat naudojant ugniasienes suteikiama galimybė padaryti saugų ryšį tarp dviejų įstaigų – VPN *(virtual private network*). Naudojantis juo internete sukuriamas tunelis, kuris apsaugotas kriptografinėmis procedūromis. Tokį virtualų privatų tinklą galima sudaryti ne tik tarp dviejų ugniasienių, bet ir tarp darbo stoties ir kliento kuris yra kitoje institucijoje.

Kitas alternatyvus metodas VPN tai SSL (*Secure socket layers*), kuris gali būti naudojamas sukurti saugu ryšį. Šitas metodas garantuoja saugumą, mažą kaina perduoti informaciją nuo vieno galo iki kito per nesaugų internetą.

Kitas metodas, kurio kaina palyginti aukšta ir metodas nepasižymintis lankstumu yra fiksuotos linijos tarp dviejų bendradarbiaujančių institucijų, bet šitas būdas suteikia saugiausia būdą persiųsti konfidencialius duomenis.

Vien saugaus ryšio neužtenka, reikalinga autorizacija, kad būtų galima pasiekti kliento duomenis. Todėl informacinė sistema turi nustatyti vartotoją ir suteikti jam tokias teisės, kokios apibrėžtos jo grupei. Vokietijoje egzistuoja tokia autorizacija, kad vien tik gydytojas gali pasiekti paciento duomenis, kurie yra kitoje institucijoje ir tik tuo atveju jei jis yra įtrauktas į šio paciento gydytojus. Šiai problemai spręsti buvo sukurta RAD (Resource Access Decision), kuri turi mechanizmą nustatanti ryšius tarp gydytojų ir pacientų. Nustato priėjimą prie paciento duomenų[10].

PCASSO [11] projektas naudoja penkis lygius užtikrinti paciento sveikatos informacijos konfidencialumą ir išskiria skirtingas vartotojų vaidmenis: tyrinėtojas, pacientas, pagrindinis

prižiūrėtojas, antrinis prižiūrėtojas ir kritiškos padėties prižiūrėtojas. Kiekvienam vaidmeniui suteikiamos skirtingas apsaugos lygis ir teisės, atsakomybė už pacientus.

Būdas autorizuoti vartotojus panaudojant slaptažodžius nėra geras, nes dažniai jis pamirštamas ir užrašomas kur nors ant lapelio. Norint patikimos autorizacijos, reikia naudoti kitokias technologijas, kaip kortelės identifikuojantį asmenį ar biometrines procedūras [12,13].

Analizuojamame projekte svarbiausia užtikrinti, kad duomenys nebūtų pasiekiami asmenims, kurie neturi priėjimo prie informacinės sistemos. Tam prie informacinės sistemos priėjimas bus leistas tik su jai skirtu vartotoju ir prisijungimo slaptažodžiu. Kad būtų apsaugoti siunčiami duomenys reikia naudoti HTTPS protokolą.

2.5. Bandomasis projektas

Šiam darbui sukurtas projektas “Psichofiziologinės reabilitacijos pacientų informacinė sistema” – tai informacinė sistema, integruota interneto aplinkoje. Produktas skirtas psichofiziologinės reabilitacijos ligoninėms, kurioms reikia kaupti informaciją apie pacientų tyrimo rezultatus. Sukurtoje sistemoje bus galima registruoti atvykusius reabilitacijai pacientus, atliktus tyrimus ir saugoti tolimesnei statistinei analizei.

Kuriamas produktas turės atlikti šias pagrindines funkcijas:

- įtraukti paciento tyrimų rezultatus;
- pacientų tyrimų rezultatų išsaugojimas duomenų bazėje, galimybė juos koreguoti ar peržiūrėti;
- atlikti tolimesnę statistinę analizę.

Programa taip pat turės generuoti reikiamas ataskaitas, remiantis suvestais ir apskaičiuotais pacientų tyrimų rezultatais, pateikti jas patogia forma, kad būtų galima spausdinti, ar persiųsti kolegoms.

Kuriamas produktas bus pasiekiamas naudojant interneto naršyklę. Vartotojui teks surinkti adresą, kuriuo bus pasiekiama informacinė sistema. Prie sistemos galima prisijungti tik leistiniems vartotojams. Todėl sistema visada turi paprašyti prisijungimo duomenų, kad būtų identifikuojamas vartotojas.

3. Projektinė dalis

3.1. Projekto paskirtis

Produktas skirtas psichofiziologinės reabilitacijos ligoninėms, kurioms reikia kaupti informaciją apie pacientų tyrimo rezultatus. Sukurtoje sistemoje bus galima registruotis atvykusiems reabilitacijai pacientams, atliktus tyrimus saugoti tolimesnei statistinei analizei.

Kuriamas produktas turės atlikti šias pagrindines funkcijas:

- įtraukti paciento tyrimų rezultatus;
- pacientų tyrimų rezultatai išsaugojimas duomenų bazėje ir pasiekiamumas koregavimui ar peržiūrai;
- atlikti tolimesnę statistinę analizę.

Programa taip pat turės generuoti reikiamas ataskaitas, remiantis suvestais ir apskaičiuotais pacientų tyrimų rezultatais, pateikti jas patogioje formoje, kad būtų galima spausdinti, ar persiųsti kolegoms.

3.2. Vartotojai

Būsimieji produkto vartotojai - medicininis personalas, kuris turi pakankama išsilavinimą greitai išmokti naudotis kuriama informacine sistema.

Produkto vartotojai turi turėti minimalius darbo su kompiuteriais įgūdžius. Kadangi informacinė sistema bus pasiekiama internetu, tai yra interneto naršykle, tai išvengs papildomų programų įdiegimo vartotojo kompiuteriuose ir reikės minimalių pastangų vartotojų apmokymui.

Sistema naudotis galės tik autorizuoti vartotojai, todėl bet kuris kitas priėjimas prie sistemos ar sistemos duomenų bus uždraustas.

Skiriami tokie būsimi informacinės sistemos vartotojai:

Medicinos sesuo – vartotojas, kuris įvedinės mėginių informaciją į sistemą ir taip pat juos redaguos.

Gydytojas – vartotojas, kuris naudosis ir peržiūrinės įvestus mėginius ir analizuos juos.

3.3. Projekto apribojimai

3.3.1. Įpareigojantys reikalavimai

Būsima informacinė sistema turi sugebėti valdyti didelius informacijos kiekius. Turi būti numatyta, kad įvesti mėginių duomenis bus visada pasiekiama ir vykdymo laikas neturi būti ilgas. Taip pat sistemą gali tekti papildyti naujomis savybėmis, todėl projektuojama sistema turi būti lengvai keičiama ir nesugadinti esančio funkcionalumo.

Sistema neturi būti sudėtinga ir nereikalauti ilgo vartotojų apmokymo, kad ja būtų galima sėkmingai naudotis.

3.3.2. Apribojimai sprendimui

Projektuojama sistema, bus užsakovui nauja, todėl daug apribojimų išskirti labai sunku. Svarbiausi reikalavimai:

- informacinę sistemą pasiekti interneto naršykle;
- informacinę sistemą gali naudoti lygiagrečiai visi vartotojai

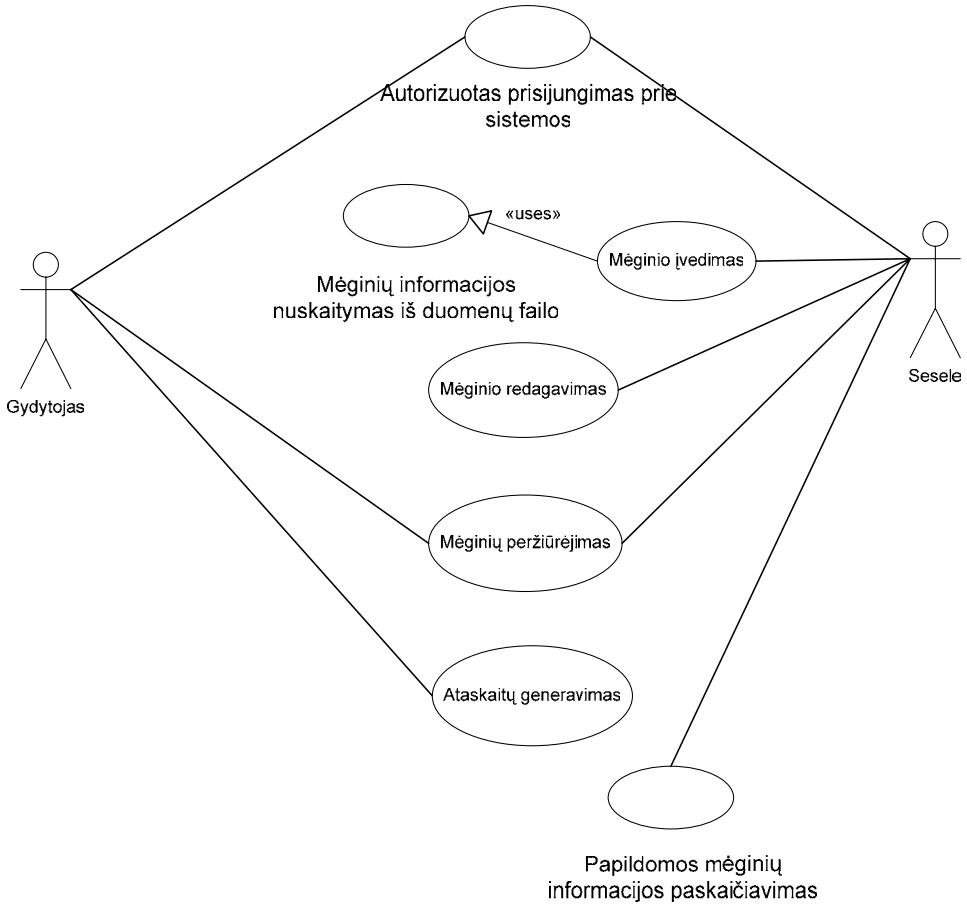
3.3.3. Diegimo aplinka

Programinė įranga veiks LINUX šeimos operacinėje sistemoje. Šioje darbinėje stotyje papildomai turės būti įdiegta:

- Oracle duomenų bazių valdymo sistema;
- HTTP paslauga – dažniausiai naudojamas Apache;
- Programavimo kalbos kompiliatorius – PHP 4 arba aukštesnė versija.

Kliento kompiuteriuose instaliuoti papildomai nieko nereikės, nes informacinei sistemai pasiekti naudos internetinę naršyklę, kuri būna įdiegta kartu su operacine sistema

3.4. Pagrindiniai reikalavimai



3.1. pav. Panaudojimų atvejų diagrama

3.4.1. Panaudojimo atvejai

1. PANAUDOJIMO ATVEJIS:	Mėginio įvedimas
Vartotojas/Aktorius:	Seselė
Aprašas:	Įvedamas atlikto mėginio informaciją konkrečiam sistemoje esančiam pacientui
Prieš sąlyga:	Sistemoje turi būti užregistruotas pacientas, kuriam įvedamas mėginys, aktorius turi būti prisijungęs prie sistemos
Sužadinimo sąlyga:	Buvo pacientui atliktas naujas tyrimas
Po-sąlyga:	Sistemoje užregistruojamas pacientui mėginys

2. PANAUDOJIMO ATVEJIS:	Mėginio redagavimas
Vartotojas/Aktorius:	Seselė
Aprašas:	Atliekama įvesto mėginio redagavimas, taip pat galima pabaigti įvedinėti pradėta įvesti mėginį
Prieš sąlyga:	Sistemoje turi būti įvestas redaguojamas mėginys, aktorius turi būti prisijungęs prie sistemos
Sužadinimo sąlyga:	Įvestame mėginyje buvo pastebėtos klaidos arba norima pabaigti įvesti pradėtą įvedinėti mėginį
Po-sąlyga:	Sistemoje užregistruojami mėginio pakeitimai

3. PANAUDOJIMO ATVEJIS:	Mėginių peržiūrėjimas
Vartotojas/Aktorius:	Seselė, Gydytojas, 2PA
Aprašas:	Išvedami įvesti mėginiai pagal aktoriaus nurodytus kriterijus
Prieš sąlyga:	Sistemoje turi būti įvesti mėginiai, aktorius turi būti prisijungęs prie sistemos
Sužadinimo sąlyga:	Norima peržiūrėti sistemoje įvestus pacientų mėginius
Po-sąlyga:	Aktoriui išvestas norimų mėginių sąrašas

4. PANAUDOJIMO ATVEJIS:	Papildomos mėginių informacijos paskaičiavimas
Vartotojas/Aktorius:	Seselė
Aprašas:	Mėginiuose kai kurie laukai paskaičiuoji pagal formules ir užpildomi automatiškai
Prieš sąlyga:	Sistemoje turi būti įvestas pacientas, kurio tyrimus norima užpildyti ir taip pat užpildyti laukai kurie reikalingi paskaičiavimai, aktorius turi būti prisijungęs prie sistemos
Sužadinimo sąlyga:	Mėginio duomenys pilnai užpildyti ir reikalingi tik paskaičiavimai
Po-sąlyga:	Sistemoje registruojami paskaičiavimai mėginiams

5. PANAUDOJIMO ATVEJIS:	Mėginių informacijos nuskaitymas iš duomenų failo
Vartotojas/Aktorius:	Seselė, 1PA
Aprašas:	Aktorius pateikia sistemai mėginio informacijos failą, iš kurio nuskaitomi duomenys ir bereikalinga ranka suvesti duomenų
Prieš sąlyga:	Sistemoje turi būti įvestas pacientas, kurio tyrimus norima užpildyti
Sužadinimo sąlyga:	Mėginio duomenys pilnai užpildyti ir reikalingi nuskaityti informaciją iš duomenų failo
Po-sąlyga:	Sistemoje įvedamas pacientui mėginys

6. PANAUDOJIMO ATVEJIS:	Ataskaitų generavimas
Vartotojas/Aktorius:	Gydytojas
Aprašas:	Generuojamos ataskaitos pagal aktoriaus nurodytus poreikius
Prieš sąlyga:	Sistemoje turi būti įvesti paciento mėginiai ir atlikti paskaičiavimai jiems jei trūksta informacijos
Sužadinimo sąlyga:	Gydytojui reikalinga informaciją apie pacientą ar įvestus mėginius
Po-sąlyga:	Gydytojui suformuojama norima ataskaita

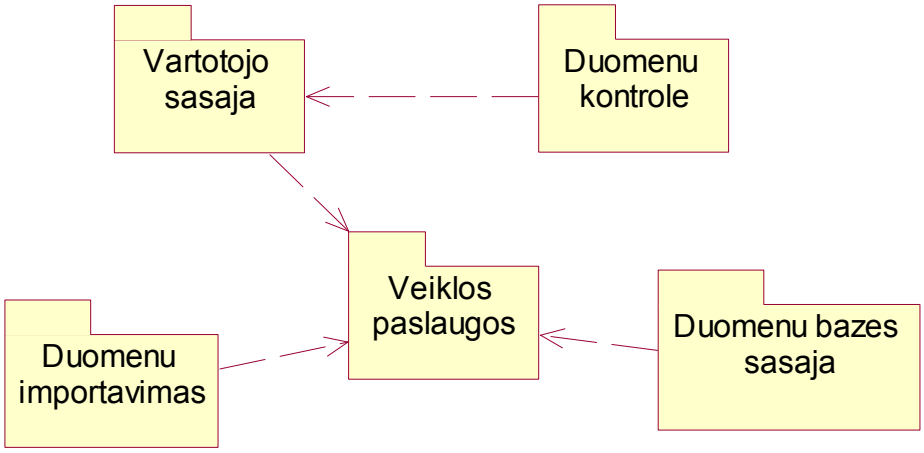
7. PANAUDOJIMO ATVEJIS:	Autorizuotas prisijungimas prie sistemos
Vartotojas/Aktorius:	Gydytojas, seselė
Aprašas:	Norint identifikuoti vartotoją sistemoje, reikalinga vartotojo autorizacija. Formoje įvedami duomenys: prisijungimo vardas ir slaptažodis.
Prieš sąlyga:	Vartotojas turi turėti prisijungimo duomenis ir būti registruotas sistemos vartotojas
Sužadinimo sąlyga:	Gydytojui ar sesutei reikalinga pasiekti duomenis sistemoje ir atlikti kažkokius numatytus veiksmus
Po-sąlyga:	Vartotojas identifikuojamas sistemoje ir suteikiama galimybė atlikti reikiamus veiksmus

8. PANAUDOJIMO ATVEJIS:	Atsijungimas nuo sistemos
Vartotojas/Aktorius:	Gydytojas, seselė
Aprašas:	Vartotojai, norėdami, kad niekas kitas nepasinaudotų jiems suteiktomis privilegijomis, turi atsijungti iš sistemos.
Prieš sąlyga:	Vartotojai turi būti prisijungę prie sistemos
Sužadinimo sąlyga:	Vartotojas nori baigti darbą su sistema
Po-sąlyga:	Vartotojas atjungiamas iš sistemos ir išvedama į ekraną prisijungimo forma

3.5. Architektūra

Projektuojamai sistemai buvo išskirti penki paketai:

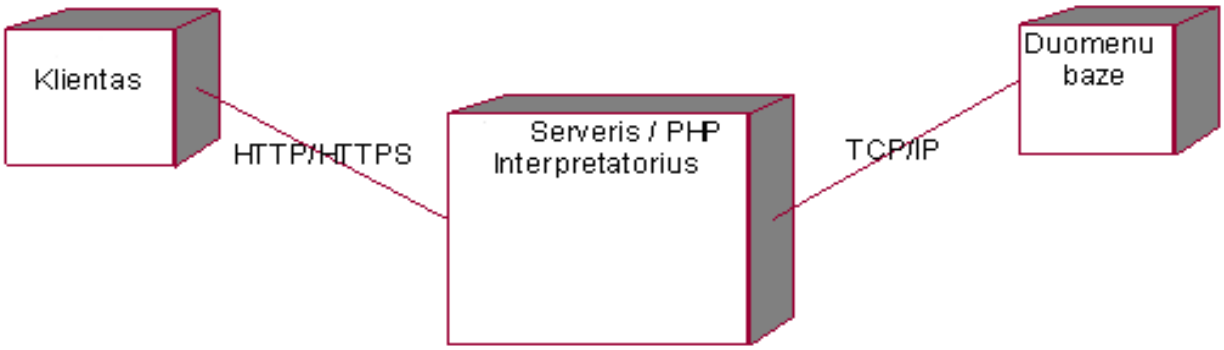
- Vartotojo sąsajos
- Veiklos paslaugų
- Duomenų kontrolės
- Duomenų bazės sąsajos
- Duomenų importavimas



3.2. pav. Sistemos išskaidymas į paketus

3.6. Išdėstymo vaizdas

Programinės įrangos išdėstymo perspektyvos aprašymas apibrėžia įvarius fizinius mazgus daugumos platformų suderinamumui. Taip pat apibrėžia užduočių paskirstymą fiziniams mazgams.



3.3. pav. Sistemos išdėstymo vaizdas

3.6.1. Klientas

Kliento kompiuteryje turi būti įdiegta interneto naršyklė (dažniausiai ji būna įdiegiama kartu su operacine sistema).

Klientams rekomenduojama

Minimalus CPU: 450 MHz

Minimalus RAM kiekis: 128 MB

Minimalus laisvos disko vietos dydis 10 MB

3.6.2. Darbinė stotis ir PHP interpretatorius

HTTP paslauga įdiegiama Apache 1.3 ir aukštesnės versijos ir taip pat PHP interpretatorius 4 ir aukštesnės versijų. Naudojama operacinė sistema Linux.

Žiniatinklio darbinės stoties techninės įrangos reikalavimai:

Minimalus CPU: 800 MHz

Minimalus RAM kiekis: 256 MB

Minimalus disko dydis 2 GB

3.6.3. Duomenų bazė

Duomenų bazė įdiegiama Linux tipo operacinėje sistemoje. Galima tame pačiame Apache ir PHP serveryje. Duomenų bazės valdymo sistema bus naudojama Oracle 9.2.0.

DBVS techninės įrangos reikalavimai:

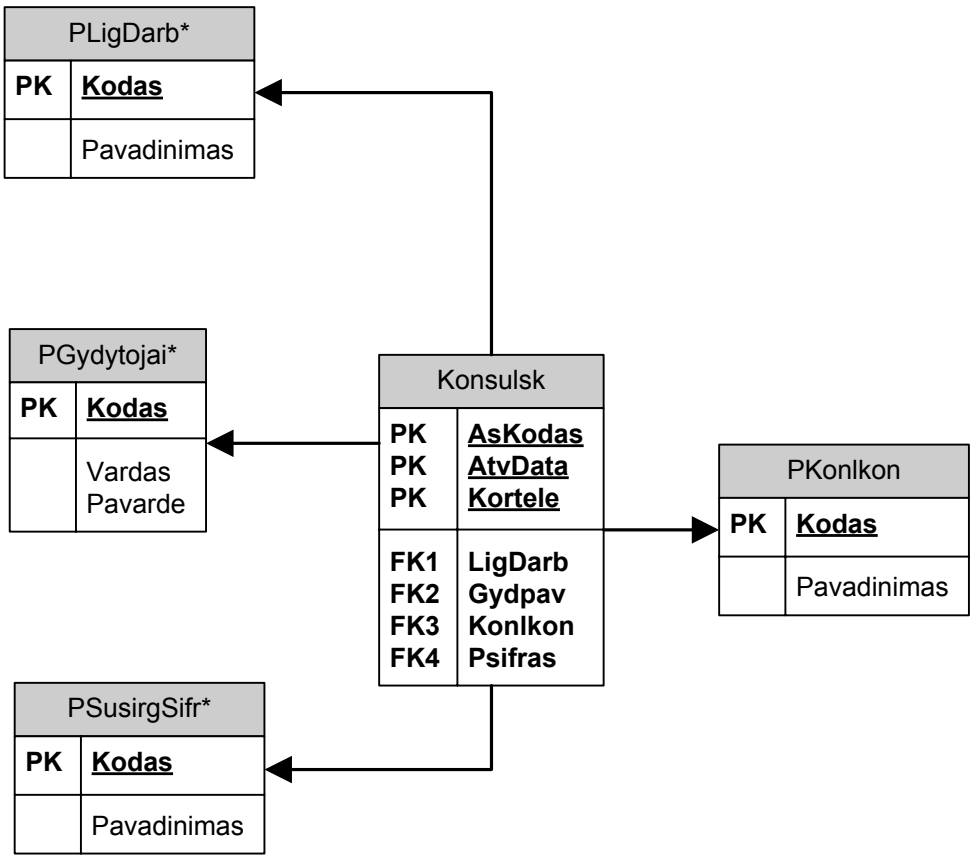
Minimalus CPU: 800 MHz

Minimalus RAM kiekis: 256 MB

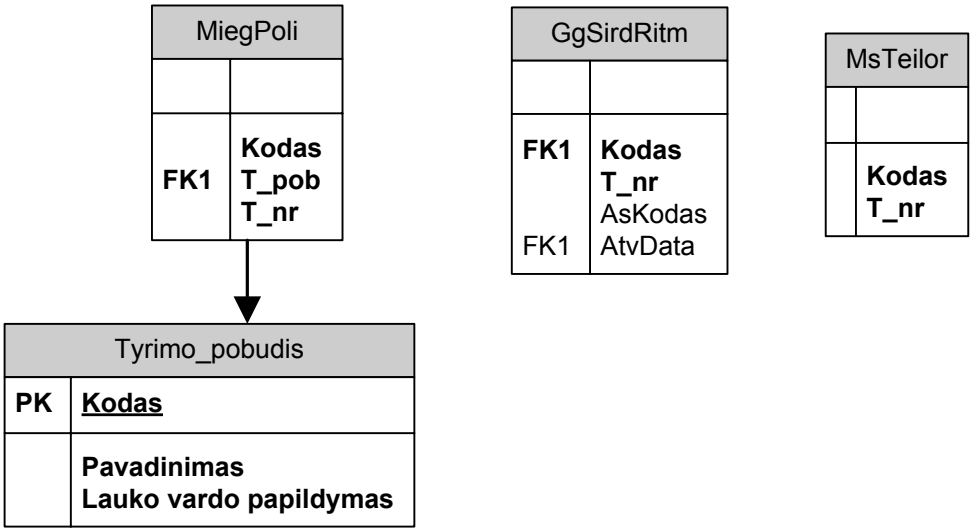
Minimalus Disko dydis laisvos vietos kiekis 200 MB po įdiegimo.

3.7. Duomenų vaizdas

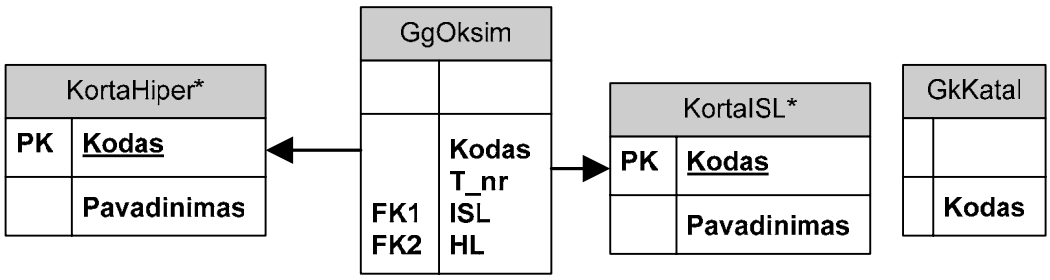
Duomenų bazės modelis pateiktas tik su lentelių raktiniais laikais (kai kurios lentelės pateiktos pilnai). Modelis išskaidytas į susijusias dalis, kiekvienas mėginys rišasi su lentele “Konsulsk”. Lentelės su žvaigždute reiškia, kad ji naudojama iš kitos informacinės sistemos – “kardiovaskulinės reabilitacijos pacientų informacinės sistemos”. Duomenų bazės modelis pateiktas naudojantis Microsoft Viso 2003 projektavimo įrankiu.



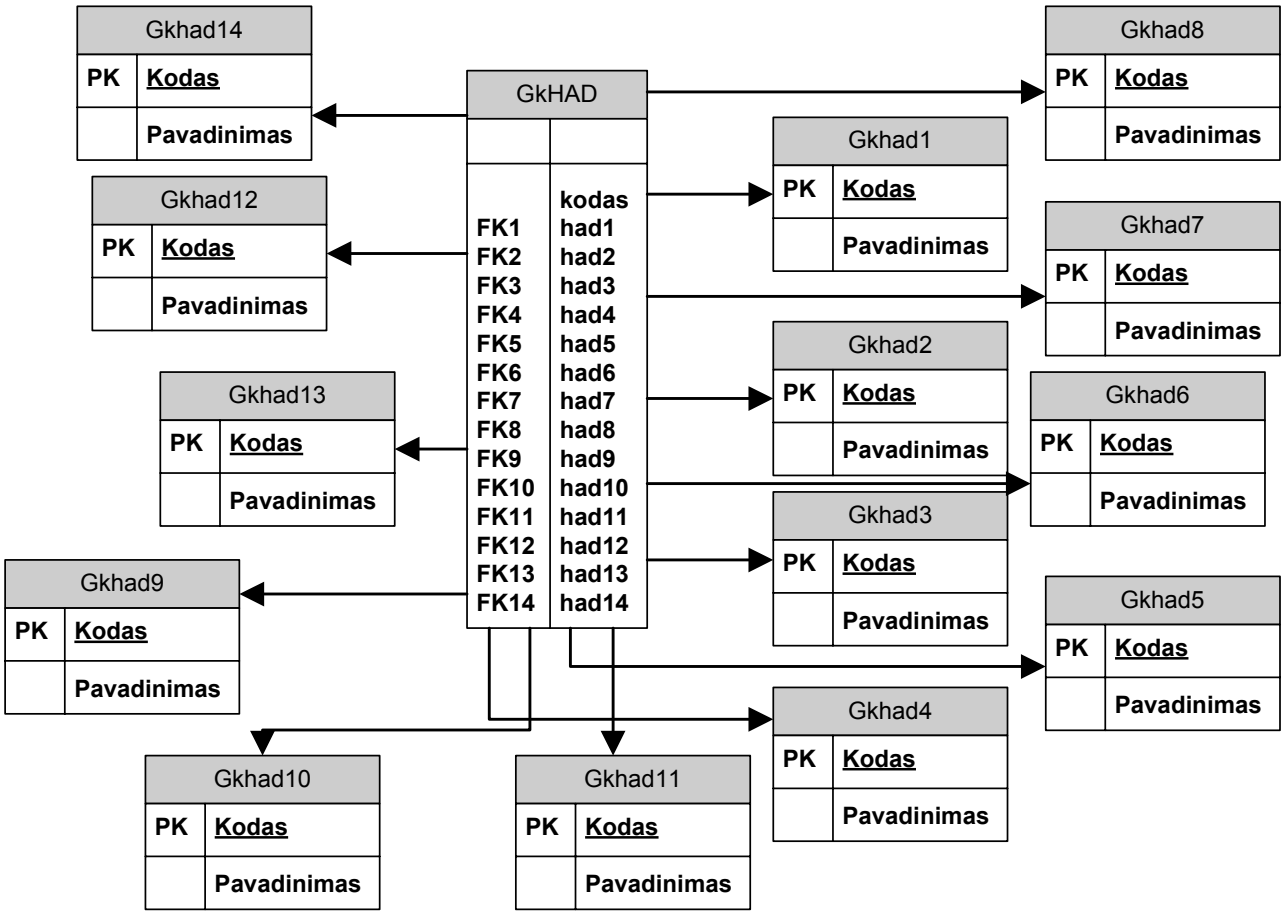
3.4 pav. Konsultuojami pacientai - duomenų bazė schema (fragmentas 1)



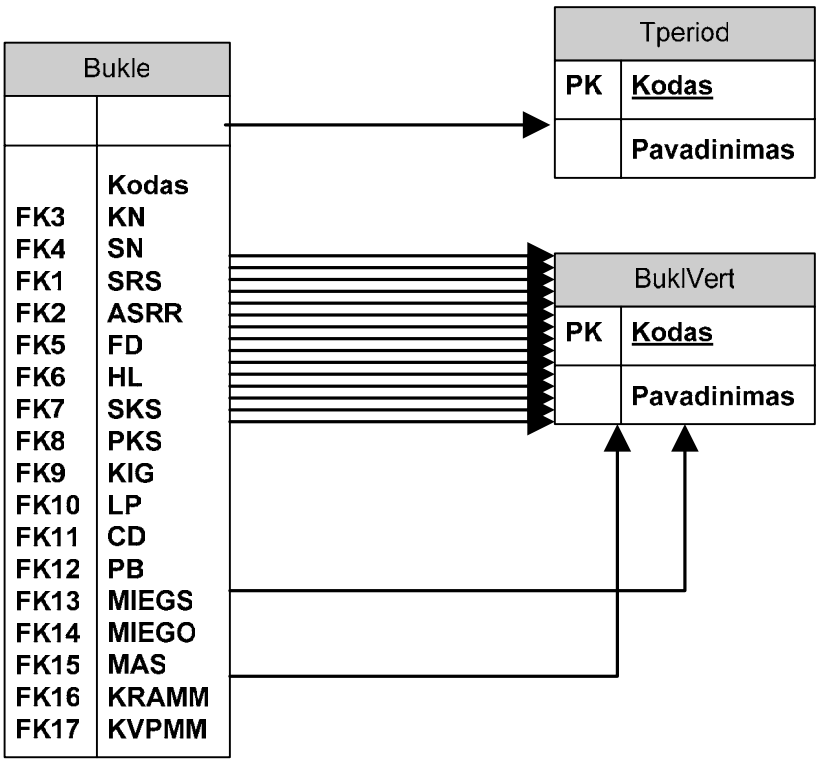
3.5 pav. Mėginių: polisomnografinis tyrimas nakties miego metu, sinusinio širdies ritmo duomenys, Teiloro metodika - duomenų bazės schema (fragmentas 2)



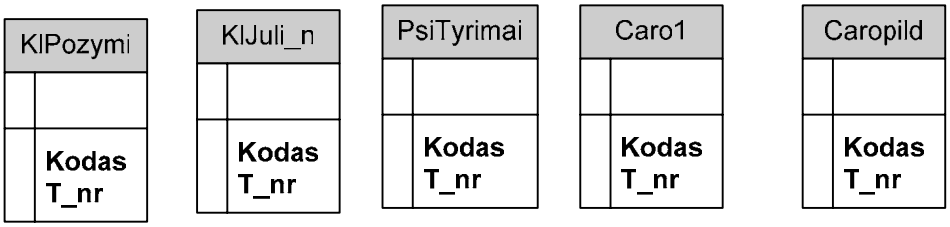
3.6 pav. Mėginių: oksimetrijos duomenys, Katalikybės anketa – duomenų bazės schema (fragmentas 3)



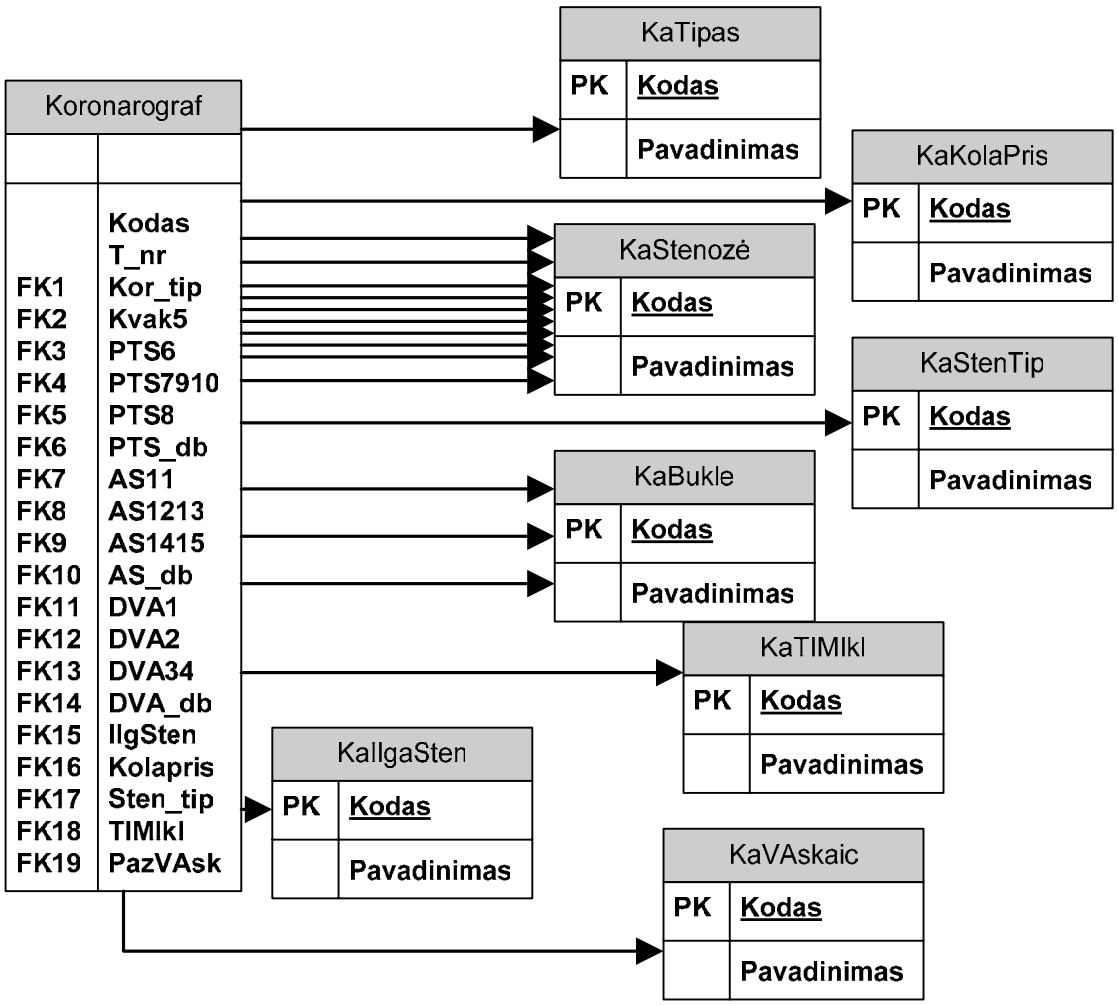
3.7 pav. Mėginio HAD – duomenų bazės schema (fragmentas 4)



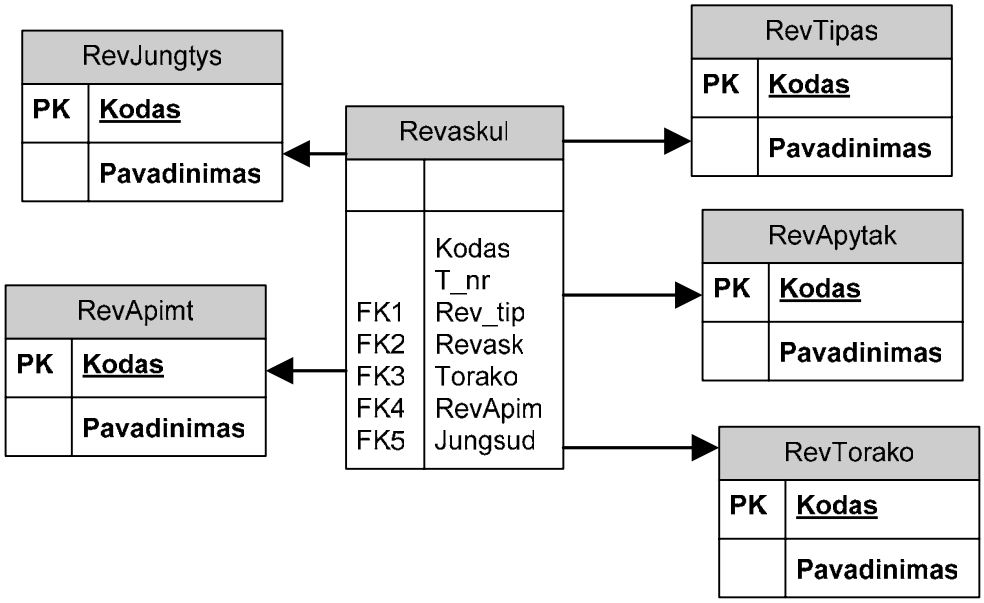
Pav. 3.8. Mėginio Būklės efektyvumo vertinimas – duomenų bazės schema (fragmentas 5)



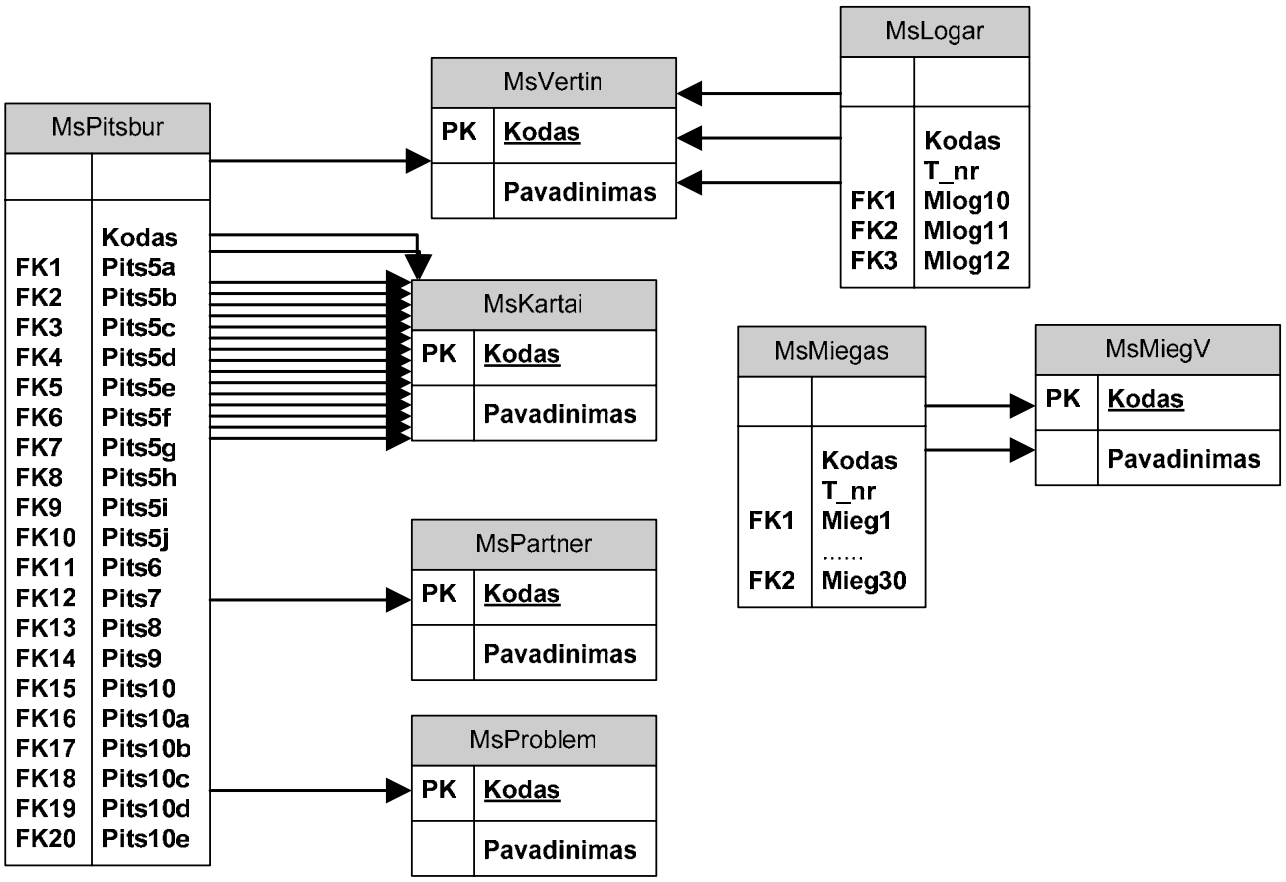
3.9 pav. Mėginių: dujų apykaitos rodikliai, psichologiniai tyrimai, nitroglicerino mėginys – duomenų bazės schema (fragmentas 6)



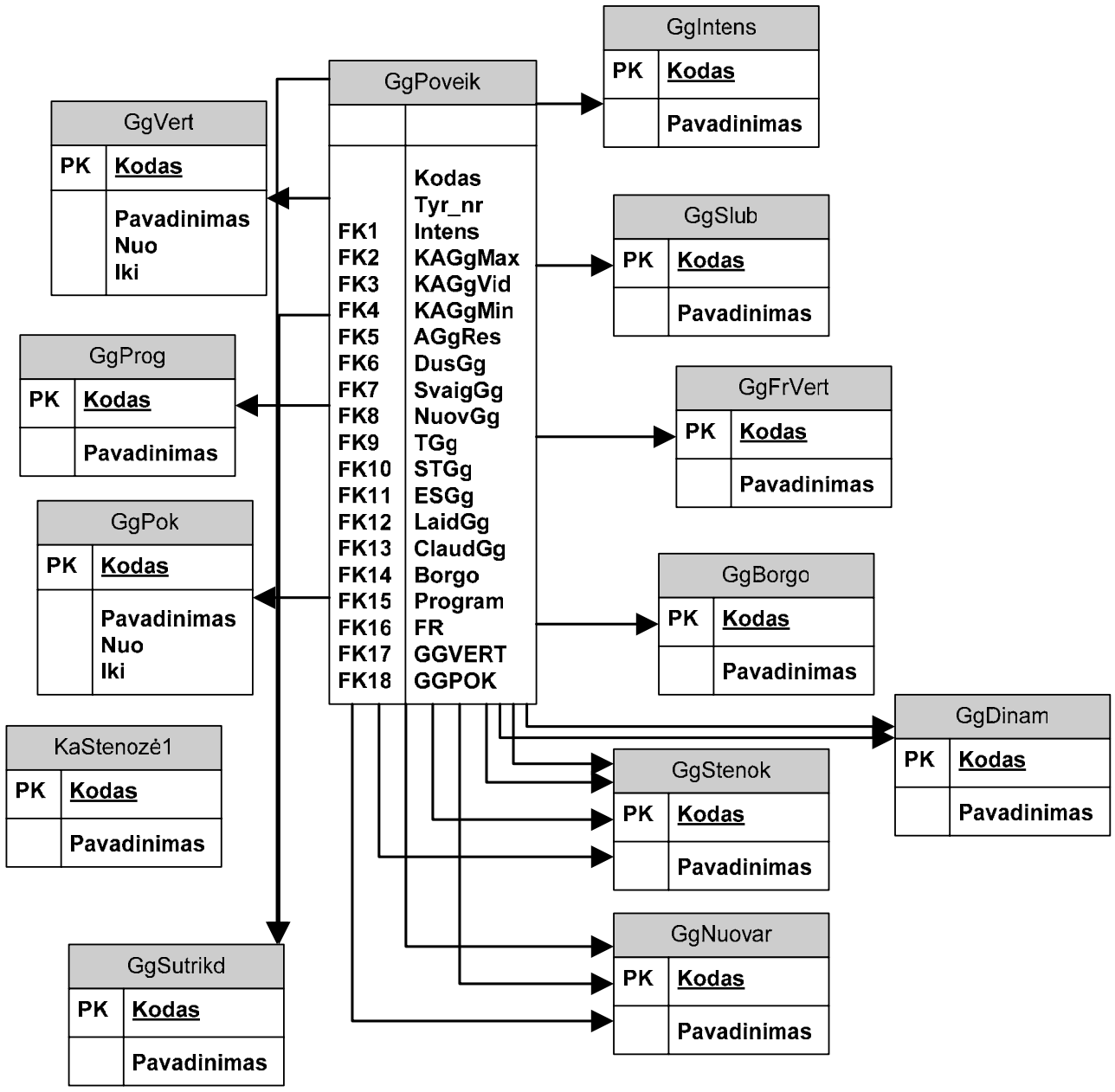
3.10 pav. Mėginio koronarografija – duomenų bazės schema (fragmentas 7)



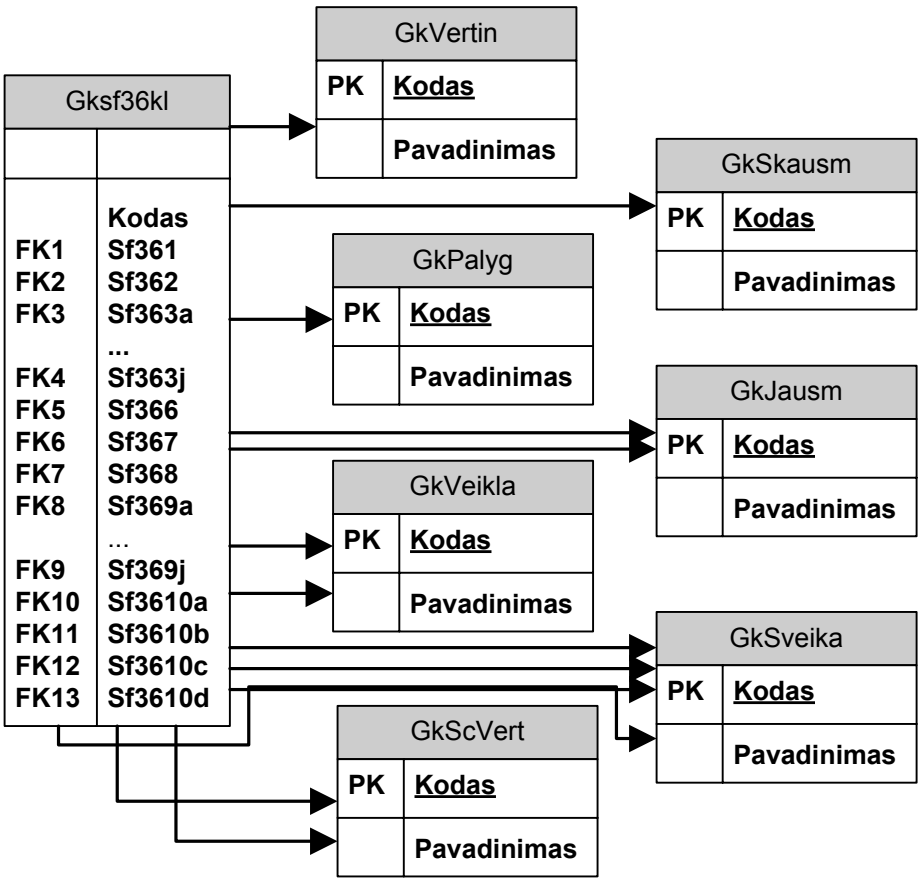
Pav. 3.10. Mėginio revaskulerizacija – duomenų bazės schema (fragmentas 8)



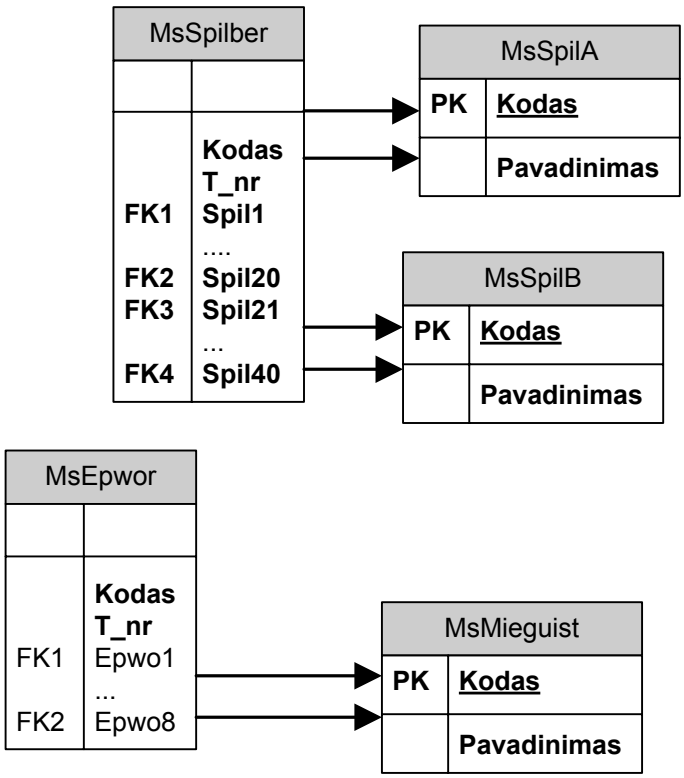
3.11 pav. Mėginių: pitsburgo miego kokybės indeksas, Miego logaritmas, Kaip jūs vertinate savo miegą – duomenų bazės schema (fragmentas 9)



3.12 pav. Mėginio gydomosios gimnastikos poveikio vertinimo duomenys – duomenų bazės schema (fragmentas 10)



3.13 pav. Mėginio SF-36 klausimynas – duomenų bazės schema (fragmentas 11)



3.14 pav. Mėginių spilbergerio skalė, Epworth mieguistumo skalė – duomenų bazės schema (fragmentas 12)

4. Tyrimo dalis

Šio tyrimo metu analizuojama sukurtos programinės įrangos per magistratūros kursą užklausų generavimo privalumai, lyginant su išsaugotų procedūrų panaudojimo PL/SQL kalba. Tyrimas susideda iš dviejų etapų: kaip pagreitina pasirinktas metodas programinės įrangos sukūrimą ir kaip pagreitinamas programinės įrangos priežiūros darbai. Apie sukurtos informacinės sistemos "Psichofiziologinės reabilitacijos pacientų informacinė sistema" architektūrą galite paskaityti 3 skyriuje. Palyginimas skirtas ne programinės įrangos našumo testavimui, kaip pasirinktas metodas įtakoja sukurtos informacinės sistemos darbo našumą.

4.1. PĮ sukūrimas

Turint programinės įrangos reikalavimų specifikaciją, galima nustatyti kiek reikia parašyti komponentų, norint atlikti reikalavimuose numatytus veiksmus. Šiame konkrečiame projekte, buvo pateikti reikalavimai, kurie turi apdoroti 21 mėginio duomenis. Šiame skyriuje aprašysime kiek reikia atlikti veiksmų užklausų generavimo ir PL/SQL kalba. Išskaidysime pagal atliekamus veiksmus: įterpimą, atnaujinimą, šalinimą ir išrinkimą.

4.1.1. Įterpimas

Pasirinkus metodą naudoti išsaugotas procedūras PL/SQL kalba, reikia kiekvienam mėginiui sukurti po išsaugotą procedūrą. Iš viso mėginių - 21 todėl reikės sukurti 21 išsaugotą procedūrą. Kadangi naudojama PHP programavimo kalba sąsajai su duomenų baze, tai papildomai reikalinga kiekvienam mėginiui sukurti po metodą, kuris pateiktu duomenis išsaugotai procedūrai. Kaip pavyzdį pateiksime 9.1 priede metodą, iš kurio matyti ko reikia, norint pasinaudoti išsaugota procedūra. Iš jos matyti, kad reikia 2 kartus pakartoti lentelės atributų laukus rašant šį metodą. Dar pažvelgus į išsaugotos procedūrą 9.2 priede matome, kad ten reikia išvardinti parametrus (visi įrašomi laukai), išvardinti lentelės laukų pavadinimus ir jiems priskirti kintamuosius su reikšmėmis. Iš viso tenka išvardinti laukus 5 kartus. Dėl to atsiranda didelė tikimybė, kad programuotojas gali įvelti klaidą, o jos suradimas ir ištaisymas bus ilgas, nes klaidos įvėlimas galimas keliose vietose.

Pasirinkus metodą, kai generuojamos užklausos, reikia sukurti vieną metodą, kuris gali atlikti visiems mėginiams (ir visoms kitoms užklausoms, kurios susijusios su įrašymu) įrašymą į

duomenų bazę. Kiekvienas mėginys objektas paveldėjęs šį metodą perduos savo atributus per parametrą, kuris yra masyvas. Šio metodo realizacijos pavyzdį galite peržiūrėti 9.3 priede. Panaudojus šį metodą, mums reikalinga paduoti parametrus tik vieną kartą, todėl klaidos įvėlimas sumažėja, o programinės įrangos sukūrimo laikas jau nebe taip priklauso nuo mėginių skaičiaus ir mėginį apibūdinančių atributų skaičiaus.

4.1.2. Atnaujinimas

Duomenų atnaujinimui pasirinkus išsaugotas procedūras, kaip ir duomenų įterpimui, reikalinga rašyti atskiras išsaugotas procedūras ir metodus šioms procedūroms iškviesti ir paduoti joms duomenis. Skirtumas tarp įterpimo užklausų ir atnaujinimo toks, kad reikia pateikti įrašą identifikuojančių laukų reikšmes. Tai taip pat kiekvienam mėginiui reikia skirtingos procedūros, kurių reikalinga tiek, kiek egzistuoja mėginių. Iš pateiktų pavyzdžių 9.4 ir 9.5 prieduose galite matyti, kad šitas metodas taip pat reikalauja daug kartų pakartoti mėginio atributus, kas gali sukelti klaidų įvėlimą.

Naudojant užklausų generavimą užtenka parašyti vieną metodą, kurį paveldės kiekvieno mėginio objektas ir pasinaudos mėginio duomenims pakeisti. Taip vėl sumažinama rizika įvelti klaidų sumaišant mėginio atributų vardus ar jų reikšmių priskirimus. Metodo realizacija pateikiama 9.6 priede.

4.1.3. Šalinimas

Šalinimo veiksmams atlikti standartiniu metodu nesudaro labai didelių problemų, nes šalinamam mėginiui dažniausiai užtenka nurodyti keletą jį idenfikuojančių parametrų. Priešingai nei mėginio įterpimui ar atnaujinimui čia suklysti šansų sumažėja. Bet vis tiek naudojant išsaugotas procedūras PL/SQL kalboje tenka kurti kiekvienam mėginiui atskirą išsaugotą procedūrą ir taip pat metodą PHP kalboje, atsakingą už šios procedūros iškvietimą. Todėl atsiranda vėl žymiai daugiau darbo nei naudojant užklausų generavimą. Panaudojus užklausų generavimą sukuriamas vienas metodas, kuris užtikrina visų mėginių ištrynimą, pagal jam pateiktus parametrus. Šio metodo realizaciją galite matyti 9.7. priede.

4.1.4. Išrinkimas

Nagrinėjamos sistemos architektūros duomenų bazės modelis leidžia gan efektyviai išnaudoti ir užklausų generavimą duomenų išrinkimui. Kadangi buvo pasirinktas būdas

duomenims saugoti sutraukiant vieno mėginio duomenis į vieną eilutę [14]. Taip bus galima išrinkinėti patogiai duomenis iš vienos lentelės, išvengiant jungtinių lentelių kas pasunkintu užklausų generavimo panaudojimą.

Išsaugotų procedūrų atveju reikalinga sukurti kiekvienam mėginiui išrinkti pagal raktą atskirą išsaugotą procedūrą ir taip pat papildomai metodą, kuris perduotų gautus duomenis į veiklos logiką.

Naudojant užklausų generavimą užtenka vieno metodo, kuris generuoja užklausas išrinkimui pagal raktą, kuris pateikiamas per parametrą. Taip gaunamas vieno mėginio atributų reikšmės ir nereikia kiekvienam mėginiui kurti atskiro metodo. Šios užklausos generavimo metodą galite peržiūrėti 9.8. priede.

4.2. PĮ priežiūra

Dažnai tenka sukurtą programinę įrangą atnaujinti ar taisyti klaidas. Šiame darbe nagrinėjamos sukurtos informacinės sistemos pagrindinis elementas mėginiai, kurie laikui bėgant gali keistis ir jų atsirasti daugiau. Šiame skyriuje paanalizuosime kokių pastangų reikia įdėti, kad būtų atlikti atsiradę nauji pakeitimai. Kiekvienu metodu bus trumpai aptariamos pagrindinės užklausos, kurias teks modifikuoti ir ką tiksliai reiks daryti. Analizuosime du atvejus kada atsiras papildomai naujas mėginys ir kada egzistuojantis mėginys papildomas naujais atributais.

Kiekvienu metodu bus reikalinga atnaujinti vartotojo sąsają, bei duomenų bazės struktūra. Šiame skyriuje apie šių pakeitimų įtaka nekalbėsime, nes ir vienu ir kitu metodu pastangos atlikti yra lygiavertės.

4.2.1. Įterpimas

Atsiradus naujiems mėginiams informacinėje sistemoje, naudojant metodą kai kuriamos PL/SQL užklausos, reikės sukurti kiekvienam naujam mėginiui po išsaugotą procedūrą, taip pat metodus PHP kalboje sąsajai su sukurtomis procedūromis. Atliekami veiksmai tokie kaip ir kuriant programinę įrangą, kurie aprašyti skyriuje 4.1.1. *Įterpimas*.

Naudojant užklausų generavimą, nereikės nieko papildomai daryti kas susiję su naujo mėginio įrašymu. Aišku bus reikalinga sukurti tam mėginiui skirta objektą, kuris paveldės metodą duomenų įrašymui į duomenų bazę. Naudojant išsaugotas procedūras šitoks objektas taip pat reikalingas.

Esant tik naujų mėginio atributų atsiradimui, naudojant išsaugotas procedūras, reikia peržiūrėti ir atnaujinti išsaugotą procedūrą, susijusią su mėginių apdorojimu ir metodą, atsakingą su šios išsaugotos procedūros iškvietimu ir duomenų pateikimui duomenų bazės valdymo sistemai.

Panaudojus metodą generuojant užklausas keisti papildomai nieko nereikia, kad nauji duomenys būtų perduoti duomenų bazės valdymo sistemai.

4.2.2. Atnaujinimas

Jei atsiranda nauji mėginiai ar pakeitimai egzistuojančiuose mėginiuose, papildomai atsiranda atributų. Tada neišvengiamai reikia keisti visus veiksmus, susijusius su keičiamu mėginiu ar naujai atsiradusiu. Naudojant išsaugotas procedūras reikia peržvelgti senas išsaugotas procedūras, kurios susijusios su mėginių apdorojimu, o jei mėginys naujas, tai atliekami tokie patys veiksmai, kaip buvo aptarta 4.1.2. *Atnaujinimas* skyriuje. Norint pakeisti egzistuojantį mėginį, reikia atnaujinti išsaugotą procedūrą, metodą, atsakingą už šios procedūros iškvietimą ir duomenų pateikimą.

Naudojant užklausų generavimą, problemos kaip ir nėra, nes generuojama užklausa pagal pateiktus parametrus, todėl taisyti nieko nereikia.

4.2.3. Šalinimas

Šalinimo operacijos įtrakta pakeitimai tik tada, kada keičiasi jį identifikuojantys laukai. Todėl naudojant išsaugotas procedūras esant pasikeitimams esančiuose mėginiuose nesudarytų papildomų veiksmų. Atsiradus naujiems mėginiams, reikės kurti papildomai išsaugotas procedūras, kurios apdorotų šių naujų mėginių trynimą.

Sugeneruotu užklausų metode nieko papildomai kurti nereikėtu jei pasikeistu mėginį identifikuojantys laukai ar atsiradus naujam mėginiui.

4.2.4. Išrinkimas

Išrinkimo atveju keičiantis mėginių atributams reikėtų peržiūrėti išsaugotas procedūras ir metodus, skirtus jiems iškviesti. Atsiradus naujiems mėginiams, parašyti naujas išsaugotas procedūras ir metodus iškviečiančius jas.

Naudojant sugeneruotas užklausas to bus išvengiama.

5. Eksperimentinė dalis

Šioje dalyje pateikiama statistiniai palyginimai, kurie buvo atlikti su užklausų generavimu ir naudojant pl/SLQ kalbą. Testai buvo atliekami su magistratūros studijų metu sukurta programine įranga, bei papildomais sukurtais komponentais, kurie skirti palyginti kaip įtakoja pasirinktas metodas sistemos našumui. Eksperimentas suskirstytas į tris dalis pagal skirtingas vykdomas užklausas. Trynimo užklausos nebuvo analizuojamos, nes mėginiai trinami tik esant vartotojų įvestoms klaidoms, todėl šio tipo užklausos sukurtoje informacinėje sistemoje sudaro mažą dalį. Testams buvo paimta 3 dydžių lentelės, kurių atributų skaičius: 8, 102, 203. Taip pat buvo testuojama sistemos užduočių atlikimas su skirtingu lygiagrečių vartotojų skaičiumi.

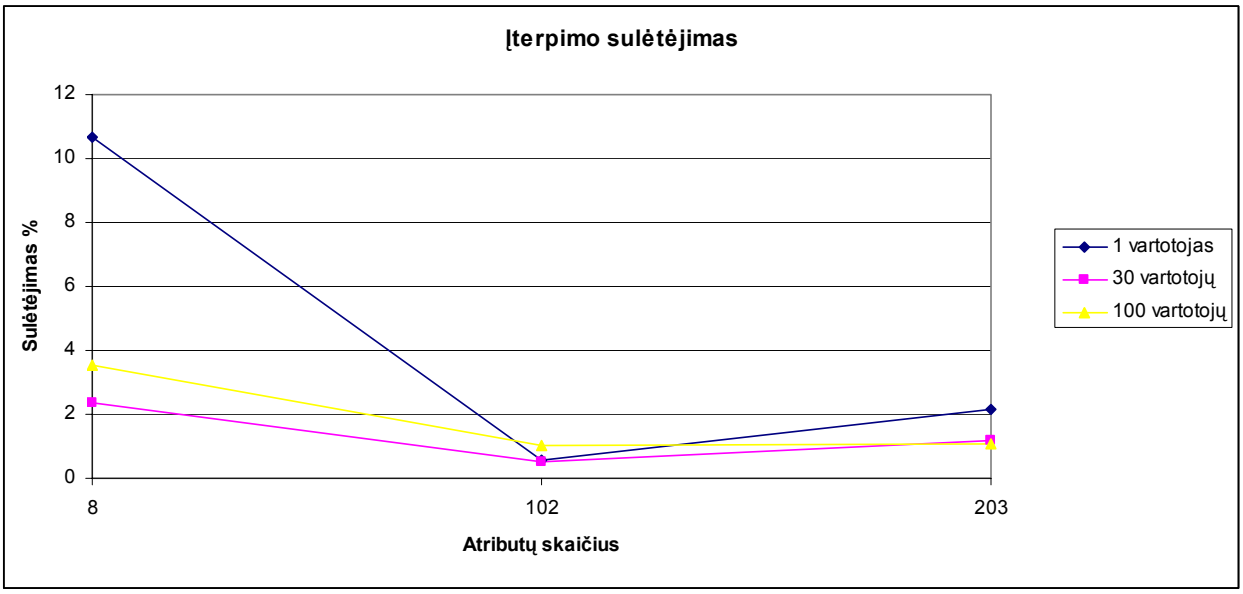
5.1. Įterpimas

Lentelė 5.1. *Duomenų įterpimo rezultatų palyginimo lentelė*

	1 vartotojas			30 vartotojų			100 vartotojų		
Lentelės atributų skaičius	8	102	203	8	102	203	8	102	203
PL/SQL (ms)	57,24	60,72	70,55	26,66	26,62	28,50	28.75	27,10	28,99
Užklausų generavimas (ms)	64,06	61,05	72,09	27,30	26,76	28,84	29,81	27,38	29,31
Sulėtėjimas %	10,65	0,54	2,14	2,34	0,52	1,18	3,56	1,02	1,09

Kadangi testuojama buvo daugiaprocesorinėje sistemoje, tai rezultatai su daugiau lygiagrečių vartotojų buvo atliekami greičiau, t.y. sistema išnaudojama efektyviau.

Pateikiamas sulėtėjimo grafikas kai naudojame sugeneruotas užklausas.



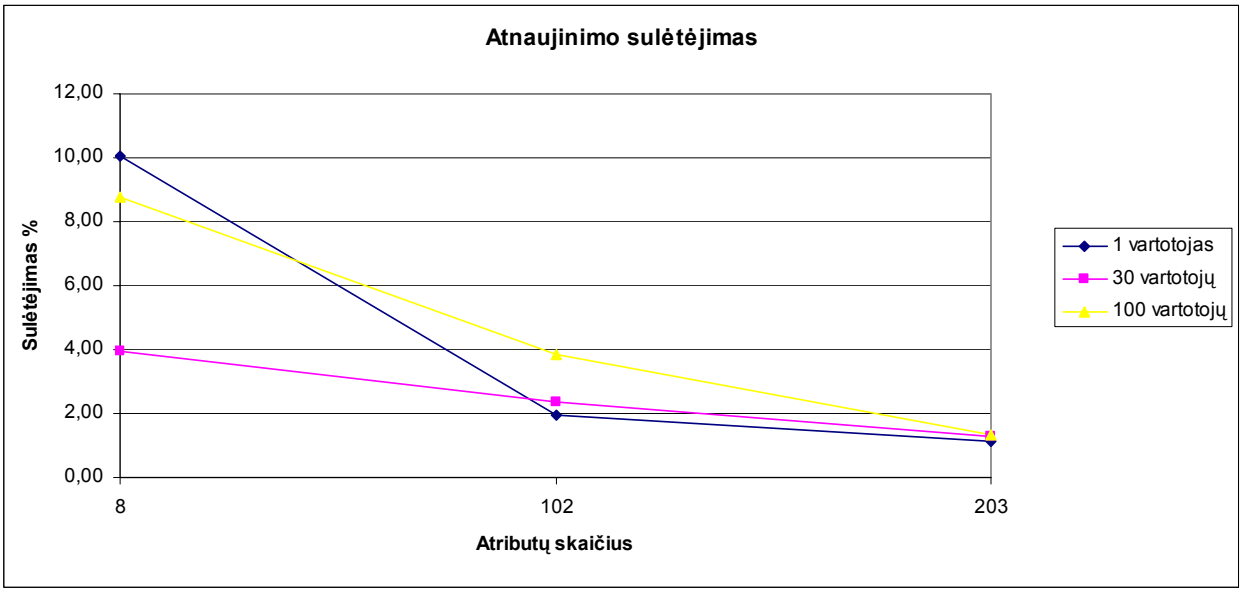
5.1 pav. Įterpimo sulėtėjimas

5.2. Atnaujinimas

Lentelė 5.2. *Duomenų atnaujinimo rezultatų palyginimo lentelė*

	1 vartotojas			30 vartotojų			100 vartotojų		
Lentelės atributų skaičius	8	102	203	8	102	203	8	102	203
PL/SQL (ms)	49,32	87,01	170,23	27,80	46,52	99,89	26,94	43,89	96,03
Užklausų generavimas (ms)	54,84	88,73	172,18	28,95	47,65	101,21	29,53	45,64	97,31
Sulėtėjimas %	10,07	1,94	1,13	3,97	2,37	1,30	8,77	3,83	1,32

Pateikiamas sulėtėjimo grafikas kai naudojame sugeneruotas užklausas.



5.2 pav. Atnaujinimo sulėtėjimas

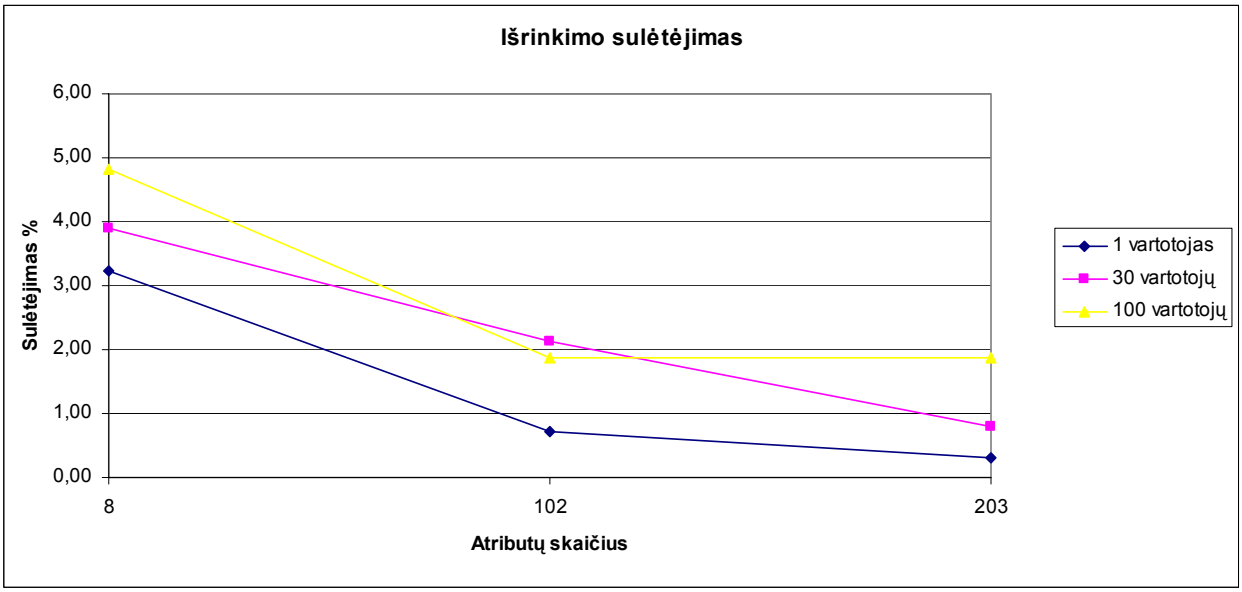
5.3. Išrinkimas

Išrinkimo metu testuota išrenkamo vieno paciento mėginio tyrimus ir jų sugražinimą į logikos sluoksnį.

Lentelė 5.3. *Duomenų išrinkimo rezultatų palyginimo lentelė*

	1 vartotojas			30 vartotojų			100 vartotojų		
Lentelės atributų skaičius	8	102	203	8	102	203	8	102	203
PL/SQL (ms)	52,97	91,12	198,89	27,71	44,01	93,99	28,01	44,21	92,81
Užklausų generavimas (ms)	54,74	91,78	199,52	28,83	44,97	94,74	29,43	45,05	94,57
Sulėtėjimas %	3,23	0,72	0,32	3,88	2,13	0,79	4,83	1,86	1,86

Pateikiamas sulėtėjimo grafikas kai naudojame sugeneruotas užklausas.



5.3 pav. Išrinkimo sulėtėjimas

Iš visų atliktų rezultatų matome, kad šis metodas nesukelia žymaus sistemos sulėtėjimo, lyginant su išsaugotų procedūrų naudojimu. Žymus sulėtėjimas pastebimas atliekant mažai mėginių turinčius veiksmus. Kadangi projekte dauguma mėginių turi apie 70 atributų, bendras sistemos sulėtėjimas yra nežymus.

6. Išvados

1. Pasirinkta kliento-serverio architektūra, palengvina sukurtos informacinės sistemos “Psichofiziologinės reabilitacijos pacientų informacinė sistema” priežiūrą.
2. Sistemos duomenų sąsajos sukūrimo laikas nepriklauso nuo mėginių skaičiaus, naudojant užklausų generavimo metodą.
3. Pasirinktos technologijos leido įgyvendinti užklausų generavimo metodą, kuris padidina darbo našumą ir palengvina sistemos priežiūrą.
4. Užklausų generavimo metodas mažai įtakoja informacinės sistemos našumo sumažėjimą.

7. Literatūra

[1] Kolp M. *What is information System Analysis* [žiūrėta 2005-03-10]. Prieiga per Internetą: http://www.cs.toronto.edu/~mkolp/1-Intro3831.pdf.

[2] Bhattacharaya S., Ravi S. Behara, David E. Gundersen *Business risk perspectives on information systems outsourcing*. International Journal of Accounting Information Systems 4, 75-93, 2003.

[3] Wiryana M. *Information System Development: An Interdiscipline Approach*. 1998.

[4] Mahaney R. C., Lederer A. *Information systems project management: an agency theory interpretation*. Journal of Systems and Software 68(1):1-9, 2003.

[5] PHP: Hypertext Preporcessor [žiūrėta 2005-05-8]. Prieinga per Internetą: http://www.php.net/

[6] Štuikys V., Montvilas M., Damaševicius R., Ziberkas G., Limanauskienė V., Bartkevičius M., Pakašius A. *Analysis of Application of Multi-Languages Design Paradigm for Creating user-orientated web pages*. Information Technology & Control, 4(29), 57-66, 2003.

[7] Oracle® PL/SQL™ by Example, 3 rd. ed. Benjamin Rosenzweig, Elena Silvestrova, 2003 [žiūrėta 2005-05-02].

[8] Refsnes Data *Introduction to SQL* [žiūrėta: 2005-05-04]. Prieiga per Internet: http://www.w3schools.com/sql/sql_intro.asp

[9] Damaševičius R. *Programų kūrimas ir tobulinimas* [žiūrėta 2005-02-5]. Prieiga per Internetą: http://www.soften.ktu.lt/~damarobe/T120M013/Ld1/Ld1.htm

[10] Barkley J., Beznosov K., Uppal J. *Supporting Relationships in Access Control Using Role Based Access Control*. In Proceedings of the Fourth ACM, 55-65, 1999.

[11] A Research and Development project of Science Applications International Corporation and the University of California, San Diego *Patient Centered Access to Secure Systems Online* [žiūrėta 2005-03-28]. Prieiga per Internetą: http://medicine.ucsd.edu/pcasso/

[12] Haak M., Wolff A.C., Brandner R., Drings P., Wannenmacher M., Wetter Th. *Data security and protection in cross-institutional electronic patient records*. International Journal of Medical Informatics, Volume 70, Issue 2-3, Pages 117-130, July 2003.

[13] Coarfa C., Druschel P., Wallach Dan S., *Performance Analysis of TLS Web Servers*. Network and Distributed Systems Security Symposium '02, San Diego, California, February 2002.

[14] Šimkevičiutė G. *Daugiaatribučių duomenų struktūrų sukūrimas ir jų alternatyvų įvertinimas,* Kaunas, 2004

8. Terminų ir santrumpų žodynas

8.1. Terminai

Mėginys – tam tikram psichofiziologijos reabilitacijos pacientui atlikto vieno iš tyrimo užfiksuoti duomenys.

SQL **-** SQL yra standartas, struktūrizuota užklausų kalba (Structured Query Language), kuri skirta valdyti duomenims duomenų bazės valdymo sistemose [8]. Šis standartas labai populiarus ir naudojamas kiekvienoje duomenų bazės valdymo sistemoje, nors skirtingose DBVS atsiranda skirtingų SQL versijų. Bet pagrindinės komandos išlieka visose sistemose vienodos.

RSA - kodavimo metodo ir tokių kodavimo raktų generavimas, kad su vienu – viešu raktu užkodavus, atkoduoti galima būtų tik su kitu – slaptu raktu.

8.2. Santrumpos

PHP – **(***Hypertext preprocesor*) preprocesorius, programavimo kalba.

DBVS – duomenų bazių valdymo sistema.

PĮ – programinė įranga.

RAM – (*Random Access Memory*) operatyvioji atmintinė.

HTTP – (*HyperText Transmission Protocol*) protokolas, skirtas perduoti duomenis internete.

HTTPS – (*HyperText Transmission Protocol, Secure*) apsaugotas tinklo protokolas duomenims internete perduoti.

VPN – (*virtual private network*) virtualus privatus tinklas.

SSL – (*Secure socket layers*) saugaus protokolo sluoksnis.

DES – (*Data Encryption Standard*) užšifravimo algoritmas.

RSA – užšifravimo algoritmas.

9. Priedai

9.1 Įterpimas naudojant PL/SQL

```
function Insert($array)
{
   $this->db->Query_for_bind("begin INSERTSAMPLE(:code, :dates, :field1,
:field2); end;");
   $this->db->Bind(":code", $array[‘code’]);
   $this->db->Bind(":dates", $array[‘dates‘]);
   $this->db->Bind(":field1", $array[‘field1’]);
   $this->db->Bind(":field2", $array[‘field2’]);
   if ( $this->db->Execute())
      return true;
   else return false;
}
```

9.2. Išsaugota procedūra skirta įterpti mėginį

```
CREATE OR REPLACE PROCEDURE INSERTSAMPLE (
      code in CHAR,
      dates in DATE,
      field1 in NUMBER,
      field2 in NUMBER
)
AS
BEGIN
      INSERT INTO SAMPLE (CODE, DATES, FIELD1, FIELD2)
      VALUES (code, dates, field1, field2);
END;
```

9.3. Įterpimo užklausos generavimas

```
function InsertSample($array)
{
      $query = "INSERT INTO {$this->table} ( ";
      $qs = "";
      foreach ($array as $key => $value)
      {
            if ( substr($key, 0, 2) == 'i_')
            {
                  $query .= substr($key, 2).", ";
                  if ( is_float($value))
                        $qs .= " ".$value.", ";
                  elseif ( is_int($value))
                        $qs .= " ".$value.", ";
                  else $qs .= "'".$value."', ";
             }
       }
       $query = substr($query, 0, -2);
       $qs = substr($qs, 0, -2);
       $query = $query." ) VALUES ( ".$qs." )";
       if ( !$this->db->Query($query) )
           return false;
       else return true;
```

```
}
```

9.4. Atnaujinimas naudojant PL/SQL

```
function Update($array)
{
   $this->db->Query_for_bind("begin UPDATESAMPLE(:code, :dates, :field1,
:field2); end;");
   $this->db->Bind(":code", $array[‘code’]);
   $this->db->Bind(":dates", $array[‘dates‘]);
   $this->db->Bind(":field1", $array[‘field1’]);
   $this->db->Bind(":field2", $array[‘field2’]);
   if ( $this->db->Execute())
      return true;
   else return false;
}
```

9.5. Išsaugota procedūra skirta atnaujinti mėginį

```
CREATE OR REPLACE PROCEDURE UPDATESAMPLE (
      code in CHAR,
      dates in DATE,
      field1 in NUMBER,
      field2 in NUMBER
)
AS
BEGIN
      UPDATE SAMPLE SET DATES = dates, FIELD1 = field1, FIELD2 = field2
      WHERE CODE = code;
END;
```

9.6. Atnaujinimo užklausos generavimas

```
function UpdateSample($array)
{
  $qs = "";
  $query = "UPDATE {$this->table} SET ";
  foreach ($array as $key => $value)
  {
     if ( substr($key, 0, 2) == 'u_')
     {
        $qs .=  substr($key, 2)." = '".$value."' and ";
     }
     if ( substr($key, 0, 2) == 'i_')
     {
         if ( is_float($value))
               $query .= substr($key, 2)." = ".$value.", ";
         else $query .= substr($key, 2)." = '".$value."', ";
     }
   }
   $query = substr($query, 0, -2);
   $qs = substr($qs, 0, -4);
   $query .=  "WHERE $qs";
   if ( !$this->db->Query($query) )
     return false;
```

```
   else return true;
}
```

9.7. Šalinimo užklausos generavimas

```
function DeleteSample($array)
{
      $qs = "";
      foreach ($array as $key => $value)
      {
         if ( substr($key, 0, 2) == 'u_')
         {
             $qs .=  substr($key, 2)." = '".$value."' and ";
         }
      }
      $qs = substr($qs, 0, -4);
      $query = "DELETE FROM {$this->table} WHERE $qs";
      if ( !$this->db->Query($query) )
          return false;
      else return true;
}
```

9.8. Išrinkimo užklausos generavimas

```
function getSample($array)
{
      $query = "SELECT * FROM {$this->table} WHERE “;
      foreach ($array as $key => $value)
      {
         if ( substr($key, 0, 2) == 'u_')
         {
             $query .=  substr($key, 2)." = '".$value."' and ";
         }
      }
      $query = substr($query, 0, -4);
      if (!$this->db->Query($query))
            return false;
      else  return $this->db->GetOne();
}
```