KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Mindaugas Kučinskas

EFEKTYVAUS MANIPULIAVIMO DUOMENIMIS INFORMACINĖSE MEDICINOS SISTEMOSE TYRIMAS

Magistro darbas

Vadovas
doc. dr. R. Butleris

KAUNAS, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

EFEKTYVAUS MANIPULIAVIMO DUOMENIMIS INFORMACINĖSE MEDICINOS SISTEMOSE TYRIMAS

Programų inžinerijos magistro baigiamasis darbas

Kalbos konsultantė
J. Mikelionienė
2006-05-23

Vadovas
doc. dr. R. Butleris
2006-05-23

Recenzentas
doc. dr. A. Lenkevičius
2006-05-23

Atliko
IFM-0/2 gr. stud.
M. Kučinskas
2006-05-23

KAUNAS, 2006

SUMMARY

This work is aimed to research the most effective way of using ADO.NET data access components in medicine system development process.

This document we start from evaluating current situation of e-medicine in Lithuania and world wide. Consider the problems, which originate doing a lot of “paper work”, introduce with EMR (Electronic Medical Record) systems, list the advantages of using EMR in medicine organizations. Later we give a brief description of EMR functionality, used standards ant technologies in EMR creation process.

After all we concentrate on data manipulating problem, trying to discover the best solution to solve it by using ADO.NET data access components. Beside, we analyze all the components of ADO.NET architecture, describe situations in which different component is most suitable, suggest how to avoid performance degradation in data manipulation processes.

Data is most expensive worth in medicine. So, we must ensure that data manipulating process will be effective and will not cause any additional problems for medicine personal.

TURINYS

Lentelių sąrašas

Paveikslėlių sąrašas

1. Įvadas

Šio darbo tikslas – įvardinti problemas, kurios neigiamai veikia medicinos personalo darbą, išnagrinėti medicinai skirtų informacijos sistemų realizacijos ypatumus, įvertinti duomenų pasiekiamumo svarbą, pateikti projektinį sprendimą, realizuotą su .NET technologija, ir ištirti, projektinėje dalyje sukurtos, informacinės sistemos duomenų manipuliavimo komponentų efektyvaus naudojimo galimybes.

Darbo pradžioje aptariama bendra medicinos procesų kompiuterizacijos padėtis Lietuvoje. Iškeliama prielaida, jog būtina kurti informacines medicinos sistemas, kad išspręsti neefektyvaus medicinos personalo darbo problemas, pateikiami pavyzdžiai, lyginantys medicinos personalo darbo efektyvumą, naudojantis informacinėmis sistemomis ir dirbant su popierine dokumentacija. Analizuojama medicinai skirtų informacijos sistemų realizacijos ypatumai, pabrėžiama duomenų pasiekiamumo svarba, įvardijamos efektyvaus manipuliavimo duomenimis problemos.

Trečiajame skyriuje nagrinėjamas ADO.NET duomenų prieigos modelis, skirtas .NET technologija realizuotoms aplikacijoms. Detaliai aptariama ADO.NET architektūra, išskiriami pagrindiniai komponentai, skirti manipuliuoti duomenimis. Pateikiami kiekvieno iš tų komponentų naudojimo privalumai ir trūkumai, įvardijamos situacijos, kuriose teoriškai vieną ar kitą komponentą yra naudoti efektyviausia.

Kitame darbo skyriuje plačiau aprašomas projektinis sprendimas – KMU Širdies centro informacijos sistema. Pateikiamas sistemos funkcinis aprašymas, duomenų modelio projektavimo diagramos, duomenų bazės schema, realizuoto sprendimo architektūros modelis, išdėstymo vaizdas, išvardinami sistemos realizacijos ypatumai.

Darbo pabaigoje pateikiami KMU Širdies centro informacijos sistemoje realizacijoje naudotų duomenų manipuliavimo komponentų efektyvumo tyrimo rezultatai ir galutinės išvados.

2. Informacijos sistemos medicinoje

2.1. Bendra padėties apžvalga

Naujos technologijos, išplėtotas bendras telekomunikacijų tinklas ir vykdomos informacinės visuomenės kūrimo programos bei šiuolaikiški bandomieji tarptautiniai elektroninės sveikatos tinklai – suteikia neblogas galimybes elektroninės sveikatos plėtrai. Tačiau verta pažymėti, kad bendra IKT (informacija, komunikacija, technologija) infrastruktūra yra netinkamai naudojama sveikatos priežiūroje: „paskutinių kelių metrų“ iki interneto problema, nepakankamas turimų kompiuterių ir interneto kanalų panaudojimas, nelegali programinė įranga, mažas „skaitmenizacijos“ lygis, be to, perkamos brangiai kainuojančios sistemos, kurios greitai sensta, užuot pasirinkus lanksčias, nesunkiai pritaikomas sistemas.

Kaupiama informacija ir duomenys yra fragmentiški, kaupiami atskiruose blokuose, remiantis skirtingais standartais: viena informacija – sveikatos priežiūros įstaigoms, kita – medikams, trečia – valdymo institucijoms, o duomenys registruose renkami skirtingose platformose ir duomenų bazių valdymo sistemose. Gydytojams nėra galimybės greitai ir lengvai gauti reikalingos informacijos, kadangi kasdieninė informacija daugiausia yra kaupiama tik popieriuje, nėra taikomųjų programų, kurios leistų tokią informaciją kaupti, statistiškai apdoroti ir analizuoti. Nepakankamai įvertinama bandomųjų projektų ir tinklų svarba, trūksta nacionalinės paramos bandomajai veiklai, kuri inicijuojama ir vykdoma tarptautiniu lygiu, pirkimai vykdomi dažniau nei plėtojamos ar papildomos mokymosi platformos, slopinamos vietos iniciatyvos ir nepakankamai išnaudojama vietos profesionalų kvalifikacija [4].

2.2. Informacijos sistemų poreikis medicinoje

Priežasčių, kodėl reikia kompiuterizuoti medicinos procesus, yra labai daug. Viena iš jų tam, kad kaip įmanoma labiau minimizuoti laiką, kurį medicinos personalas praleidžia dirbdamas su popieriais. Remiantis pasauline patirtimi galima teigti, kad medicinos procesų kompiuterizavimas turi milžinišką teigiamą poveikį tiek žiūrint iš administracijos, tiek iš medicinos personalo perspektyvų. Personalo darbuotojams nebereikia ilgą laiką ieškoti paciento kortelės ar dominančios informacijos apie paciento sveikatos būklę. Yra paskaičiuota, kad pakartotinis recepto išrašymo procesas užtrunka apie 15 minučių dirbant su dokumentais ir tik 3-4 minutes dirbant su sistema, arba tarkime vidutinės laiko sąnaudos registruojant tyrimų nurodymus užtrunka apie 7-18 minučių dirbant su dokumentais ir 3-3,5

minutes dirbant su sistema [9, 14]. Be to, medicinos personalas mažiau laiko skirdamas dokumentų tvarkymui, daugiau laiko gali skirti pacientams, jų konsultavimui ar apžiūrai.

Prieiga prie paciento duomenų yra galima iš skirtingų darbo vietų, t. y. personalo darbuotojams nereikia gaišti laiko ieškant paciento kortelės kartotekoje, tai galima atlikti tiesiog iš savo darbo kabineto. Tai ypač efektyvus laiko taupymo būdas, jeigu medicinos įstaigos skyriai geografiškai yra nutolę vienas nuo kito. Be to, sistemos realizuotos internetinėmis technologijomis yra prieinamos bet kuriuo paros metu [14].

Minimizuojant popierinių duomenų kaupimą mažėja jų dubliavimo galimybė. Duomenų patikimumas medicinoje yra kritiškai svarbus, kadangi neteisingas klaidingų duomenų interpretavimas gali būti netgi gyvybiškai pavojingas.

Informacijos sistemų naudojimas akivaizdžiai praverčia ir praktikoje. Gydytojas jau procedūrų kabinete iš karto gali registruoti apžiūros ar tyrimo eigos duomenis, kurie yra prieinami tiek medicinos personalui, tiek išvykstančiam pacientui jau apžiūros ar tyrimo pabaigoje. Taip taupomas brangus laikas. Remiantis atliktų tyrimų duomenimis, tose įstaigose, kuriose medicinos sistema naudojama vienerius metus, aptarnaujamų pacientų kiekis išaugo 15% [9].

Informacijos sistemų naudojimas turi teigiamos įtakos ne tik medicinos personalui, tačiau ir patiems pacientams. Pacientui paskambinus šeimos gydytojui, šis per kelias sekundes ekrane mato skambinančiojo ligos istoriją. Gydytojas, matydamas specifinius paciento duomenis, gali efektyviau bendrauti su pacientu.

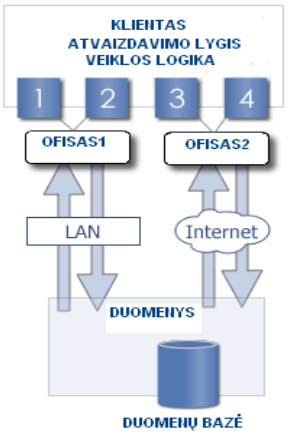
Struktūrizuotas duomenų pavidalas - dar viena priežastis, kodėl būtina kompiuterizuoti mediciną. Duomenų, sukauptų popieriuje, pakartotinis panaudojimas yra gana sudėtingas ir labai ilgas procesas. Tarkim, norint atlikti statistinę analizę kažkokiu tai medicininiu pjūviu, pirmiausia reikia rasti korteles pacientų, kurių duomenys bus analizuojami. Kuo statistinė analizė yra detalesnė, tuo sudėtingesnė reikiamų paciento duomenų paieška, grupavimas. Tarkime, norint įvertinti pacientų išgyvenamumo tikimybę, kai jie sirgo širdies nepakankamumu, gydytojams tektų ilgai „vartyti“ pacientų ligos istorijas ir ieškoti įrašų apie registruotą širdies nepakankamumą, skaityti ligos istorijos aprašus ieškant mirties priežasties, tuo tarpu paieška medicininėse sistemose užtruktų vos keletą minučių ar sekundžių. Gydytojui tereikėtų pasirinkti paciento mirties priežastį iš sistemos pateikto sąrašo ir per kelias akimirkas ekrane būtų pateikti paieškos rezultatai. Taigi struktūrizuotų duomenų kaupimo privalumas akivaizdus. Be to, duomenys sukaupti duomenų bazių valdymo sistemose gali būti perkelti į kitas sistemas ar netgi pritaikyti specializuotoms statistinės analizės taikomosioms programoms.

2.3. Medicinai skirtų informacijos sistemų realizacijos principai

Architektūros modelis

Medicinai skirtas informacijos sistemas pagal jų architektūros modelį būtų galima suskirstyti į dviejų ir trijų lygių kliento-serverio sistemas [12].

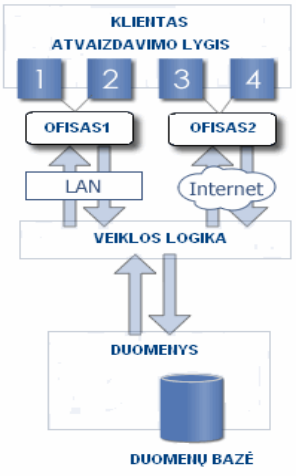
Dviejų lygių architektūros modelis:



1 pav. Dviejų lygių architektūros modelis

- **Atvaizdavimo lygis.** Klientas yra atsakingas už atvaizdavimo ir veiklos logiką. Aplikacija yra įdiegta kiekvieno kliento kompiuteryje.
- **Duomenų lygis.** Duomenų lygis sudaro galimybę prisijungti prie išorinių duomenų bazių.

Trijų lygių architektūros modelis:



2 pav. Trijų lygių architektūros modelis

- **Atvaizdavimo lygis.** Klientas yra atsakingas tik už aplikacijos vartotojo sąsają.
- **Veiklos lygis.** Veiklos lygis atsakingas už aplikacijos funkcionalumo užtikrinimą.
- **Duomenų lygis.** Duomenų lygis sudaro galimybę prisijungti prie išorinių duomenų bazių.

Sistemos realizuotos trijų lygių architektūra privalumai [6, 10]:

- Palyginti su dviejų lygių architektūra, paprastesnis sistemos naudojimas - klientas atsakingas tik už atvaizdavimo programinę įrangą.
- Nepriklausomumas nuo kalbos tarp atvaizdavimo ir veiklos lygių.
- Sistemos kūrimo proceso aiškumas.
- Mažesnės išlaidos techninei įrangai.
- Trijų lygių architektūra realizuotos sistemos kūrimo bei palaikymo išlaidos didėjant sistemos sudėtingumui didėja tiesiškai, tuo tarpu dviejų lygių architektūros – eksponentiškai.

Standartai

Medicinos sistemų realizacijoje naudojami standartai [4, 16]:

- Diagnostinei vaizdinei medžiagai taikomas DICOM standartas, kuris dabar apibrėžia tinklo protokolą, naudojantį TCP/IP, paslaugų klasių naudojimą ne tik paprastam duomenų perdavimui, siūlo mechanizmą unikaliam informacijos objektų, su kuriais dirbama tinkle, identifikavimui. DICOM apibrėžė informacijos objektus ne tik vaizdiniams, bet ir pacientams, studijoms, ataskaitoms bei kitoms duomenų grupėms. Patobulinus DICOM (3.0 versija), standartas galėjo ne tik perduoti medicininius vaizdinius tarp daugelio pardavėjų, bet ir pagerinti vaizdinių archyvavimo bei komunikacijos sistemų plėtrą ir sąveiką su medicinos informacijos sistemomis. Paprastai šis standartas naudojamas radiologijoje, tačiau gali būti pritaikytas ir kardiologijos, ultragarso, branduolinėje medicinoje, patologijoje ir dermatologijoje.
- HL7 (Health Level 7) standartas skirtas informacijai apie sveikatos būklę kaupti ir keistis. HL7 yra susijusi su bendresniais žinių perdavimo aspektais. Ji neturi elektroninio sveikatos įrašo (ESĮ) objekto modelio standarto. HL7 v.2.x teigia, kad joje nieko nėra apie pačią loginę ir fizinę paciento išilginės sveikatos kortelės sandarą. Trečia versija bus kitokia – joje bus naujas ESĮ SIG, kuris buvo užsakytas 2001 m., siekiant sukurti aukšto lygio, ESĮ reikalavimus atitinkančią architektūrą. Nors HL7 trūksta ESĮ objekto modelio standarto, jis turi tam tikrų susijusių idėjų ir pasiūlymų: dėl klinikinio dokumento architektūros (CDA – *Clinical Document Architecture*) standarto ir specialaus organo dėl dokumento ontologijos (DOTF – *Document Ontology Task Force*). CDA standartas, kuris iki šiol buvo žinomas kaip paciento įrašo architektūra (PRA – *Patient Record Architecture*), siūlo keitimosi klinikiniais dokumentais modelį (išrašymo reziumė, įrašai apie būklės pokyčius ir pan.) ir priartina sveikatos apsaugą prie ESĮ įgyvendinimo. Remiantis pasauline patirtimi – tai plačiausiai praktikoje naudojamas standartas. Jis apima tokias sritis kaip paciento registracijos duomenys, vizito metu registruoti duomenys, darbų grafikai, klinikiniai stebėjimai. Šis

standartas turi platų funkcinį pritaikymą: laboratorijų, vaistinių darbas, diagnostinės paslaugos, slauga ir kitų sričių procesų kompiuterizavimas.

- CORBAmed – objektinė technologija, kuri leidžia kurti greitai veikiančias, pakartotinai panaudojamas ir nuo platformos bei techninės įrangos nepriklausomas sistemas.

Lentelėje pateikiamos pagrindinės funkcijos, kurios dažniausiai būna realizuotos medicinai skirtose informacijos sistemose, ir palaikomi standartai:

1 lentelė. Standartų ir funkcijų palyginimas

	STANDARTAI		
FUNKCIJOS	DICOM	HL7	CORBAmed
Asmens identifikavimas (Master Pacient Index)	**-**	**-**	**X**
Paciento registravimas, perkėlimas, išrašymas	**X--**	**X**	**-**
Būklės administravimas	**-**	**X**	**-**
Darbų planavimas	**-**	**X**	**-**
Paveikslėlių įkėlimas	**X**	**-**	**-**
Prieiga prie paveikslėlių	**X**	**X--**	**-**
Rezultatai ir ataskaitos	**X**	**X**	**X**

X – pilnas funkcionalumo palaikymas.

X-- – ribotas funkcionalumo palaikymas.

Kuriant naują sistemą, nėra tiesaus atsakymo, kada ir kokį standartą naudoti. Standartai turėtų būti pritaikomi atsižvelgiant į kompiuterizuojamos aplinkos specifiką.

Duomenų kaupimas

Medicinai skirtose informacijos sistemose kaupiami duomenys skirstomi į „koduotus“ ir „laisvo teksto“. Pirmuoju atveju saugomas laukas įrašomas į duomenų bazę kaip atskiras atributas. Tokio metodo privalumas tas, kad kiekvienas elementas, kuris yra saugomas duomenų bazėje, ateityje gali būti panaudotas paieškose ar duomenų užklausose. Tokio tipo duomenų kaupimas vertingas toms gydymo įstaigoms, kurios atlieka tyrimus siekdamos nustatyti, kiek pacientų serga vienokia ar kitokia liga, pasižymi specifine sveikatos būkle. Šio metodo trūkumas yra tas, kad duomenų registravimas yra imlus laikui, t. y. gydytojui tenka daug kartų pažymėti/pasirinkti formos laukelius, kol užpildomas visas dokumentas, kai „laisvo teksto“ sistemose duomenų registravimas yra paprastas ir įprastas procesas. Tačiau tokių sistemų sukauptų duomenų pakartotinis panaudojimas yra labai ribotas [5].

Duomenų organizavimas

Dauguma popierinių medicininių kortelių turi „apibendrinimo lapus“, kuriuose pateikiama apibendrinta informacija apie paciento būklę. Kalbant apie medicinos sistemas, tai taip pat svarbu, kadangi gydytojui nereikia per kelias skirtingas vietas žiūrėti ir ieškoti reikiamos informacijos. Tokiu būdu gydytojo darbas tampa efektyvesnis.

Be to, duomenų registravimo laukai lange turi būti išdėstyti taip, kad gydytojui būtų nesudėtinga orientuotis ir manipuliuoti tais laukais. Šios problemos vienas iš sprendimų – langams pritaikyti panašų elementų išdėstymo dizainą, kaip iki tol naudotose duomenų registravimo formose ar lapuose.

Prieinamumas

Kuriant medicinai skirtą informacijos sistemą reikia atsakyti į tokius klausimus: ar sistema leidžia daugiau nei vienam sistemos vartotojui tuo pačiu metu naudotis vienu ir tuo pačiu įrašu, ar yra galimybė prieiti prie sistemos iš išorės, ar sistema leidžia tuo pačiu metu dirbti su keliais įrašais.

HIPAA standartų laikymasis

HIPPA – standartas, kurio pagrindinis tikslas yra užtikrinti paciento duomenų saugumą. Norint užtikrinti paciento duomenų saugumą, turi būti realizuotas sistemos vartotojų veiksmų registracijos žurnalas, kuriame saugoma informacija apie tai, koks gydytojas peržiūrėjo ar keitė paciento duomenis, galima naudoti įvairias šiuolaikines technologijas, užtikrinančias papildomą saugumą. Tarkim, atlikus apklausą, kurioje dalyvavo 100 gydymo įstaigų, net 93% respondentų išskyrė HIPAA kaip vieną iš esminių saugumo užtikrinimo priežasčių. Technologijų, užtikrinančių saugumą, naudojimo pasiskirstymas tarp respondentų atrodė taip: anti-virusinė programinė įranga (100%), saugasienių naudojimas (96%), virtualaus privataus tinklo naudojimas (83%), duomenų kodavimas (65%), įsibrovimų aptikimas (60%), pažeidžiamumo įvertinimas (57%), viešo rakto infrastruktūra (20%) ir biometrikų naudojimas (10%) [8].

2.4. Duomenų pasiekiamumo svarba

Ankstesniuose skyreliuose buvo apžvelgti informacijos sistemų naudojimo medicinoje ir tokių sistemų realizacijos klausimai. Ten daugiau buvo koncentruojamasi ties duomenų kaupimo bei panaudojimo problemomis, įvardinti sprendimai, kaip didinti gydytojų darbo efektyvumą funkciniu atžvilgiu, tačiau praktiškai nieko neužsiminta apie tokius nefunkcinius reikalavimus sistemoms, kaip manipuliavimo duomenimis greičiai, tinklo pralaidumas ir kiti. Galbūt nefunkcinių reikalavimų problemą tikimasi išspręsti, naudojant vien techninę įrangą, nuperkant serverį su maksimalios spartos procesoriumi ir maksimaliu vidinės atminties kiekiu. Tačiau neoptimaliai parašytas sistemos kodas gali stabdyti darbą net ir dirbant su naujausiais kompiuteriais. Be to, laikui bėgant techninė įranga sensta, duomenų kiekiai sukaupti duomenų bazėje didėja, o sistemos veikimo charakteristikos prastėja. Jeigu aplikacija pritaikyta darbui internete, dar pridėkime tinklo apkrautumo, vėlinimo problemas ir sistemos vartotojui gali susidaryti toks įspūdis, jog sistema nebeveikia.

Kauno medicinos universiteto Širdies centre per metus yra užregistruojama daugiau kaip 10 000 į stacionarą guldomų pacientų ir per 4 000 kardiologinių tyrimų. Registruojama informacija yra labai įvairiapusė, t. y. registruojami dideli informacijos kiekiai. Tarkime, vienoje formoje registruojamų parametrų kiekis svyruoja nuo 10 iki 160. Kadangi sąveika su sistema vyksta interaktyviai, atsakymai net į daug skaičiavimų reikalaujančias užklausas turi būti gaunami kuo greičiau. Greita užklausa reiškia, kad rezultatai turi būti grąžinami per keletą sekundžių. Atlikti tyrimai parodė, jog vartotojai užklausą laiko nesėkminga, jei ji negrąžina rezultato per 30 sekundžių.

Duomenys sistemoje yra kaupiami siekiant juos pakartotinai panaudoti ataskaitoms įvairiais pjūviais formuoti ir statistinei analizei. Kai duomenų yra nedaug, jų analizę galima atlikti pasitelkus ir tradicines sąryšines duomenų bazių valdymo sistemas (RDBVS), kurios kartu naudojamos ir sandoriams tikralaiku vykdyti. Jose esančių duomenų analizė dažniausiai remiasi įprastų užklausų, parašytų SQL kalba, vykdymu. Tačiau kai duomenų susikaupia daug, dažnai susiduriama su keliomis problemomis [13]:

- Didelių skaičiavimo pajėgumų reikalaujančios intensyviai vykdomos analitinės užklausos neigiamai veikia sistemos produktyvumą.
- Jeigu duomenų bazėje saugoma istorinė informacija, nereikalinga kasdienėms operacijoms atlikti, tokios sistemos produktyvumas mažėja, nes duomenų laikui bėgant vis daugėja.
- Neteisingai naudojami duomenų prieigos komponentai taip pat neigiamai veikia sistemos produktyvumą.

Siekiant išvengti pirmųjų dviejų problemų ir išsaugoti istorinius duomenis, kuriamos atskiros duomenų saugyklos, kuriose talpinama jau istorine tapusi informacija iš darbinės duomenų bazės. Istorinių duomenų saugojimą ir analizę perkėlus į duomenų saugyklas, išvengiama padidėjusių darbinių duomenų bazių serverių apkrovų. Tačiau toks problemos sprendimas nepriimtinas medicininėse sistemose, kadangi čia duomenys turi būti saugomi tik darbinėje duomenų bazėje, kadangi jų poreikis gali iškilti bet kurią akimirką. Taigi vienas iš būdų pagerinti sistemos darbo produktyvumą – efektyvus manipuliavimo duomenimis komponentų panaudojimas.

3. Prieigos komponentų naudojimas duomenų apdorojimo procesuose

ADO.NET yra duomenų prieigos modelis skirtas .NET technologija realizuotoms aplikacijoms. ADO.NET buvo sukurta tam, kad atitiktų naują programavimo modelį: atjungtų (angl. *disconnected*) duomenų architektūra, glaustas ryšys su XML, paprastas duomenų atvaizdavimas su galimybe sujungti duomenis iš kelių skirtingų duomenų šaltinių ir optimizuota prieiga prie duomenų bazės [2].

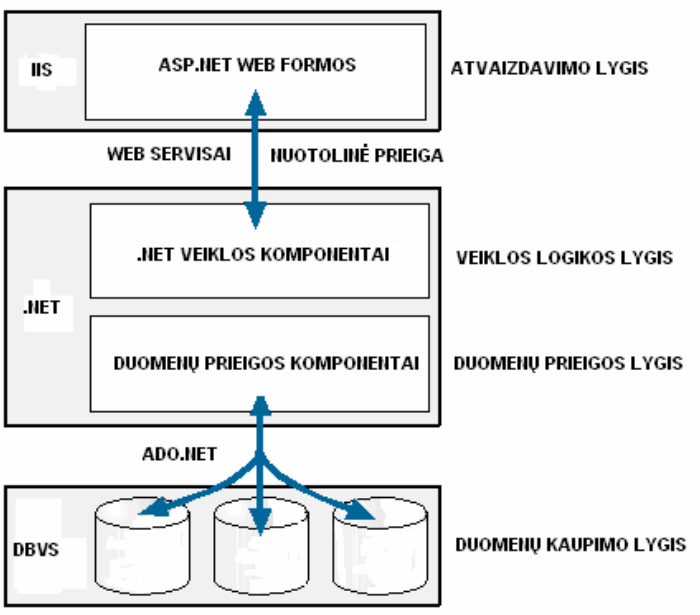
3.1. ADO.NET duomenų prieigos komponento projektavimo tikslai

Prieš projektuojant ADO.NET jai buvo iškelti tokie tikslai [11]:

- Abstraktaus n-lygių programavimo modelio palaikymas.
- Pritaikymas greitoms, į sandorius orientuotoms aplikacijoms.
- Nepriklausoma duomenų prieiga veiklos lygyje jungta su galingomis spartinančiosios atmintinės galimybėmis.
- XML panaudojimo galimybė.

3.2. Trijų lygių programavimo modelis

.NET aplikacijoms realizuoti rekomenduojama kliento-serverio architektūra su visiška vartotojo sąsajos abstrakcija, veiklos ir duomenų lygiais. Abstrakcijos tarp atskirų lygių palaikymas užtikrina lengvesnį sistemos palaikymą bėgant laikui, padidina atskirų modulių pakartotinio panaudojimo galimybę [11].

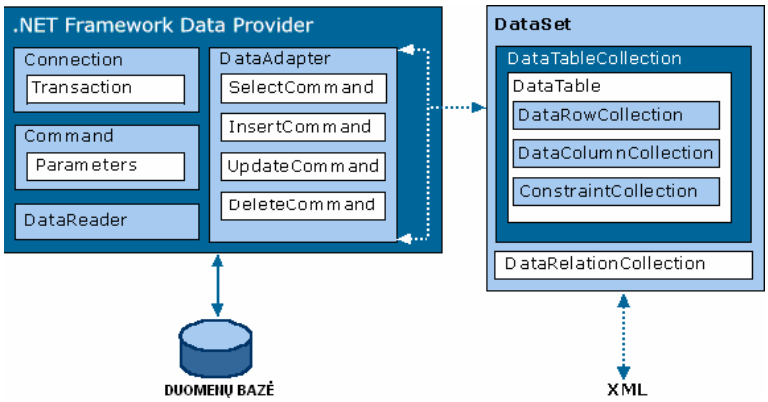


3 pav. Trijų lygių programavimo modelis

Trečiame paveikslėlyje matome, jog ADO.NET atlieka sąsajos tarp duomenų prieigos ir duomenų kaupimo lygių vaidmenį.

3.3. ADO.NET architektūra

Detalus ADO.NET architektūros modelis:



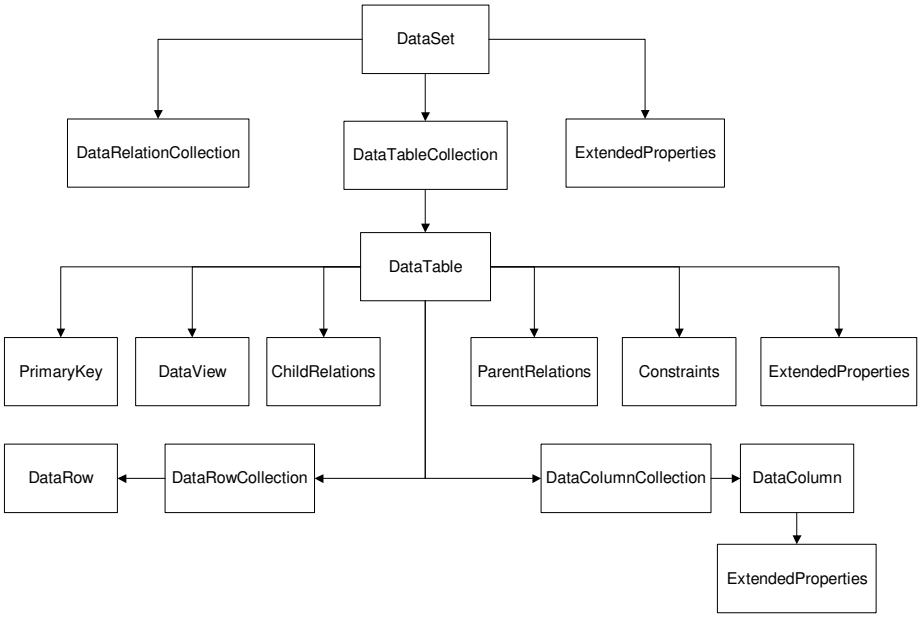
4 pav. ADO.NET architektūra

Šis modelis turi du kertinius komponentus: DataSet ir .NET Framework duomenų aprūpintoją (angl. *data provider*), kuriuos plačiau panagrinėsime kituose skyreliuose.

3.3.1. DataSet komponentas

DataSet komponentas buvo suprojektuotas duomenų prieigai nepriklausomai nuo duomenų šaltinio. Šis komponentas gali būti naudojamas su keletu skirtingų duomenų šaltinių, XML duomenimis ar vidinių aplikacijos duomenų valdymui. Be to, DataSet komponentas gali būti naudojamas kaip XML duomenų srautas duomenų sandoriams tarp heterogeninių platformų [1].

ADO.NET DataSet komponento architektūra:



5 pav. DataSet komponento architektūros modelis

Kaip matome paveikslėlyje DataSet sudaro DataTable objektų rinkinys, susidedantis iš duomenų eilučių bei stulpelių, turinčių pirminius bei antrinius raktus, ryšius tarp tų objektų ir tų ryšių apribojimus. ADO.NET naudoja DataAdapter objektą duomenų ištraukimui ir įrašymui į DataSet komponentą.

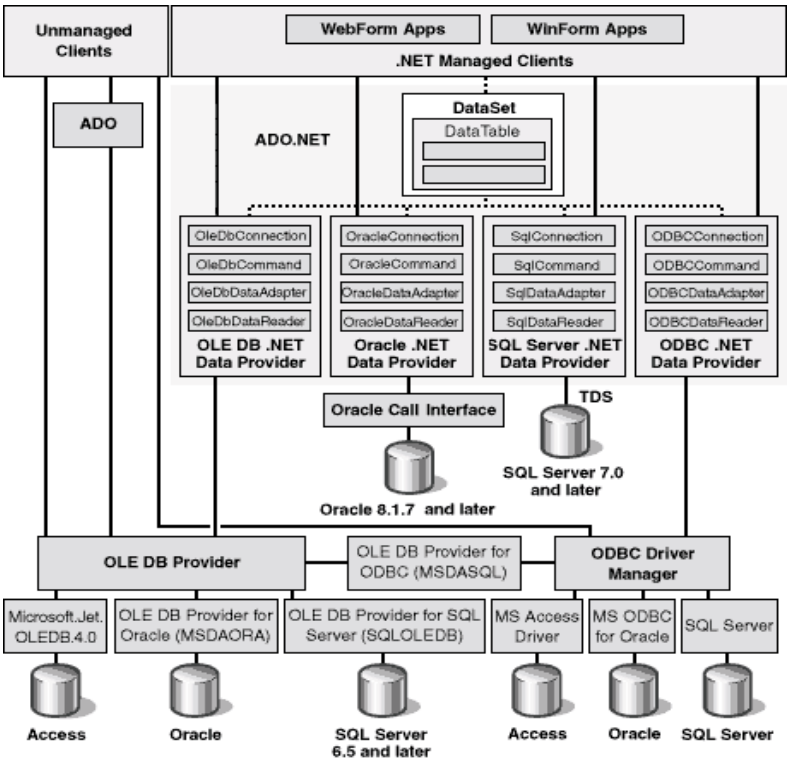
3.3.2. .NET Framework duomenų aprūpintojas

Kitas kertinis ADO.NET architektūros komponentas yra NET Framework duomenų aprūpintojas, kuris buvo suprojektuotas duomenų manipuliacijai ir greitam, vienkrypčiam duomenų skaitymui. .NET Framework duomenų aprūpintojas jungia aplikaciją su duomenų šaltiniu.

.NET Framework gali būti panaudoti šie duomenų aprūpintojai:

- SQL Server .NET.
- Oracle .NET.
- OLE DB .NET.
- ODBC .NET.

.NET Framework duomenų aprūpintojo naudojimo galimybės:



6 pav. .NET Framework duomenų aprūpintojo panaudojimo galimybių diagrama

.NET Framework duomenų aprūpintoją sudaro Connection, Command, DataReader ir DataAdapter objektai. Connection objektas užtikrina prisijungimą prie duomenų šaltinio. Command objektas įgalina prisijungimą prie duomenų bazės, užtikrina duomenų grąžinimą,

keitimą, serverio procedūrų paleidimą ir informacijos apie parametrus siuntimą bei grąžinimą. DataReader objektas užtikrina greitą duomenų srauto siuntimą iš duomenų šaltinio [1].

3.4. Sistemos veikimo charakteristikas lemiantys veiksniai

Medicinai skirtose informacijos sistemose bene svarbiausias kriterijus yra efektyvus duomenų naudojimas. Jeigu sistema veikia neefektyviai, iškyla klausimas dėl jos naudojamumo. Siekiant nustatyti, ar sistema veikia efektyviai, reikia ištirti tos sistemos veikimo charakteristikas tokias kaip skaičiavimų greitis, reagavimo laikas, pasiekiamumas ir palyginti gautus rezultatus su teoriniais. Eksperimento metu bus tiriama serverio procedūrų panaudojimo efektyvumas lyginant su tiesioginiais SQL kreipiniais, DataSet ir DataReader duomenų prieigos komponentų efektyvaus panaudojimo galimybės, naudojant įvairaus sudėtingumo SQL kreipinius, ir spartinančiosios atmintinės naudojimo galimybės.

3.4.1. Serverio procedūros ir tiesioginiai SQL kreipiniai

Keletas priežasčių, kodėl reikia naudoti serverio procedūras, o ne tiesioginius SQL kreipinius [3]:

- Naudojant serverio procedūras, gerėja sistemos veikimo charakteristikos, kadangi duomenų bazė gali optimizuoti duomenų prieigos planą ir naudoja spartinančiąją atmintinę pakartotiniam duomenų panaudojimui.
- Serverio procedūrų naudojimas sumažina tinklo užimtumą.
- Naudojant serverio procedūras, lengviau atlikti pakeitimus.
- Papildomas saugumo lygis serverio procedūroms pačioje duomenų bazėje.

3.4.2. Duomenų išgavimo objektai

Duomenų išgavimo objektai [15]:

- DataSet.
- SqlDataReader.
- XmlReader.

DataSet objekto panaudojimas

DataSet objektą patartina naudoti tada, kai:

- Reikalingi atjungti duomenys, kad būtų galima perduoti juos kitam komponentui arba aplikacijos lygiui.
- Dirbama su duomenimis iš skirtingų duomenų šaltinių, pavyzdžiui, skirtingų duomenų bazių, lentelių ar failų.
- Norima atnaujinti, keisti kai kuriuos arba visus išgautus įrašus.
- Norima surišti išgautus duomenis su valdikliu, kuris reikalauja, kad duomenų šaltinis palaikytų IList sąsają.

Papildoma informacija:

- Jeigu DataSet ar DataTable objekto kūrimui naudojamas SqlDataAdapter, tuomet nereikia atidaryti ar uždaryti prisijungimo prie duomenų bazės. SqlDataAdapter Fill metodas pats atidaro prisijungimą prie duomenų bazės ir uždaro jį, prieš grąžindamas rezultatus.
- Jeigu operacijos metu prisijungimas prie duomenų bazės jau yra atidarytas, tai Fill metodas jį palieka atidarytą ir po rezultatų grąžinimo.
- Jeigu prisijungimas prie duomenų bazės reikalingas kokiems nors kitiems tikslams, patartina jį atidaryti prieš Fill metodą. Tokiu būdu yra išvengiama bereikalingų atidaryti/uždaryti prisijungimą operacijų, o kartu gerinamos veikimo charakteristikos.
- Jeigu SqlCommand objektas naudojamas, kad įvykdyti tą pačią komandą keletą kartų, patartina nenaudoti to paties SqlCommand objekto skirtingų komandų vykdymui.

SqlDataReader komponento panaudojimas

SqlDataReader komponentą patartina naudoti, kai:

- Operuojama dideliais duomenų kiekiais.
- Siekiama sumažinti aplikacijos naudojamos atminties kiekį.
- Siekiama išvengti papildomų resursų panaudojimo objekto sukūrimui.
- Norima surišti išgautus duomenis su valdikliu, kuris reikalauja, kad duomenų šaltinis palaikytų IEnumerable sąsają.
- Norima optimizuoti prieigą prie duomenų.
- Iš duomenų bazės nuskaitomos eilutės turinčios BLOB tipo stulpelius.

Papildoma informacija:

- Atidarytas prisijungimas prie duomenų bazės negali būti panaudotas jokiems kitiems tikslams, kol SqlDataReader objektas yra aktyvus. Patartina uždaryti SqlDataReader objektą, kaip įmanoma greičiau.
- Su vienu prisijungimu prie duomenų bazės gali būti naudojamas tik vienas SqlDataReader objektas.
- Prisijungimą prie duomenų bazės reikia uždaryti iš karto, kai tik baigiamas darbas su SqlDataReader.
- Nuskaitant duomenis iš duomenų šaltinio, rekomenduojama naudoti tipizuotus prieigos metodus tokius, kaip GetInt32 ar GetString, jeigu tiksliai žinomas atitinkamo lentelės stulpelio tipas. Tokiu būdu sumažinamos laiko sąnaudas, reikalingos papildomam duomenų konvertavimui.

XmlReader komponento panaudojimas

XMLReader komponentą patartina naudoti, kai:

- Norima naudoti išgautus duomenis kaip XML, bet nenorima pabloginti veikimo charakteristikų dėl DataSet komponento sukūrimo.
- Norima išnaudoti SQL Server 2000 FOR XML sakinio funkcionalumą, leidžiantį efektyvų ir lankstų XML duomenų fragmentų išgavimą iš duomenų bazės.

Papildoma informacija:

- Prisijungimas prie duomenų bazės privalo būti atidarytas visą laiką, kol XmlReader skaito duomenis iš duomenų bazės.
- Rekomenduojama naudoti tik tada, kai reikia išgauti vieną eilutę.

Duomenų prieigos komponento pasirinkimas

Apibendrinant tai kas buvo aptarta šiame skyrelyje, būtų galima daryti išvadą: SqlDataReader duomenų prieigos komponentą patartina naudoti, jeigu siekiama optimizuoti sistemos veikimo charakteristikas, o duomenys ištraukti iš duomenų bazės reikalingi tik atvaizdavimui. Jeigu duomenys naudojami iš kelių skirtingų duomenų šaltinių arba su ištrauktais duomenimis, norima atlikti papildomus veiksmus, pavyzdžiui, duomenų atnaujinimą, rikiavimą pagal pasirinktą atributą ar kažką panašaus, patartina naudoti DataSet/DataTable komponentą. XMLReader komponento naudojimo galimybės labai panašios į DataSet/DataTable. Vienintelis akivaizdus XMLReader komponento naudojimo privalumas lyginant su DataSet/DataTable - efektyvesnis darbas su XML formato duomenimis.

3.4.3. Spartinančioji atmintinė

.NET technologija realizuojama aplikacija turėtų būti projektuojama taip, kad maksimaliai išnaudotų ASP.NET spartinančiosios atmintinės savybę, kadangi tai gali akivaizdžiai pagerinti sistemos veikimo charakteristikas.

ASP.NET spartinančiosios atmintinės panaudojimo metodai:

- ASP.NET puslapio talpinimas į spartinančiąją atmintinę.
- ASP.NET puslapio fragmento talpinimas į spartinančiąją atmintinę.
- Duomenų talpinimas į spartinančiąją atmintinę.

Yra suformuota keletas nuostatų, kada turėtų būti naudojama spartinančioji atmintinė:

- Jei tam tikri duomenys aplikacijoje yra naudojami daugiau nei vieną kartą.
- Naudojant bendresnius, ne specifinius sistemos duomenis, pavyzdžiui, klasifikatorių sąrašus, išgaunamus iš išorinio duomenų šaltinio.
- Naudojami statiniai puslapio elementai tokie kaip meniu, apatinė (angl. *footer*) ar viršutinė (angl. *header*) puslapio dalis, organizacijos kontaktiniai duomenys.

- Naudojant sistemos ar paieškos formas, kurios yra naudojamos keliuose skirtinguose puslapiuose.

Spartinančiosios atmintinės savybė gali būti gana plačiai naudojama ir taip pagerinti sistemos veikimo charakteristikas. Tačiau reikėtų pabrėžti, jog spartinančiosios atmintinės savybės naudojimas turėtų būti labai pasvertas, nes priešingu atveju gali atsitikti taip, kad sistemos veikimo charakteristikos ne tik kad ne pagerės, bet dar ir pablogės.

Neigiamą poveikį sukeliantys veiksniai:

- Į spartinančiąją atmintinę talpinama viskas be išimties – ir specifiniai duomenys, ir duomenys, kurie naudojami tik vieną kartą sistemos veikimo metu, ir klasifikatorių sąrašai, ir t. t..
- Per daug apkrauta vidinė atmintis.
- DataSet komponentų su dideliais duomenų kiekiais talpinimas į spartinančiąją atmintinę.

3.4.4. Prisijungimų kaupiklis

Kiekvieno naujo TCP prisijungimo iš internetinės sistemos prie duomenų bazių valdymo sistemos kūrimo procesas neigiamai įtakoja sistemos veikimo charakteristikas. Šios problemos sprendimui buvo sukurtas prisijungimų kaupiklis (angl. *connection pooling*), kurio veikimo principas – pakartotinis prisijungimų naudojimas. Objektas sistemos inicializacijos metu sukuria, per parametrus nurodytą, TCP prisijungimų kiekį ir patalpina juos į prisijungimų kaupiklį. Naudojant šį objektą, nereikia kiekvieną kartą kurti vis naujo TCP prisijungimo kiekvienai užklausai. Sistema pasiima laisvą prisijungimą iš prisijungimų kaupiklio. Naujas TCP prisijungimas kuriamas tik tuo atveju, kai visi prisijungimai prisijungimų kaupiklyje yra užimti. Kai tik prisijungimas uždaromas, jis grąžinamas į prisijungimų kaupiklį [7].

Efektyvus prisijungimų kaupiklio naudojimas:

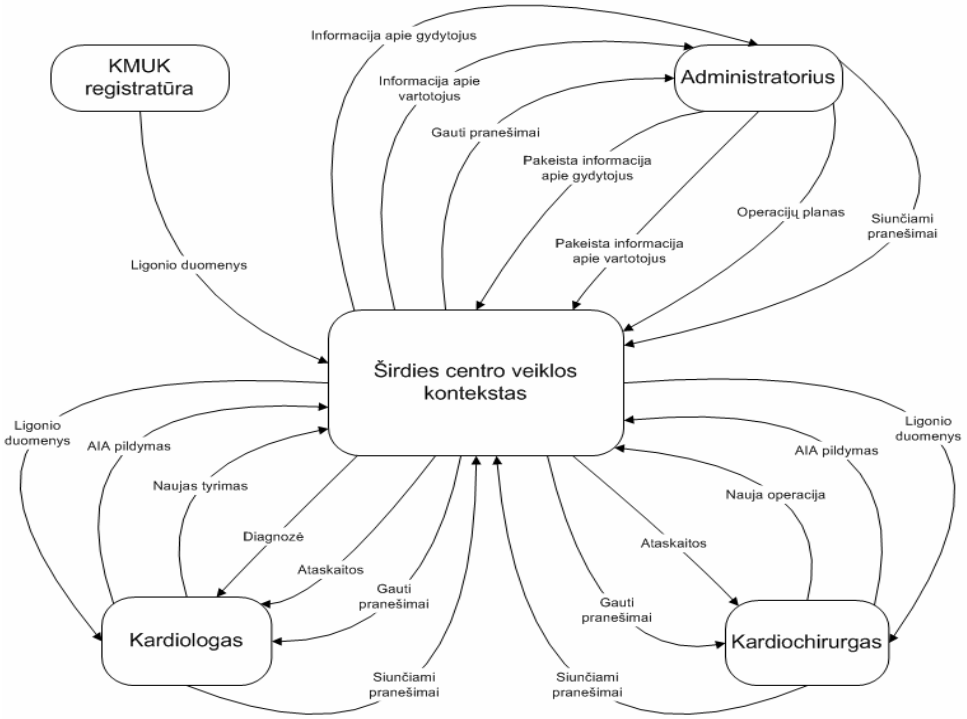
- Atidaromas prisijungimas, atliekami reikiami veiksmai ir kuo greičiau uždaromas prisijungimas. Geriau kelis kartus atlikti prisijungimo atidarymo/uždarymo operaciją, negu naudoti atidarytą prisijungimą viso darbo metu.
- Patartina naudoti tą patį prisijungimų kelią (angl. *connection string*). Jeigu yra naudojama keletas prisijungimo kelių - neišnaudojami prisijungimų kaupiklio privalumai.
- Jeigu sistemoje yra naudojamas integruotas autentifikacijos modelis ir didelis skaičius identifikuojamų vartotojų, prisijungimų kaupiklio naudojimas bus neefektyvus.

4. KMU Širdies centro informacijos sistemos sprendimai

4.1. Sistemos apibūdinimas

Sukurtas produktas – kardiologinių tyrimų surinkimo bei analizės programų sistema. Tai produktas, skirtas organizacijai, kurios pagrindinė veikla – kardiologinių bei kardiochirurginių duomenų kaupimas bei pakartotinis jų panaudojimas. Šio produkto pagalba bus registruojami KMU Širdies centre aptarnaujamų pacientų būklės, jiems atliekamų tyrimų, operacijų, statistinei analizei naudojami CARDS duomenys, iš registruotų duomenų formuojamos diagnozės, ataskaitos, vykdomi pacientų perkėlimai iš vienos skyriaus į kitą, išrašymai namo ar į kitą gydymo įstaigą.

Veiklos konteksto diagramoje pavaizduoti esminiai sistemos vartotojai ir duomenų srautai tarp vartotojų ir sistemos.



7 pav. Veiklos konteksto diagrama

Lentelėje pateikiami veiklos konteksto diagramoje esantys įvykiai bei tų įvykių įeinantys arba išeinantys informacijos srautai:

2 lentelė. Veiklos konteksto įvykiai

Eil. Nr.	Įvykio pavadinimas	Įeinantys/išeinantys informacijos srautai
1.	Registratūra perduoda paciento duomenis (asmens duomenys, asmens istorijos numeris, istorijos atvejis)	Ligonio duomenys (in)
2.	Perduodama informacija apie Širdies centro gydytojus	Informacija apie gydytojus (out)

3.	Perduodama informacija apie programų sistemos vartotojus (vartotojo vardas, kuriai vartotojų grupei priklauso, kuriuos resursus naudoja)	Informacija apie vartotojus (out)
4.	Informacijos apie gydytoją atnaujinimas	Atnaujinta informacija apie gydytojus (in)
5.	Vartotojų administravimas	Atnaujinta informacija apie vartotojus (in)
6.	Pranešimai gauti iš kitų sistemos vartotojų	Gauti pranešimai (out)
7.	Pranešimo siuntimas nurodytiems vartotojams	Siunčiami pranešimai (in)
8.	Informacijos apie paciento būklę pildymas	AIA pildymas (in)
9.	Asmeniui registruojamas naujai atliktas tyrimas	Naujas tyrimas (in)
10.	Asmeniui registruojama naujai atlikta operacija	Nauja operacija (in)
11.	Diagnozės formavimas asmeniui	Diagnozė (out)
12.	Ataskaitos spausdinimas nurodant konkrečius parametrus	Ataskaitos (out)
13.	Sudaromas planuojamų atlikti operacijų planas	Operacijų planas (in)

4.2. Sistemos funkcinis aprašymas

Kadangi programų sistema operuoja konfidencialiais KMU ligonių duomenimis, buvo realizuota autorizuota prieiga prie duomenų, tai yra tik atitinkamos kvalifikacijos medicinos personalo darbuotojai gali prisijungti prie sistemos ir naudotis duomenimis. Skirtingos kvalifikacijos medicinos personalo darbuotojai operuoja skirtingais sistemos resursais.

Sistemoje realizuotas universalus resursų bei vartotojų administravimo modulis, leidžiantis kurti naujus sistemos vartotojus, vartotojų grupes (kiekviena vartotojų grupė naudoja skirtingus sistemos resursus), priskirti vartotojus konkrečioms vartotojų grupėms, paskirti arba atimti galimybė vartotojų grupėms naudotis tam tikrais resursais (meniu, formomis). Už visus šiuos veiksmus yra atsakingas sistemos administratorius. Be to, jis dar administruoja KMU gydytojų duomenis, t.y. gali įterpti naują gydytoją, atnaujinti informaciją apie pasirinktą gydytoją arba pašalinti pasirinktą gydytoją iš sąrašo.

Iš pradžių kiekvienam vartotojui administratorius sukuria prisijungimo vardą bei slaptažodį. Vėliau kiekvienas vartotojas slaptažodį gali pasikeisti.

Sistemoje realizuota galimybė vartotojams vieni kitiems siųsti pranešimus, juos skaityti, saugoti arba šalinti iš gautų pranešimų sąrašo. Jeigu vartotojas gavo nors vieną naują pranešimą, prisijungęs prie sistemos pirmajame puslapyje jis mato užrašą, jog yra gautas naujas pranešimas.

Sistemoje realizuota ligonio guldymo paieška. Paiešką galima atlikti pagal penkis parametrus. Paieškos rezultatai pateikiami ekrane. Pasirinkus bet kurį įrašą, toliau yra operuojama pasirinkto ligonio duomenimis, t.y. visi veiksmai yra atliekami su pasirinktu ligoniu. Yra galimybė peržiūrėti bei papildyti ligonio anketinius duomenis, išsaugoti arba atnaujinti kardiologinius, diagnozės, CARDS duomenis. Be to, išspausdinti iš kardiologijos

duomenų suformuotą hospitalizacijos diagnozę. Registruoti naujus tyrimus (echokardiografija, veloergometrija, koronarografija, PTCA), pasirinkti dominantį tyrimą iš tyrimų sąrašo, atnaujinti pasirinkto tyrimo duomenis, spausdinti tyrimo rezultatus. Registruoti naują operaciją, nurodant bendrąją informaciją, pasirinktą metodiką, AVJSO, vožtuvų chirurgijos duomenis bei kitas, prieš tai ligoniui atliktas, operacijas, pasirinkti dominančią operaciją iš operacijų sąrašo, atnaujinti pasirinktos operacijos duomenis, spausdinti operacijos rezultatus. Registruoti naują perkėlimą/išrašymą, pasirinkti dominantį perkėlimą/išrašymą iš sąrašo, atnaujinti pasirinkto perkėlimo/išrašymo duomenis, spausdinti perkėlimo/išrašymo rezultatus.

Siekiant palengvinti kai kurių formų pildymą, buvo realizuota TLK kodo (ligos aprašymo) bei gydymo įstaigų paieška. Be to, buvo realizuota galimybė atlikti dinaminę CARDS, tyrimų, operacijų, kardiologijos, planuojamo guldymo paiešką pagal vartotojo pasirinktus parametrus.

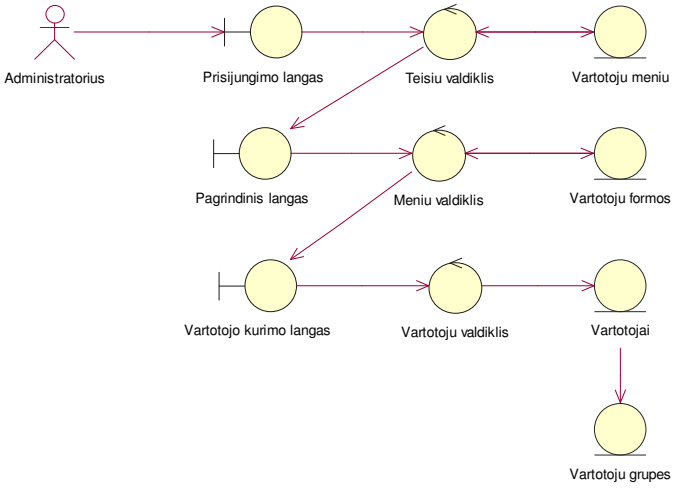
Tam kad susidaryti bendresnį sistemos vaizdą, pateikiamas sistemos panaudojimo atvejų modelio diagrama:

8 pav. Panaudojimo atvejų diagrama

4.3. Loginio sistemos modelio sudarymas

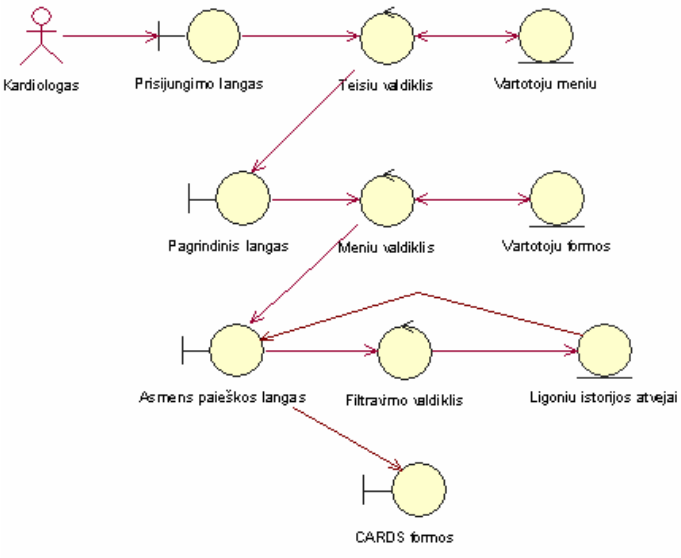
Loginio sistemos modelio sudarymo metu konceptualus sistemos vaizdas yra suskaidomas į smulkesnius loginius modulius, pavyzdžiui, konkretus panaudojimo atvejis gali būti įvardintas kaip atskiras loginio modelio modulis. Tada, nagrinėjant panaudojimo atvejų scenarijus, kiekvienam iš panaudojimo atvejų yra išskiriami objektai, tų objektų atributai, teikiamos paslaugos ir priklausomybės tarp tų objektų. Objektų bei priklausomybių tarp jų atvaizdavimui buvo pasirinktos detalizuotos klasių diagramos. Pateikiame sistemos vartotojo sukūrimo, atnaujinimo bei CARDS duomenų pildymo detalizuotas klasių diagramas.

Naujo sistemos vartotojo kūrimo, pasirinkto konkretaus vartotojo duomenų atnaujinimo detalizuota klasių diagrama:



9 pav. Naujo vartotojo kūrimas, atnaujinimas

CARDS duomenų pildymo detalizuota klasių diagrama:



10 pav. CARDS pildymas

4.4. Duomenų projektavimas

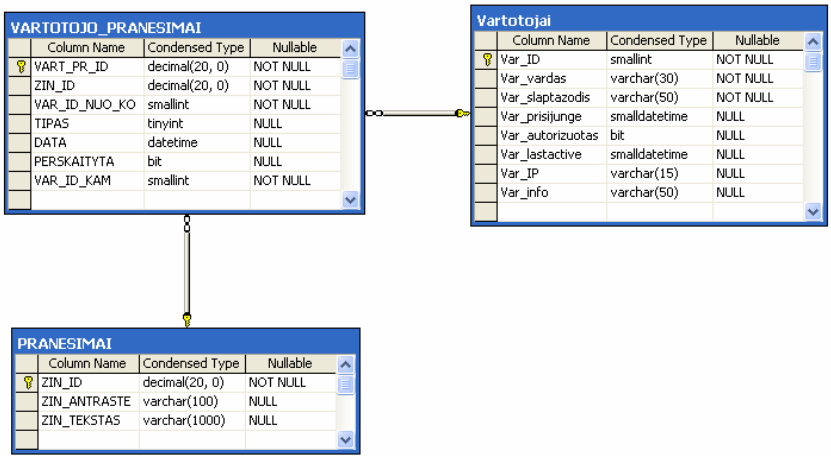
Fizinio modelio sudarymo metu, panaudojant loginio modelio metu įvardintus objektus, suprojektuojama duomenų bazės schema. Projektavimo įgyvendinimui buvo pasirinktas sąryšinis duomenų bazės realizavimo modelis. Sąryšinis modelis yra vienas iš labiausiai naudojamų modelių praktikoje, kadangi lentelės tarpusavyje gali būti susietos naudojant unikalius laukus. Sąryšinis modelis pagrindinį dėmesį skiria duomenų saugojimui, išgavimui ir duomenų integralumo palaikymui. Susiję duomenų įrašai gali būti greitai išgauti naudojant SQL užklausas, nepaisant to ar duomenys yra saugomi vienoje ar keliose tarpusavyje surištose lentelėse. Duomenų integralumas gali būti tvarkomas naudojant taisykles bei apribojimus. Projektuojant duomenų bazę bus naudojami tik sisteminiai duomenų tipai, t. y. mes apsiribosime duomenų bazių valdymo sistemos standartiniais duomenų tipais. Nustatant ryšius tarp atskirų lentelių bus naudojami pirminis bei antrinis raktai, naudojami vienas-su-vienu arba vienas-su-daug kardinalumai. Taigi, loginiame bei fiziniame modeliuose buvo išspręstas klausimas, kokie duomenys turi būti saugomi.

4.5. Duomenų vaizdas

Realizuojant KMU Širdies centro informacijos sistemą, buvo suprojektuotas duomenų bazės modelis, kurį sudarė daugiau kaip 60 duomenų lentelių, skirtų duomenims apie pacientą kaupti, ir apie 20 duomenų lentelių, skirtų klasifikatorių administravimui.

Kadangi nebuvo įmanoma duomenų modelio fiziškai atvaizduoti viename lape, tai jis buvo suskirstytas į tam tikras logines grupes: pranešimų, CARDS, vartotojų administravimo, tyrimų, operacijų, universalios paieškos bei kardiologinių duomenų kaupimo.

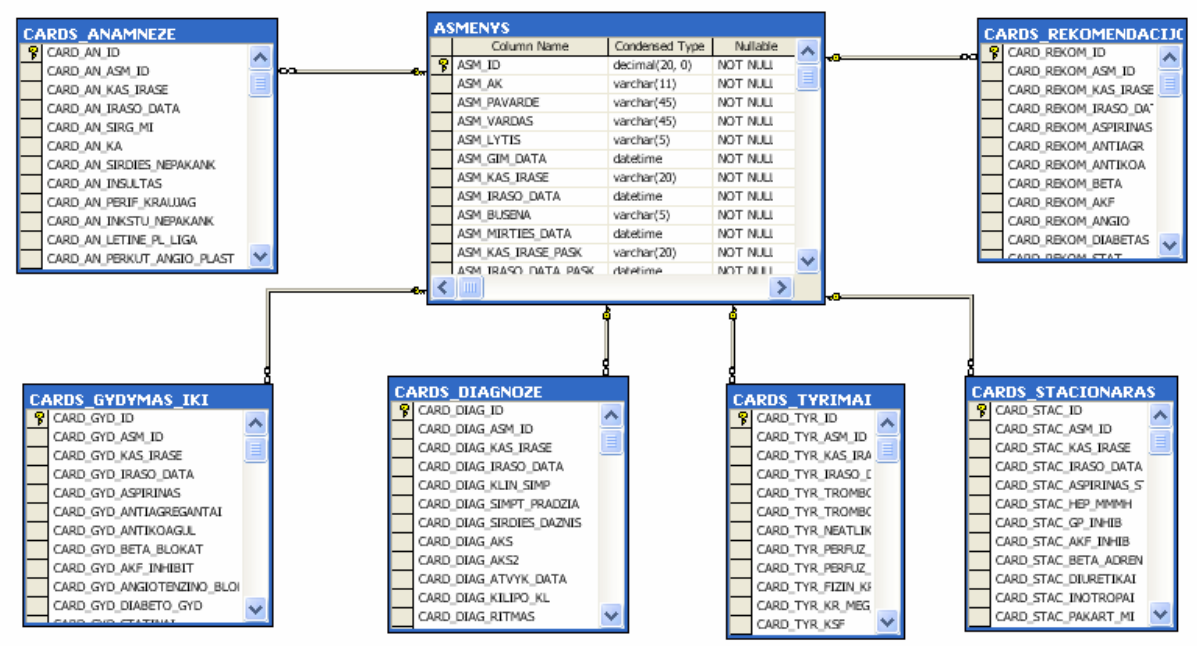
Pranešimų administravimui reikalingos lentelės ir ryšiai tarp jų:



11 pav. Pranešimų lentelės

Kiekvienas sistemos vartotojas gali siųsti arba gauti neribotą kiekį pranešimų, kurie saugomi lentelėje „Pranesimai“. Lentelė „Vartotojo_pranesimai“ padeda susieti sistemos vartotoją su konkrečiu pranešimu, bei keisti pranešimų statusą.

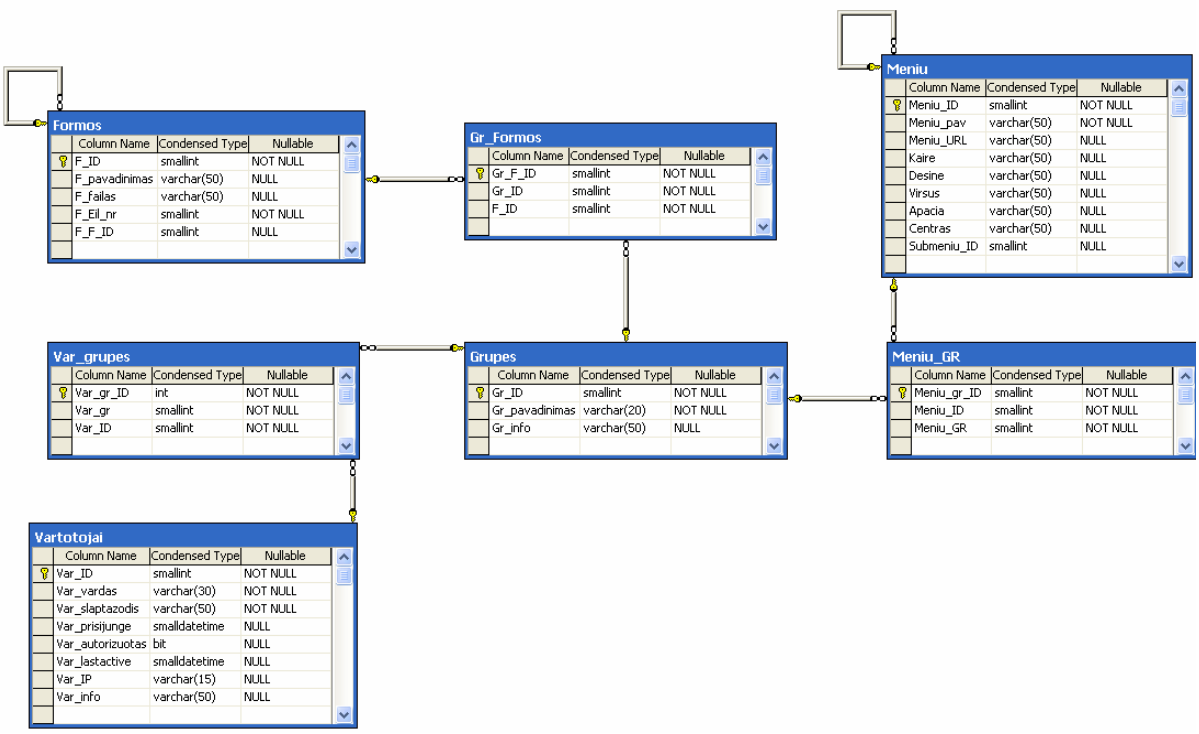
CARDS administravimui reikalingos lentelės ir ryšiai tarp jų:



12 pav. CARDS lentelės

CARDS lentelėse saugomi statistinei analizei svarbūs duomenys: specifiniai paciento anamnezės, tyrimų, diagnozės, rekomendacijų gydymui bei guldymo metu sukaupti duomenys.

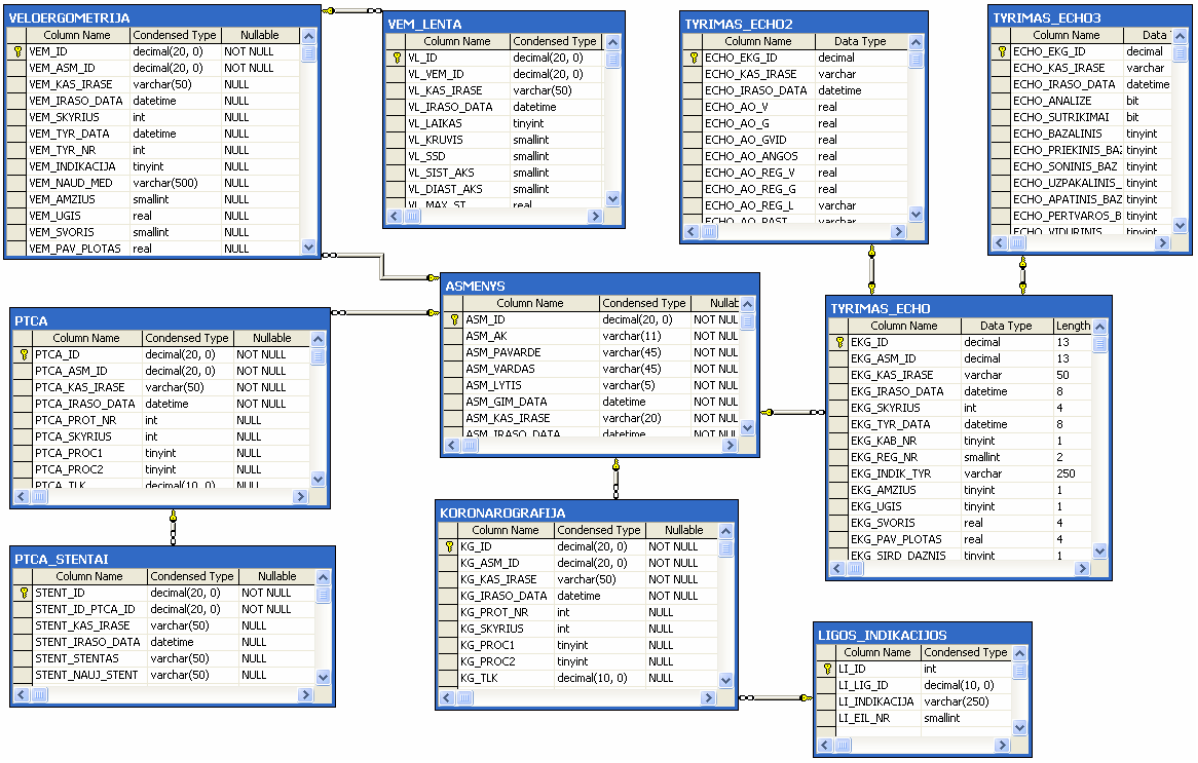
Vartotojų administravimui reikalingos lentelės ir ryšiai tarp jų:



13 pav. Vartotojų administravimas

Lentelėse „Meniu“, „Formos“ saugomi sistemos resursai. Lentelė „Grupes“ skirta saugoti sistemos vartotojų rolėms, pavyzdžiui, administratorius, kardiologas. Lentelės „Gr_Formos“ ir „Meniu_Gr“ skirtos sistemos resursų administravimui, t.y. kiekvienai vartotojų grupei yra priskiriamos tam tikros formos ar meniu. Kiekvienas sistemos vartotojas privalo priklausyti bent vienai vartotojų grupei. Su tuo susiję duomenys saugomi lentelėje „Var_grupes“. Vartotojo priklausymas grupei užtikrina, kad jis galės naudotis visais tai grupei priskirtais sistemos resursais.

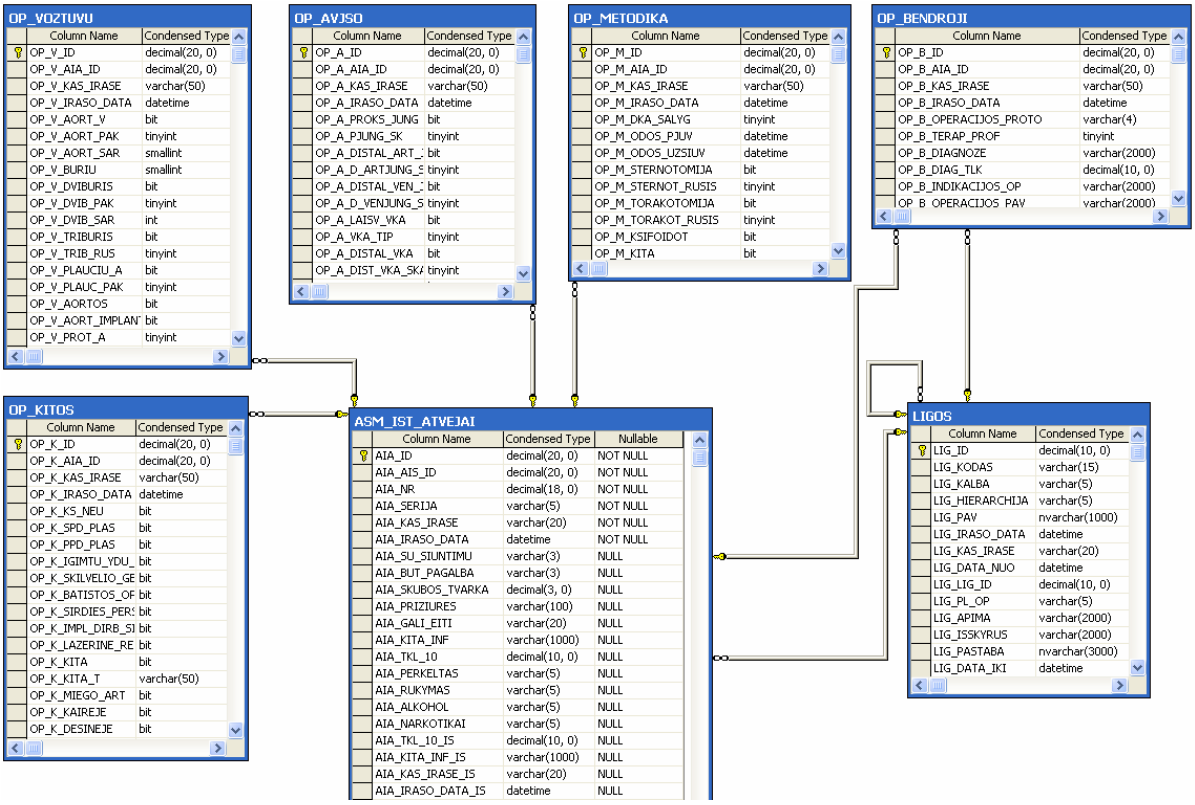
Tyrimų administravimui reikalingos lentelės ir ryšiai tarp jų:



14 pav. Tyrimų lentelės

Kol kas sistemoje realizuotas tyrimų echokardiografija (lentelės „Tyrimas_echo“, „Tyrimas_echo2“, „Tyrimas_echo3“), koronarografija (lentelės „Koronarografija“, „Ligos“, „Ligos_indikacijos“), veloergometrija (lentelės „Veloergometrija“, „Vem_lenta“) bei ptca („ptca“, „ptca_stentai“) duomenų administravimas. Kiekvienas tyrimas yra susietas su lentele „Asmenys“. Tai reiškia, kad tyrimo duomenų registravimas gali būti atliekamas net jeigu pacientas nėra paguldytas į stacionarą.

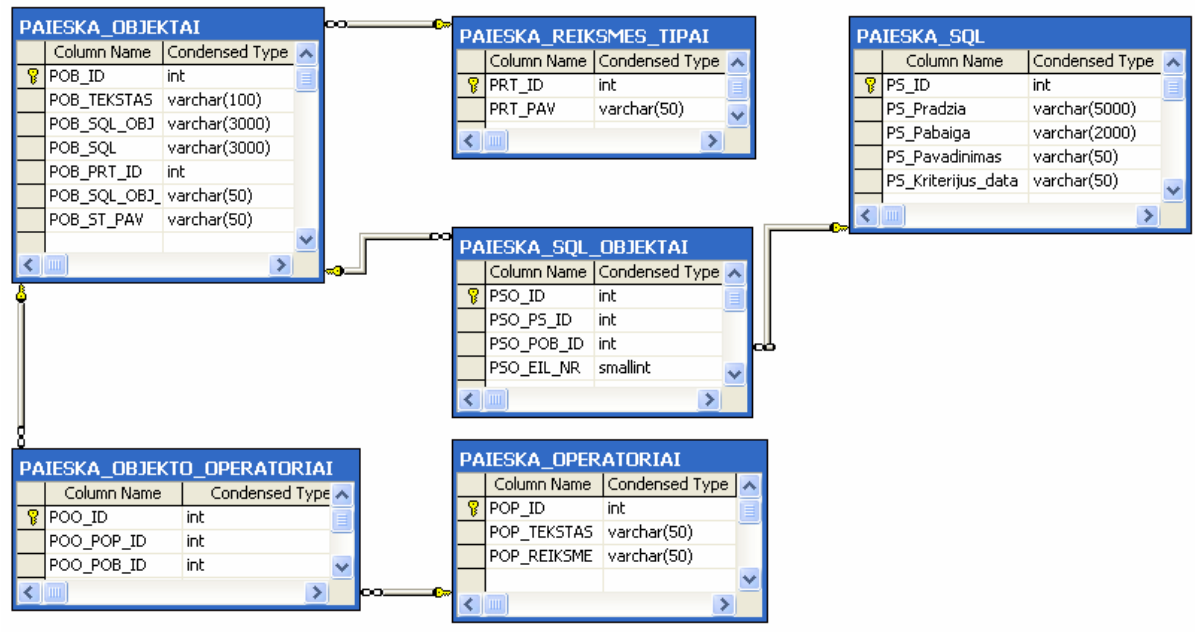
Operacijų administravimui reikalingos lentelės ir ryšiai tarp jų:



15 pav. Operacijų lentelės

Operacijų lentelėse saugomi operacijos bendrosios dalies, metodikos, vožtuvų chirurgijos, AVJSO bei kiti su chirurgija susiję duomenys.

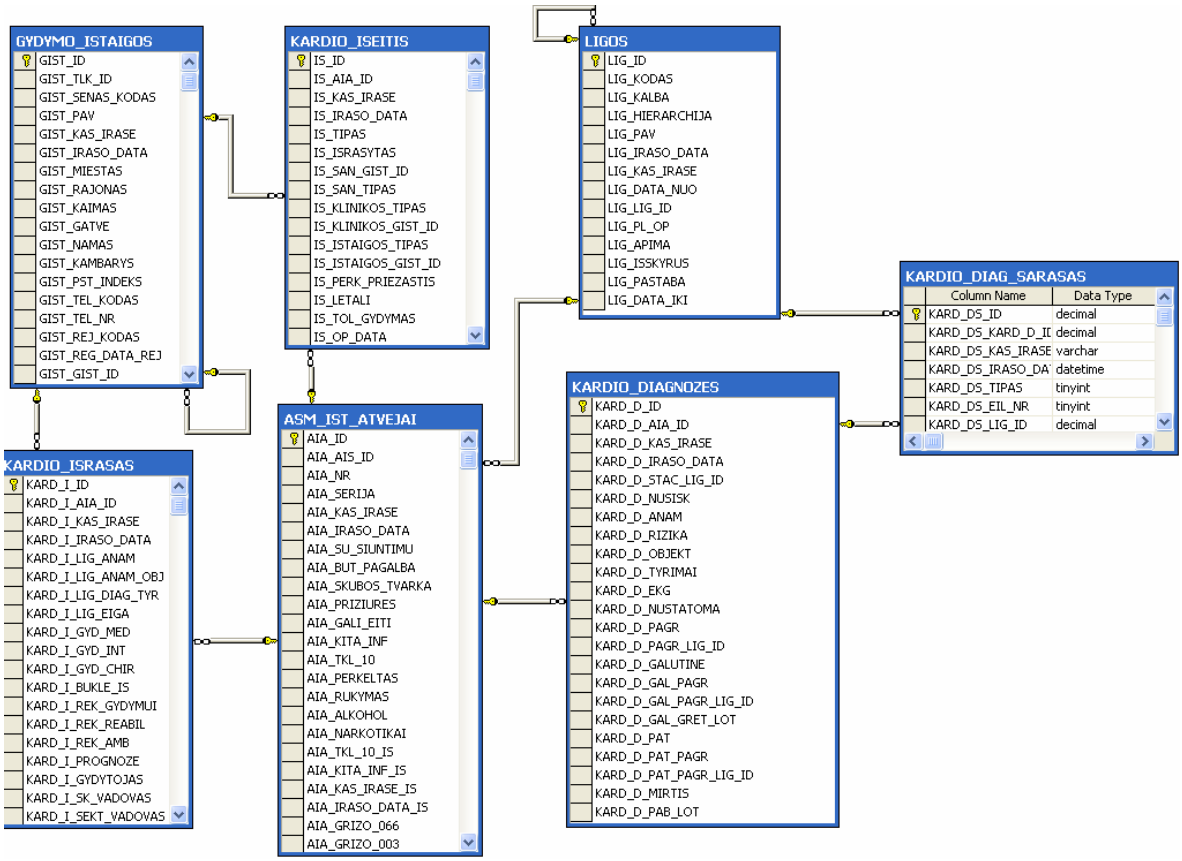
Paieškos organizavimui reikalingos lentelės ir ryšiai tarp jų:



16 pav. Universalios paieškos šablono lentelės

Lentelėje „Paieska_operatoriai“ saugomi loginio palyginimo operatoriai, pavyzdžiui, „>“. Lentelėje „Paieska_reiksmes_tipai“ nurodomas lyginamosios reikšmės šablonas, pavyzdžiui, ar tai datos tipas, ar realus skaičius. Lentelėje „Paieska_objektai“ saugomi visi laukai pagal kuriuos bus galima atlikti dinaminę paiešką. „Paieska_sql“ konkretus užklausos sakinys, kuris atrenka atitinkamus laukus, pagal nurodytus paieškos kriterijus.

Kardiologijos duomenų kaupimui, diagnozės ir išrašo formavimui reikalingos lentelės ir ryšiai tarp jų:



17 pav. Kardiologijos lentelės

KMU Širdies centro informacijos sistema pagrindinį dėmesį skiria būtent kardiologinių duomenų kaupimui, todėl kardiologijos duomenų kaupimo modulis išsiskiria duomenų lentelių kiekiu. Čia fiksuojami anamnezės, ankstesnių tyrimų ir gydymo, nusiskundimų, kardio status, išeminės ligos, arterinės hipertenzijos, laidumo sutrimikų, tachiaritmijos, rizikų, širdies ydų bei kitų širdies ligų duomenys. Paveikslėlyje parodytos tik tos lentelės, kuriose kaupiami duomenys susiję su diagnozės formavimu bei paciento išrašymu.

4.6. Sistemos architektūra

KMU Širdies centro informacijos sistema realizuota trijų lygių architektūra, kurių kiekvienas detaliai aprašytas.

Duomenų kaupimo lygis

Šiame lygyje saugomi sistemos duomenys sąryšiniame pavidale. KMU Širdies centro informacijos sistemoje naudojama Microsoft SQL Server 2000 sąryšinė duomenų bazių valdymo sistema. Manipuliavimas duomenimis atliekamas serverio procedūrų pagalba.

Veiklos logikos lygis

Veiklos logikos lygį sudaro šie komponentai: SQL_DB, KMUK_GULD, KMUK_ASM, KMUK_USER, KMUK_ADMIN, PRANESIMAI, KARD_IS, KARD_GULD, KARD_ASM, KARD_PAIESKA ir KARD_KARDIOLOGIJA.

SQL_DB - vienintelis komponentas, turintis ryšį su duomenų baze ir skirtas duomenų manipuliavimui. Visi žemiau išvadinti komponentai naudoja SQL_DB komponentą komunikacijai su duomenų baze.

KMUK_GULD - komponentą sudaro klasė skirta paciento guldymo duomenų kaupimui registratūroje ir specifiniai metodai tų duomenų panaudojimui mūsų sistemoje.

KMUK_ASM - komponentą sudaro klasė skirta paciento duomenų kaupimui registratūroje ir specifiniai metodai tų duomenų panaudojimui mūsų sistemoje.

KMUK_USER - komponentą sudaro klasė skirta sistemos vartotojo duomenų kaupimui ir specifiniai metodai tų duomenų panaudojimui.

KMUK_ADMIN - komponentą sudaro klasės skirtos sistemos resursų bei vartotojų administravimui ir specifiniai metodai tų duomenų panaudojimui.

PRANESIMAI - komponentą sudaro klasės skirtos pranešimams tarp sistemos vartotojų administruoti ir specifiniai metodai tų duomenų panaudojimui.

KARD_IS - komponentą sudaro klasė, kurioje yra metodai, naudojami ataskaitų formavimui, spausdinimui.

KARD_GULD - komponentą sudaro klasė skirta paciento guldymo duomenų kaupimui ir specifiniai metodai tų duomenų panaudojimui.

KARD_ASM - komponentą sudaro klasė skirta paciento anketinių duomenų kaupimui ir specifiniai metodai tų duomenų panaudojimui.

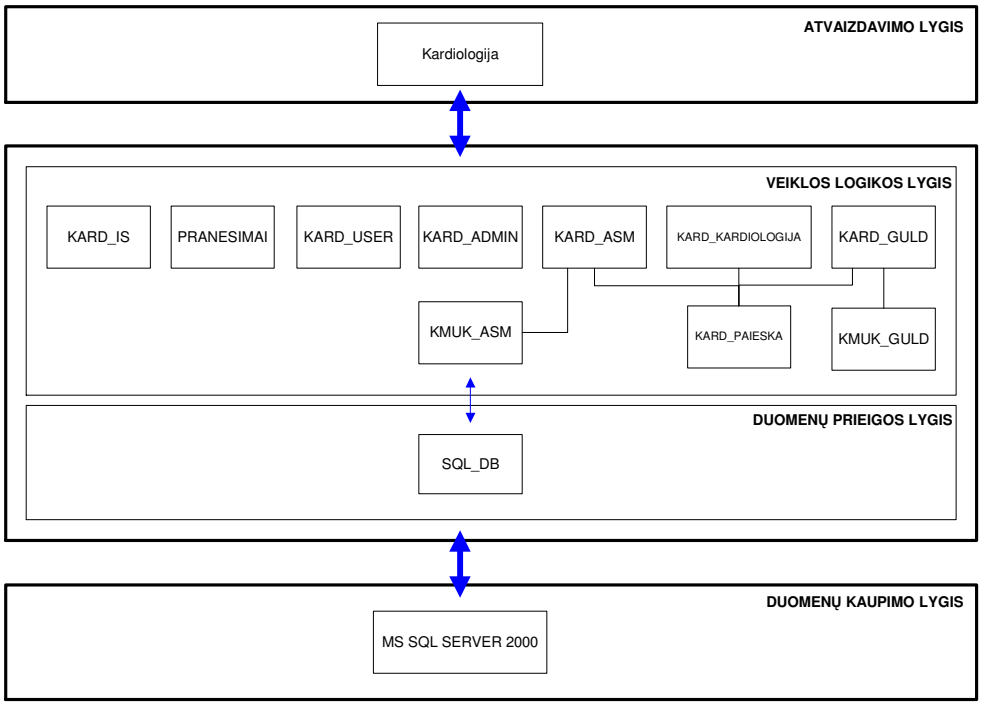
KARD_PAIESKA - komponentą sudaro dinaminei paieškai reikalingos klasės bei metodai.

KARD_KARDIOLOGIJA – kardiologijos, tyrimų, operacijų, CARDS klasės bei metodai.

Atvaizdavimo lygis

Lygį sudaro komponentas Kardiologija, kuriame saugomi galutiniam vartotojui matomi puslapiai (bylos su plėtiniu .aspx) ir naudojami šablonai (bylos su plėtiniu .ascx).

Komponentų pasiskirstymas trijų lygių architektūros modelyje:

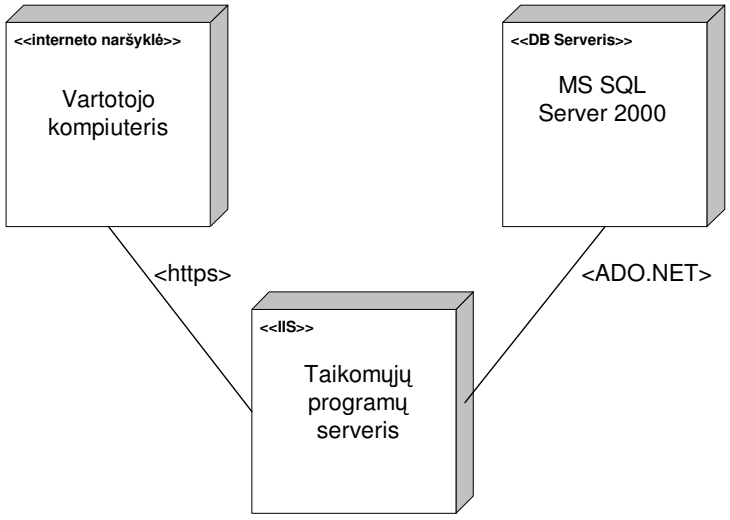


18 pav. Sistemos komponentų diagrama

4.7. Išdėstymo vaizdas

Realizuotas programinės įrangos kodas patalpinamas taikomųjų programų serveryje. Kiekvienas vartotojas su sistema bendrauja interneto naršyklės pagalba. Komunikacija tarp vartotojo kompiuterio ir taikomųjų programų serverio vyksta https protokolo pagalba. Taikomųjų programų serveris ADO.NET technologijos pagalba manipuliuoja duomenimis, esančiais MS SQL Server 2000 duomenų bazių valdymo sistemoje.

Sistemos išdėstymo vaizdas:



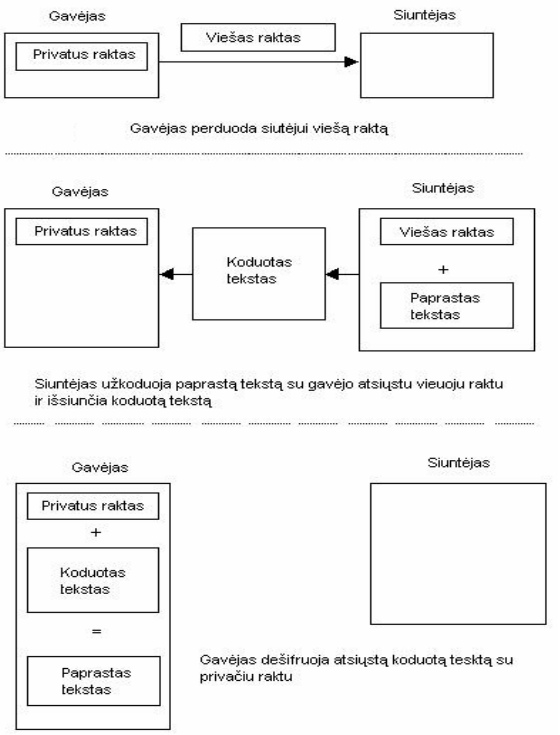
19 pav. Išdėstymo vaizdas

4.8. KMU Širdies centro informacijos sistemos projektiniai sprendimai

4.8.1. Padidintas sistemos saugumas

KMU Širdies centro informacijos sistema pasižymi padidintu saugumu. Realizuojant sistemą buvo panaudotas SSL protokolas, užtikrinantis koduotą duomenų perdavimą tarp interneto naršyklės ir taikomųjų programų serverio. Gaunamų duomenų dešifravimui, SSL protokolas naudoja 128 bitų privatų raktą. Tai reiškia, kad privačiam raktui yra suteikiama viena konkreti reikšmė iš 2^{128} galimų variantų. Taigi norint nulaužti sistemą, reikia atspėti kodą, kuris turi net 2^{128} galimų variantų.

SSL protokolu paremtos sistemos veikimo principas:



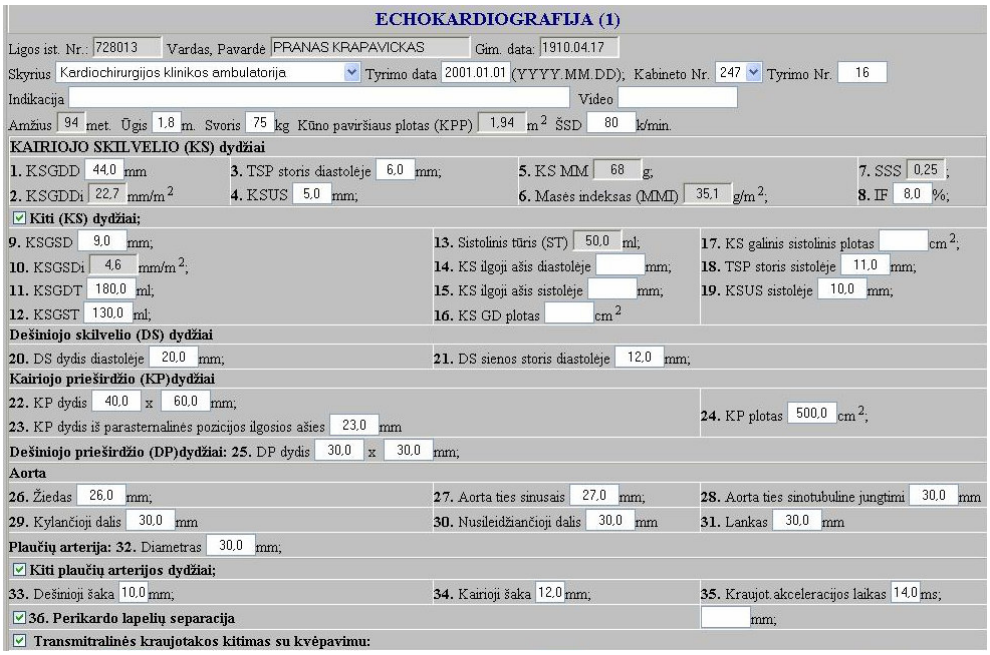
20 pav. Sistemos, naudojančios SSL protokolą, veikimo principas

SSL protokolas, naudojantis privataus rakto kodavimo technologiją, užtikrina siunčiamų ir gaunamų duomenų kodavimą ir serverio bei kliento autentifikaciją kiekvienam TCP/IP prisijungimui.

4.8.2. Dideli duomenų kiekiai patogioje sąsajoje

Sistemoje registruojama informacija yra labai įvairiapusė, t.y. registruojami dideli informacijos kiekiai. Tarkime vienoje formoje registruojamų parametrų kiekis svyruoja nuo 10 iki 160 atributų, todėl buvo iškilusi patogios sąsajos problema. Buvo nuspręsta sukurti tokį dizainą, kuriame elementų išdėstymas būtų kiek įmanoma labiau panašus į popierinių dokumentų analogus. Be to, kuriant dizainą buvo pastoviai palaikomas ryšys su būsimuoju vartotoju, t. y. vartotojui buvo pateikiamos tarpinės formų realizacijos ir laukiama pastabų iš

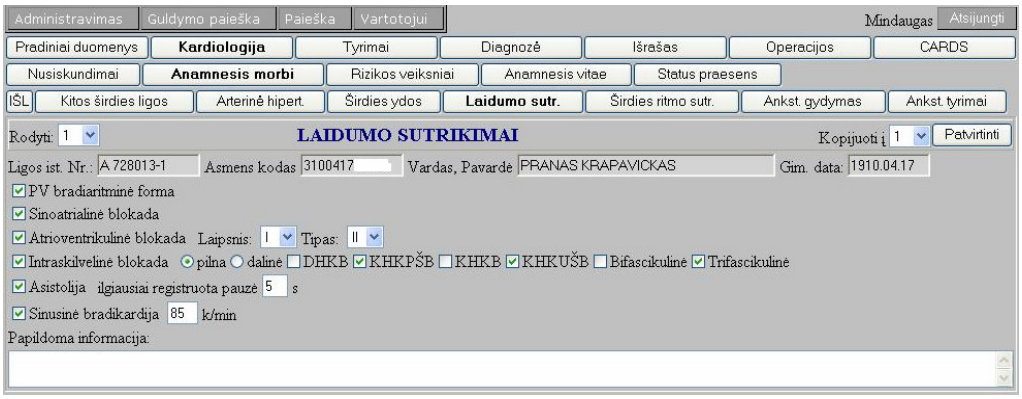
jo. Tokiu būdu, kelių iteracijų pagalba, buvo sukurtas sistemos dizainas, pilnai atitinkantis vartotojo reikalavimus.



21 pav. Echokardiografijos tyrimo langas

4.8.3. Koduotas duomenų saugojimas

Informacijos sistemose skirtose medicinai kaupiami duomenys skirstomi į „koduotus“ ir „laisvo teksto“. Pirmuoju atveju saugomas laukas įrašomas į duomenų bazę kaip atskiras atributas. Tokio metodo privalumas tai, kad kiekvienas elementas, kuris yra saugomas duomenų bazėje, gali būti panaudotas paieškose ar duomenų užklausose ateityje. Tokio tipo duomenų kaupimas vertingas toms gydymo įstaigoms, kurios atlieka tyrimus siekdamos nustatyti, kiek pacientų serga vienokia ar kitokia liga, pasižymi specifine sveikatos būkle. Šio metodo trūkumas yra tas, kad duomenų registravimas yra imlus laikui, t. y. gydytojui tenka daug kartų pažymėti/pasirinkti formos laukelius, kol užpildomas visas dokumentas, kai tuo tarpu „laisvo teksto“ sistemose duomenų registravimas yra paprastas ir įprastas procesas, tačiau tokių sistemų sukauptų duomenų pakartotinis panaudojimas yra labai ribotas.



22 pav. Koduotų duomenų pildymas

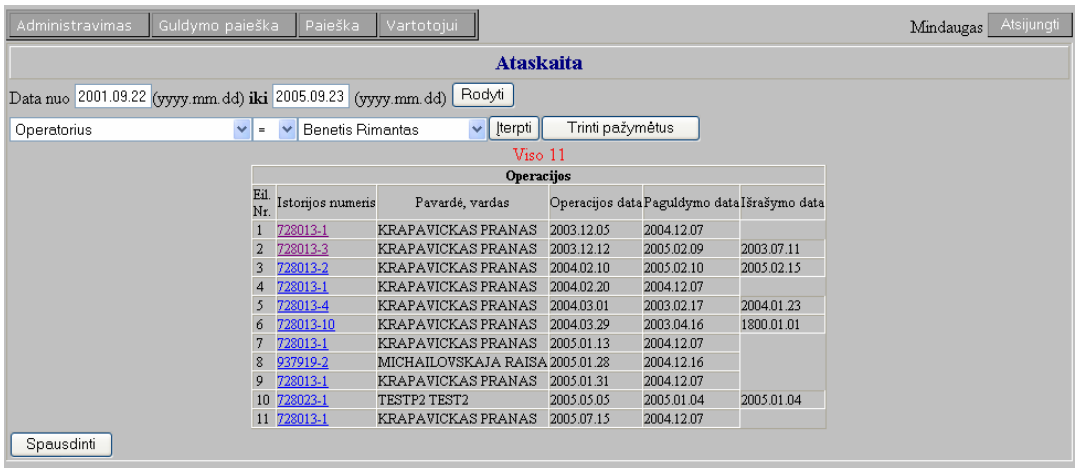
4.8.4. Dinaminė paieška

Pagrindinis šios realizacijos privalumas yra tas, jog neribojamas paieškos kriterijų kiekis. Vartotojas pats pasirenka pagal kokius kriterijus jis nori atlikti paiešką, o didindamas paieškos kriterijų skaičių, gali ją detalizuoti iki laukiamo rezultato.

Kiekvienas paieškos kriterijus yra susietas su atitinkamos formos atitinkamu atributu. Tarkime operacijų paieškos lange paieškos kriterijus „I asistentas“ yra susietas su formos „Operacijos bendroji dalis“ atributu „I asistentas“. Reikėtų pabrėžti, jog priklausomai nuo formos atributo tipo, yra suformuojamas atitinkamo tipo paieškos kriterijus. Tarkime, jeigu formoje atributas buvo teksto įvedimo laukelis, tai ir paieškos formoje jis bus analogiško tipo.

Pasirinkus atitinkamą paieškos kriterijų, suformuojama galimų operatorių aibė, kuri priklauso nuo paieškos kriterijaus tipo, pavyzdžiui, jeigu paieškos kriterijus yra sąrašo tipo, tai operatorių aibė suformuojama tokia {=, <>}, jeigu paieškos kriterijus yra sveikas skaičius, tai operatorių aibė - {=, <>, >=, <=, >, <}.

Operacijų paieškos langas:



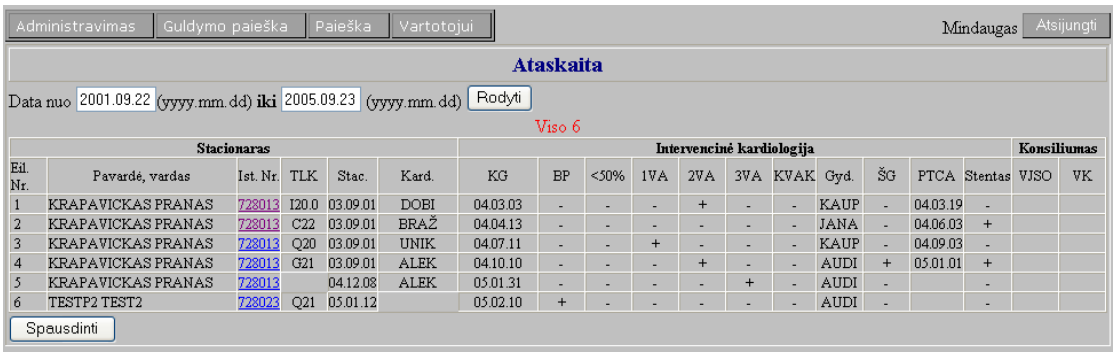
23 pav. Operacijų paieškos langas

4.8.5. Įvairiapusis sukauptų duomenų panaudojimas

Kardiologinių tyrimų surinkimo bei analizės programų sistema formuoja dviejų tipų ataskaitas: apibendrintas, kurių formavimui panaudojami visi sistemoje sukaupti duomenys, ir konkrečiam pacientui pritaikytas ataskaitas.

Apibendrintose ataskaitose pateikiama tik kritinė informacija. Norint sužinoti detalesnę informaciją, reikia pasirinkti konkretų pacientą iš suformuoto rezultatų sąrašo.

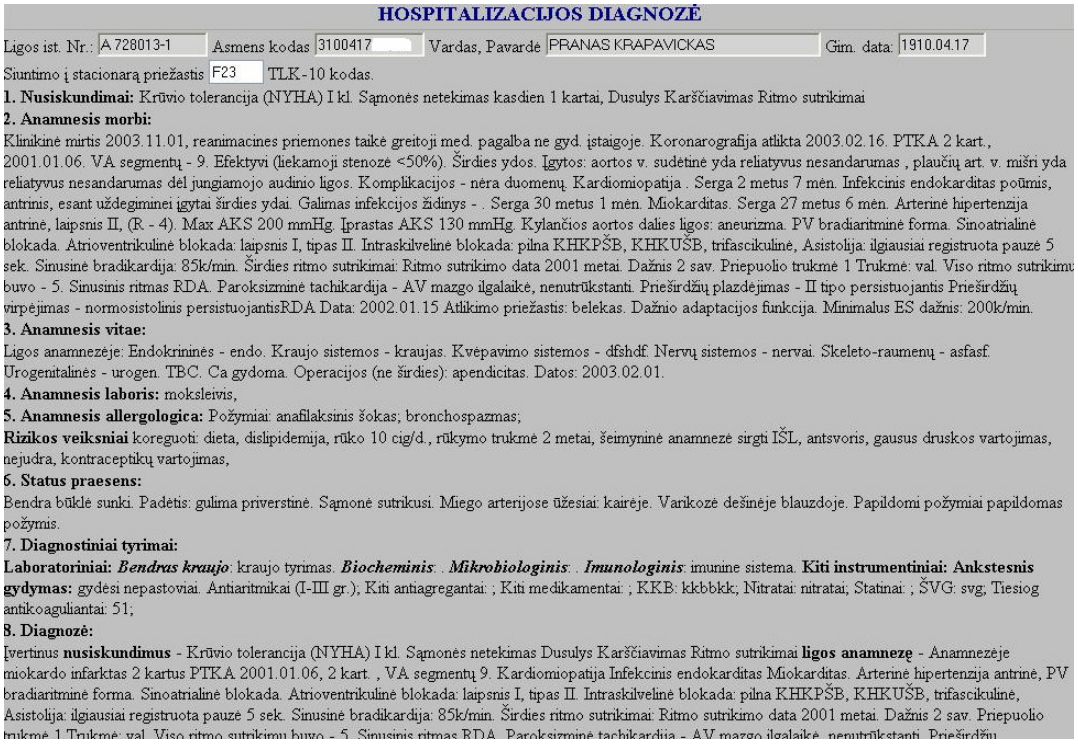
Apibendrintos ataskaitos pavyzdys, kurioje pateikiamas sąrašas pacientų, kuriems nurodytu laiko tarpu buvo atlikta intervencinė diagnostika:



Stacionaras						Intervencinė kardiologija										Konsiliumas	
Eil. Nr.	Pavardė, vardas	Ist. Nr.	TLK	Stac.	Kard.	KG	BP	<50%	1VA	2VA	3VA	KVAK	Gyd.	ŠG	PTCA	Stentas	VJSO	VK
1	KRAPAVICKAS PRANAS	728013	I20.0	03.09.01	DOBI	04.03.03	-	-	-	+	-	-	KAUP	-	04.03.19	-		
2	KRAPAVICKAS PRANAS	728013	C22	03.09.01	BRAŽ	04.04.13	-	-	-	-	-	-	JANA	-	04.06.03	+		
3	KRAPAVICKAS PRANAS	728013	Q20	03.09.01	UNIK	04.07.11	-	-	+	-	-	-	KAUP	-	04.09.03	-		
4	KRAPAVICKAS PRANAS	728013	G21	03.09.01	ALEK	04.10.10	-	-	-	+	-	-	AUDI	+	05.01.01	+		
5	KRAPAVICKAS PRANAS	728013		04.12.08	ALEK	05.01.31	-	-	-	-	+	-	AUDI	-		-		
6	TESTP2 TEST2	728023	Q21	05.01.12		05.02.10	+	-	-	-	-	-	AUDI	-		-		

24 pav. Intervencinės diagnostikos suvestinės langas

Antrojo tipo ataskaitos yra formuojamos tik konkrečiam pacientui. Šiose ataskaitose pateikiama visa, su dalykine sritimi susijusi, užpildyta informacija apie paciento sveikatos būklę. Tarkime hospitalizacijos diagnozės ataskaita yra formuojama iš duomenų užpildytų kardiologijos modulyje:



HOSPITALIZACIJOS DIAGNOZĖ

Ligos ist. Nr.: A 728013-1 Asmens kodas 3100417 Vardas, Pavardė PRANAS KRAPAVICKAS Gim. data: 1910.04.17

Siuntimo į stacionarą priežastis F23 TLK-10 kodas.

1. Nusiskundimai: Krūvio tolerancija (NYHA) I kl. Sąmonės netekimas kasdien 1 kartai, Dusulys Karščiavimas Ritmo sutrikimai

2. Anamnesis morbi:

Klinikinė mirtis 2003.11.01, reanimacines priemones taikė greitoji med. pagalba ne gyd. įstaigoje. Koronarografija atlikta 2003.02.16. PTKA 2 kart., 2001.01.06. VA segmentų - 9. Efektyvi (liekamoji stenozė <50%). Širdies ydos. Įgytos: aortos v. sudėtinė yda reliatyvus nesandarumas , plaučių art. v. mišri yda reliatyvus nesandarumas dėl jungiamojo audinio ligos. Komplikacijos - nėra duomenų. Kardiomiopatija . Serga 2 metus 7 mėn. Infekcinis endokarditas poūmis, antrinis, esant uždegiminei įgytai širdies ydai. Galimas infekcijos židinys - . Serga 30 metus 1 mėn. Miokarditas. Serga 27 metus 6 mėn. Arterinė hipertenzija antrinė, laipsnis II, (R - 4). Max AKS 200 mmHg. Įprastas AKS 130 mmHg. Kylančios aortos dalies ligos: aneurizma. PV bradiaritminė forma. Sinoatrialinė blokada. Atrioventrikulinė blokada: laipsnis I, tipas II. Intraskilvelinė blokada: pilna KHKPŠB, KHKUŠB, trifascikulinė, Asistolija: ilgiausiai registruota pauzė 5 sek. Sinusinė bradikardija: 85k/min. Širdies ritmo sutrikimai: Ritmo sutrikimo data 2001 metai. Dažnis 2 sav. Priepuolio trukmė 1 Trukmė: val. Viso ritmo sutrikimų buvo - 5. Sinusinis ritmas RDA. Paroksizminė tachikardija - AV mazgo ilgalaikė, nenutrūkstanti. Priešlaikinis Priešlaikinių virpėjimas - normosistolinis persistuojantisRDA Data: 2002.01.15 Atlikimo priežastis: belekas. Dažnio adaptacijos funkcija. Minimalus ES dažnis: 200k/min.

3. Anamnesis vitae:

Ligos anamnezėje: Endokrininės - endo. Kraujo sistemos - kraujas. Kvėpavimo sistemos - dfshdf. Nervų sistemos - nervai. Skeleto-raumenų - asfasf. Urogenitalinės - urogen. TBC. Ca gydoma. Operacijos (ne širdies): apendicitas. Datos: 2003.02.01.

4. Anamnesis laboris: moksleivis,

5. Anamnesis allergologica: Požymiai: anafilaksinis šokas; bronchospazmas;

Rizikos veiksniai koreguoti: dieta, dislipidemija, rūko 10 cig/d., rūkymo trukmė 2 metai, šeimyninė anamnezė sirgti IŠL, antsvoris, gausus druskos vartojimas, nejudra, kontraceptikų vartojimas,

6. Status praesens:

Bendra būklė sunki. Padėtis: gulima priverstinė. Sąmonė sutrikusi. Miego arterijose ūžesiai: kairėje. Varikozė dešinėje blauzdoje. Papildomi požymiai papildomas požymis.

7. Diagnostiniai tyrimai:

Laboratoriniai: ***Bendras kraujo***: kraujo tyrimas. ***Biocheminis***: . ***Mikrobiologinis***: . ***Imunologinis***: imunine sistema. **Kiti instrumentiniai: Ankstesnis gydymas:** gydėsi nepastoviai. Antiaritmikai (I-III gr.); Kiti antiagregantai: ; Kiti medikamentai: ; KKB: kkbbkk; Nitratai: nitratai; Statinai: ; ŠVG: svg; Tiesiog antikoaguliantai: 51;

8. Diagnozė:

Įvertinus **nusiskundimus** - Krūvio tolerancija (NYHA) I kl. Sąmonės netekimas Dusulys Karščiavimas Ritmo sutrikimai **ligos anamnezę** - Anamnezėje miokardo infarktas 2 kartus PTKA 2001.01.06, 2 kart. , VA segmentų 9. Kardiomiopatija Infekcinis endokarditas Miokarditas. Arterinė hipertenzija antrinė, PV bradiaritminė forma. Sinoatrialinė blokada. Atrioventrikulinė blokada: laipsnis I, tipas II. Intraskilvelinė blokada: pilna KHKPŠB, KHKUŠB, trifascikulinė, Asistolija: ilgiausiai registruota pauzė 5 sek. Sinusinė bradikardija: 85k/min. Širdies ritmo sutrikimai: Ritmo sutrikimo data 2001 metai. Dažnis 2 sav. Priepuolio trukmė 1 Trukmė: val. Viso ritmo sutrikimų buvo - 5. Sinusinis ritmas RDA. Paroksizminė tachikardija - AV mazgo ilgalaikė, nenutrūkstanti. Priešlaikinių

25 pav. Hospitalizacijos diagnozės ataskaita

4.8.6. Automatizuotas duomenų pildymas

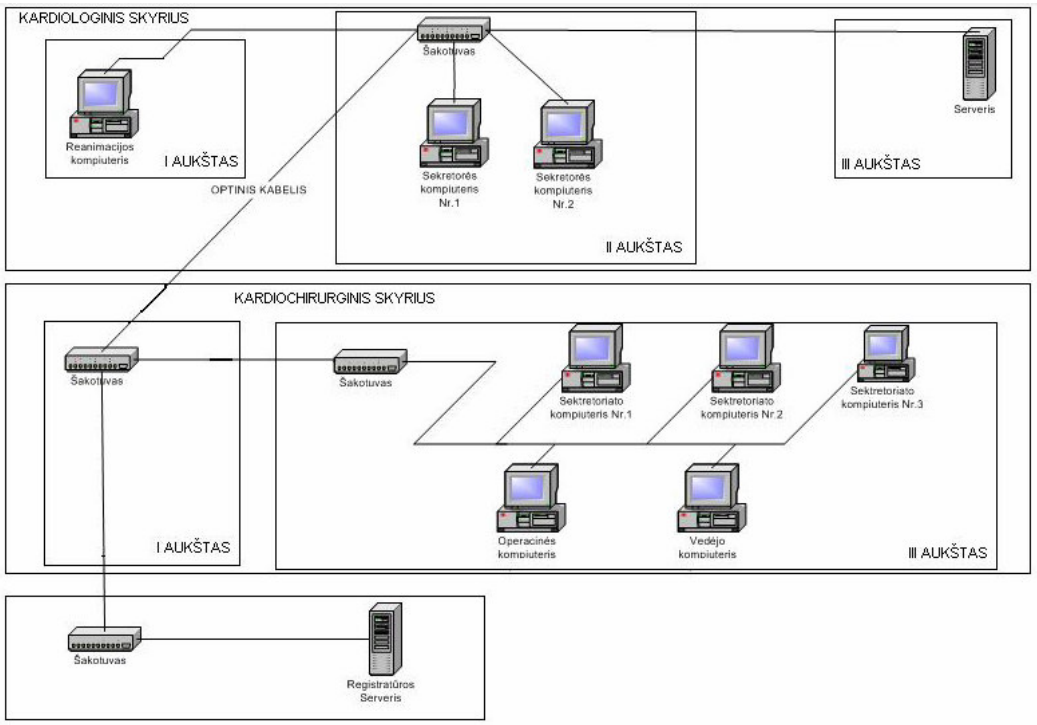
CARDS modulyje kaupiama informacija, kuri turėtų būti panaudota statistinei analizei. CARDS formose yra atrinkti tik tie laukai, kurie yra reikšmingi statistinei analizei. Daugumos tų laukų užpildymas yra automatizuotas, siekiant išvengti papildomo darbo suvedinėjant atitinkamus duomenis, o kartu ir išvengiant galimų pildymo klaidų.

Pirmiausia pagal tam tikrus nustatytus kriterijus sistema automatiškai atrenka paciento guldymą. Vėliau vartotojui paspaudus mygtuką „Įterpti“, raudona žvaigždute pažymėti laukai užpildomi atitinkamais duomenimis. Užvedus pelės kursorių ant atitinkamo lauko pavadinimo, parodoma iš kokios formos kokio atributo duomenys buvo priskirti šiam laukui.

26 pav. CARDS anamnezės duomenų automatizuotas pildymas

4.9. Sistemos eksploatavimas

KMU Širdies centro informacijos sistema šiuo momentu yra realiai naudojama. Šiuo metu vietiniame tinkle yra sukurtos aštuonios darbo vietos, kuriose galima naudotis sistema. Tiesa, sukurta sistema yra prieinama iš bet kurio vartotojo kompiuterio, turinčio internetinį ryšį. Vietinio tinklo struktūros diagrama:



27 pav. KMU Širdies centro vietinio tinklo struktūra

5. Efektyvaus manipuliavimo duomenimis informacijos sistemose skirtose medicinai tyrimas

Priežasčių kodėl sistema neefektyviai panaudoja sukauptus duomenis gali būti labai daug. Viena iš jų – tai neoptimaliai parašytas sistemos kodas. Dideli kiekiai kintamųjų, kurių dalis yra nepanaudojama viso programos gyvavimo ciklo metu, bereikalingi ciklai, klaidingas tam tikrų objektų naudojimas - neigiamai įtakoja sistemos veikimą. Netgi pati naujausia techninė įranga gali būti bejėgė, siekiant padidinti sistemos veikimo charakteristikas. Be to, laikui bėgant, techninė įranga sensta, duomenų kiekiai, sukaupti duomenų bazėje, didėja, o sistemos veikimo charakteristikos - prastėja. Jeigu aplikacija pritaikyta darbui internete, dar pridėkime tinklo apkrautumo, vėlinimo problemas, ir sistemos vartotojui gali susidaryti įspūdis, jog sistema nebeveikia arba ji veikia nepakankamai greitai.

Taigi, šio tyrimo esmė yra atskleisti efektyvaus manipuliavimo duomenimis galimybes, panaudojant ADO.NET technologijos duomenų prieigos komponentus, serverio procedūras bei spartinančiąją atmintinę. Eksperimento metu buvo tiriama, kurį iš metodų naudoti kreipiniams į duomenų bazę: serverio procedūras ar tiesioginius SQL kreipinius, kaip optimizuoti duomenų manipuliaciją, panaudojant spartinančiąją atmintinę, kurį iš dviejų ADO.NET duomenų prieigos komponentų: DataSet ar SqlDataReader naudoti duomenų išgavimui, esant šioms situacijoms:

- Iš vienos duomenų lentelė išrenkamas vienas įrašas.
- Iš vienos duomenų lentelės išrenkama keletas įrašų.
- Iš vienos duomenų lentelės išrenkami visi įrašai.
- Iš keleto duomenų lentelių išrenkamas vienas įrašas.
- Iš keleto duomenų lentelių išrenkama keletas įrašų.
- Iš keleto duomenų lentelių išrenkami visi įrašai.

5.1. Serverio procedūrų ir tiesioginių SQL kreipinių panaudojimo tyrimas

Eksperimento tikslas – išsiaiškinti serverio procedūrų ir tiesioginių SQL kreipinių panaudojimo galimybes.

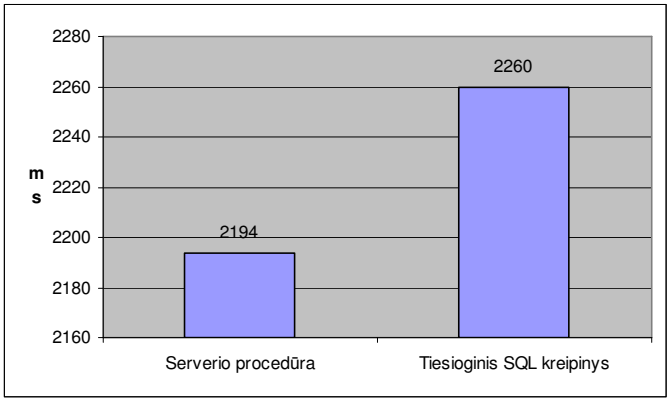
Eksperimentas buvo atliekamas su viena lentele, kurioje buvo 10 000 įrašų. Tam kad eksperimento rezultatai vizualiai labiau skirtųsi, buvo atliekama 50 iteracijų, t.y. ta pati užklausa ar serverio procedūra vykdoma 50 kartų, ir tik po to grąžinamas įvykdymo laikas milisekundėmis. Eksperimento metu buvo atlikta 12 bandymų, kurių rezultatai užfiksuoti lentelėje.

Užklausos vykdymo rezultatai, naudojant serverio procedūras ir tiesioginius SQL kreipinius:

3 lentelė. Serverio procedūrų ir tiesioginių SQL kreipinių tyrimo rezultatai

	1	2	3	4	5	6	7	8	9	10	11	12	Vidurkis
Serverio procedūra	2313	2103	2213	2203	2273	2153	2143	2133	2143	2273	2143	2233	**2194**
Tiesioginis SQL kreipinys	2293	2403	2253	2313	2243	2163	2253	2243	2403	2143	2153	2253	**2260**

Eksperimento rezultatai, kai iš vienos duomenų lentelės išrenkami reikalingi įrašai, naudojant serverio procedūras ir tiesioginius SQL kreipinius:



28 pav. Serverio procedūros ir tiesioginio SQL kreipinio vykdymo laikų palyginimo diagrama

Diagramoje matome, jog efektyviau yra naudoti serverio procedūras nei tiesioginius SQL kreipinius, nors užklausos įvykdymo laiko skirtumas tarp jų yra nežymus. Tiesa, reikėtų pažymėti tai, jog naudojant serverio procedūras, duomenų bazė gali optimizuoti duomenų prieigos planą ir naudoti spartinančiąją atmintį pakartotiniam panaudojimui, be to, serverio procedūrų naudojimas sumažina tinklo užimtumą.

5.2. Manipuliavimo duomenimis įvertinimas naudojant duomenis iš vienos DB lentelės

Eksperimentas buvo atliekamas su viena duomenų bazės lentele, kurioje buvo 10 000 įrašų. Duomenų išgavimui buvo panaudoti ADO.NET duomenų prieigos komponentai: DataSet ir SqlDataReader. Tam, kad eksperimento rezultatai vizualiai labiau skirtųsi, buvo atliekama 20 iteracijų, t.y. ta pati užklausa vykdoma 20 kartų, ir tik po to grąžinamas užklausos įvykdymo laikas milisekundėmis. Eksperimentas buvo atliekamas, vykdant užklausas, kurios grąžina vieną, keletą (mūsų atveju konkrečiai 109 įrašus) ir visus įrašus iš duomenų lentelės. Tokiu principu kiekvieno eksperimento metu buvo atlikta 12 bandymų, kurių rezultatai užfiksuoti lentelėse.

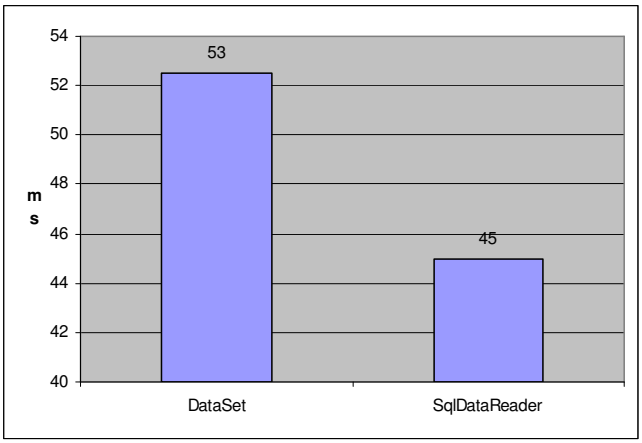
Situacija, kai reikiamas įrašas išrenkamas iš vienos duomenų lentelės, gali susidaryti užkraunant atitinkamo paciento duomenis į atitinkamą formą. Tokia situacija pasitaiko gana dažnai mūsų sistemoje, todėl svarbu ją įvertinti eksperimentiškai.

Eksperimento rezultatai, kai iš vienos duomenų lentelės išrenkamas vienas įrašas:

4 lentelė. Eksperimento rezultatai, kai iš vienos duomenų lentelės išrenkamas vienas įrašas

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Vidurkis**
DataSet	50	60	50	60	50	60	50	50	50	50	50	50	**52,5**
SqlDataReader	50	40	40	50	50	40	50	40	50	40	40	50	**45,0**

DataSet ir SqlDataReader komponentų panaudojimo galimybių palyginimo diagrama:



29 pav. Vieno įrašo išrinkimas iš vienos duomenų bazės lentelės

Diagramoje matyti, jog duomenų išgavimas, naudojant SqlDataReader komponentą buvo 1,17 karto efektyvesnis nei DataSet komponento pagalba.

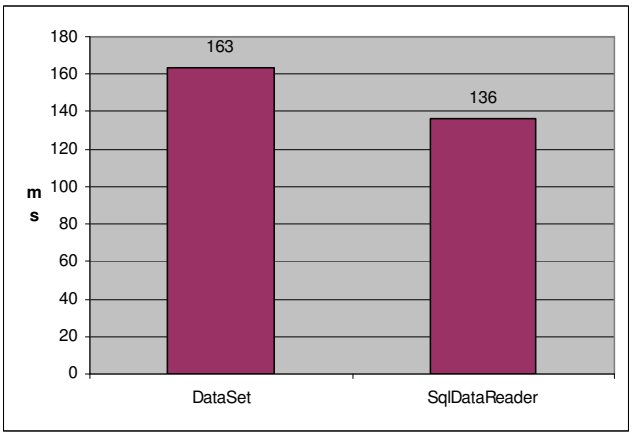
Atliekant pacientų ar gydymo įstaigų paiešką, gali susidaryti tokia situacija, kad pagal nurodytus kriterijus bus rasta daugiau nei vienas įrašas lentelėje. Taigi, svarbu nustatyti, kurį komponentą naudoti duomenų paieškai.

Eksperimento rezultatai, kai iš vienos duomenų lentelės išrenkama keletas įrašų, nurodant tam tikrą identifikatorių:

5 lentelė. Eksperimento rezultatai, kai iš vienos duomenų lentelės išrenkami kels įrašai

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Vidurkis**
DataSet	170	150	160	170	150	170	170	160	160	160	170	170	**163**
SqlDataReader	130	130	140	140	130	140	140	140	140	130	130	140	**136**

DataSet ir SqlDataReader komponentų panaudojimo galimybių palyginimo diagrama:



30 pav. Keleto įrašų išrinkimas iš vienos duomenų bazės lentelės

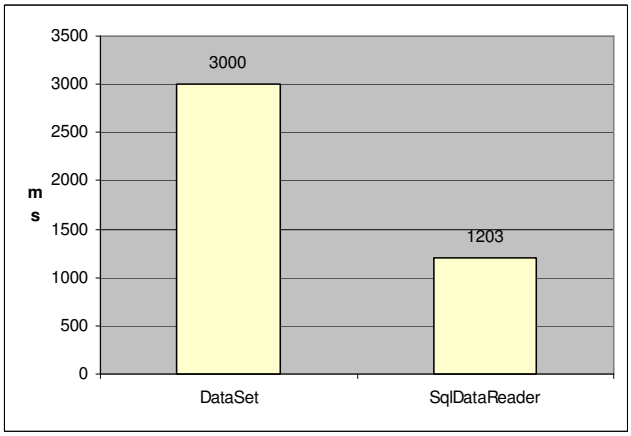
Jei konkretaus įrašo išgavimas iš DB lentelės, panaudojant SqlDataReader komponentą, buvo maždaug 1,17 karto efektyvesnis nei DataSet komponento pagalba, tai šiuo atveju SqlDataReader komponento panaudojimo efektyvumas yra labai panašus ir veikimo charakteristikomis DataSet komponentą lenkia 1,2 karto.

Galiausiai buvo atliktas eksperimentas, kai iš vienos duomenų lentelės išrenkami visi įrašai, ir gauti tokie eksperimento rezultatai:

6 lentelė. Eksperimento rezultatai, kai iš vienos duomenų lentelės išrenkami visi įrašai

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Vidurkis**
DataSet	3014	2914	2991	3154	2884	3324	2904	3004	2974	2824	2864	3154	**3000**
SqlDataReader	1221	1201	1221	1091	1231	1221	1221	1171	1231	1231	1171	1221	**1203**

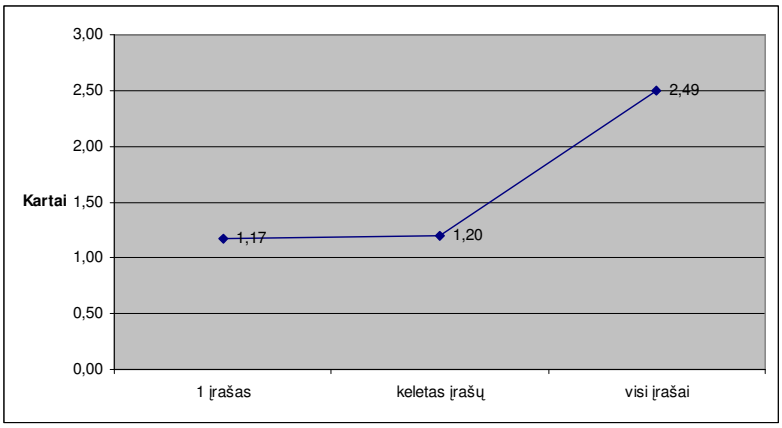
DataSet ir SqlDataReader komponentų panaudojimo galimybių palyginimo diagrama:



31 pav. Visų įrašų išrinkimas iš vienos duomenų bazės lentelės

Šįkart diagramoje matyti akivaizdus SqlDataReader komponento pranašumas prieš DataSet komponentą. SqlDataReader veikimo charakteristikos beveik 2,5 karto geresnės nei DataSet.

DataSet ir SqlDataReader komponentų santykinių greičių palyginimo diagrama parodo, kiek kartų efektyviau yra naudoti SqlDataReader komponentą, užklausų iš vienos duomenų lentelės vykdymui:



32 pav. DataSet ir SqlDataReader santykinių greičių palyginimo diagrama

5.3. Manipuliavimo duomenimis įvertinimas naudojant duomenis iš dviejų DB lentelių

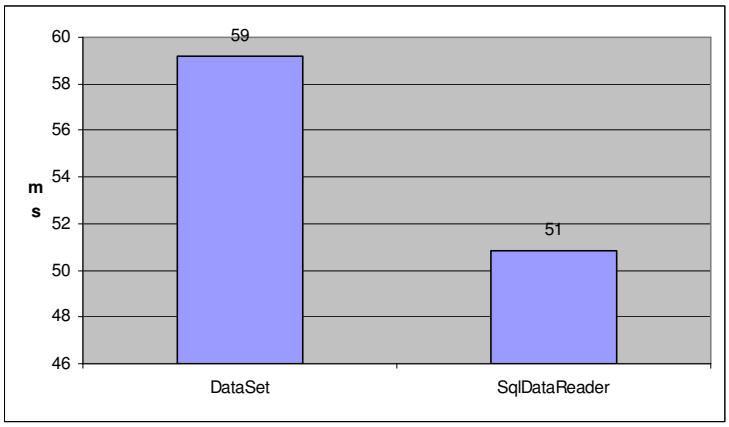
Šis eksperimentas buvo atliekamas su dviem duomenų lentelėm. Pirmojoje buvo 100 įrašų, antrojoje 10 000 įrašų. Duomenų išgavimui buvo panaudoti ADO.NET duomenų prieigos komponentai: DataSet ir SqlDataReader. Tam kad eksperimento rezultatai vizualiai labiau skirtųsi, buvo atliekama 20 iteracijų, t.y. ta pati užklausa vykdoma 20 kartų ir tik po to grąžinamas užklausos įvykdymo laikas milisekundėmis. Eksperimentas buvo atliekamas, vykdant užklausas, kurios grąžina vieną, keletą (mūsų atveju konkrečiai 109 įrašus) ir visus įrašus iš duomenų lentelių. Tokiu principu kiekvieno eksperimento metu buvo atlikta 12 bandymų, kurių rezultatai užfiksuoti lentelėse.

Gali susiklostyti tokia situacija, kad tarkime formoje reikia atvaizduoti ne tik duomenis iš paciento lentelės, bet ir tam tikro klasifikatoriaus reikšmę. Taigi, eksperimentas, kai iš dviejų duomenų lentelių išrenkamas vienas įrašas, gali įvertinti būtent tokią situaciją. Eksperimento rezultatai:

7 lentelė. Eksperimento rezultatai, kai iš dviejų duomenų lentelių išrenkamas vienas įrašas

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Vidurkis**
DataSet	100	60	50	50	60	60	50	50	50	60	60	60	**59,2**
SqlDataReader	70	50	50	50	50	50	50	50	40	50	50	50	**50,8**

DataSet ir SqlDataReader komponentų naudojimo galimybių palyginimo diagrama:



33 pav. Vieno įrašo išrinkimas iš keleto duomenų bazės lentelių

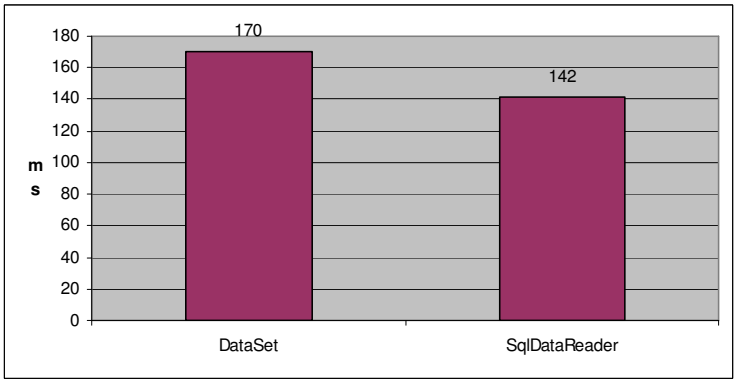
Skirtumas tarp užklausos įvykdymo laikų praktiškai vienodas, kaip ir atliekant analogišką eksperimentą su viena duomenų lentele - duomenų išgavimas efektyvesnis naudojant SqlDataReader komponentą. Tačiau reikia atkreipti dėmesį į tai, jog laiko sąnaudos išaugo tiek naudojant SqlDataReader, tiek DataSet komponentus. Abiem atvejais užklausa buvo vykdoma 1,13 karto ilgiau.

Pacientui atliktų operacijų, tyrimų paieškai ir peržiūrai gali prireikti duomenų, esančių keliose duomenų lentelėse, todėl svarbu, efektyviai išnaudoti ADO.NET komponentus. Taigi, eksperimento rezultatai, kai iš dviejų duomenų lentelių išrenkama keletas įrašų, pagal nurodytą identifikatorių, padės įvertinti šią situaciją:

8 lentelė. Eksperimento rezultatai, kai iš dviejų duomenų lentelių išrenkami keli įrašai

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Vidurkis**
DataSet	190	170	170	160	190	160	160	160	170	180	160	170	**170**
SqlDataReader	140	150	150	130	150	140	140	130	140	150	140	140	**142**

DataSet ir SqlDataReader komponentų naudojimo galimybių palyginimo diagrama:



34 pav. Keleto įrašų išrinkimas iš keleto duomenų bazės lentelių

Vėl gi diagramoje galime matyti, jog DataSet komponentas veikimo charakteristikomis nusileidžia SqlDataReader komponentui maždaug 1,2 karto. Beje, reikėtų

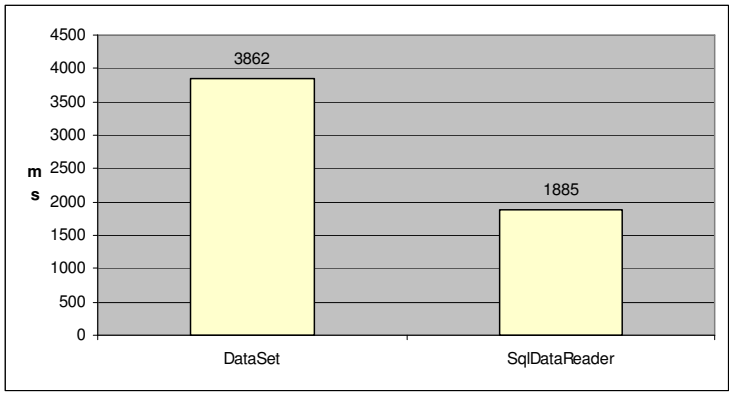
pastebėti, jog vidutinis užklausos įvykdymo laikas yra praktiškai identiškas laikui, kuris buvo gautas, vykdant analogišką eksperimentą su viena duomenų lentele.

Trečioje šio eksperimento dalyje pateikiami rezultatai, kai iš dviejų duomenų lentelių išrenkami visi įrašai:

9 lentelė. Eksperimento rezultatai, kai iš dviejų duomenų lentelių išrenkami visi įrašai

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Vidurkis**
DataSet	3775	3845	3745	3755	4075	3765	3895	3745	3915	3935	3875	4015	**3862**
SqlDataReader	1872	1922	1792	1982	1882	1872	1872	1842	1862	1922	1922	1872	**1885**

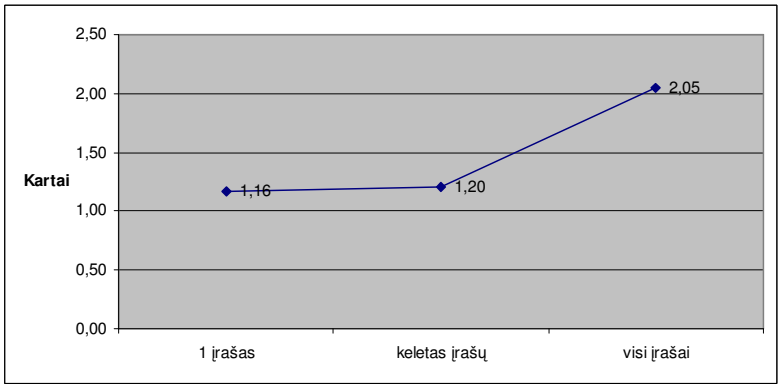
DataSet ir SqlDataReader komponentų naudojimo galimybių palyginimo diagrama:



35 pav. Visų įrašų išrinkimas iš keleto duomenų bazės lentelių

Susiklosčius tokiai situacijai, kai išrenkami visi dviejose lentelėse esantys duomenys, matyti akivaizdus SqlDataReader komponento pranašumas prieš DataSet. Tokio tipo užklausa, naudojant SqlDataReader komponentą, įvykdoma daugiau nei 2 kartus greičiau.

DataSet ir SqlDataReader komponentų santykinių greičių palyginimo diagrama:



36 pav. DataSet ir SqlDataReader santykinio greičio palyginimo diagrama

5.4. Manipuliavimo duomenimis įvertinimas naudojant duomenis iš trijų DB lentelių

Eksperimentas buvo atliekamas su trimis duomenų lentelėmis. Pirmojoje buvo 100 įrašų, antrojoje – 1 000, o trečiojoje – 10 000 įrašų. Tam, kad eksperimento rezultatai vizualiai labiau skirtųsi, buvo atliekama 20 iteracijų, t.y. ta pati užklausa vykdoma 20 kartų ir

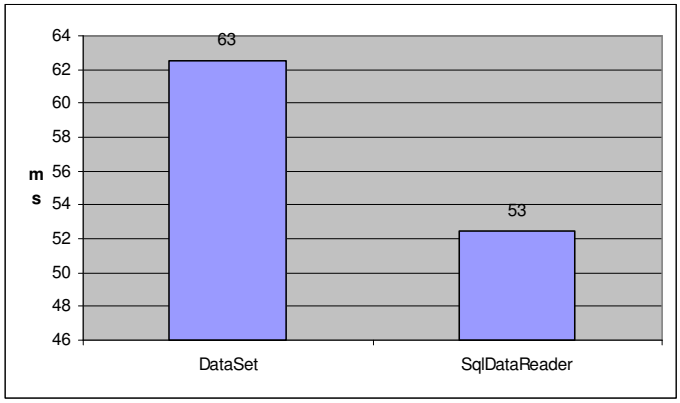
tik po to grąžinamas užklausos įvykdymo laikas milisekundėmis. Eksperimentas buvo atliekamas, vykdant užklausas, kurios grąžina vieną, keletą (mūsų atveju konkrečiai 109 įrašus) ir visus įrašus iš duomenų lentelių. Tokiu principu kiekvieno eksperimento metu buvo atlikta 12 bandymų, kurių rezultatai užfiksuoti lentelėse ir diagramose.

Eksperimento rezultatai, kai iš trijų duomenų lentelių išrenkamas vienas įrašas:

10 lentelė. Eksperimento rezultatai, kai iš trijų duomenų lentelių išrenkamas vienas įrašas

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Vidurkis**
DataSet	60	60	60	70	60	70	70	60	60	60	60	60	**62,5**
SqlDataReader	50	60	50	60	50	50	50	50	50	60	50	50	**52,5**

DataSet ir SqlDataReader komponentų naudojimo galimybių palyginimo diagrama:



37 pav. Vieno įrašo išrinkimas iš keleto duomenų bazės lentelių

Eksperimento rezultatai, kai iš trijų duomenų lentelių išrenkami keli įrašai:

11 lentelė. Eksperimento rezultatai, kai iš trijų duomenų lentelių išrenkami keli įrašai

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Vidurkis**
DataSet	180	170	170	160	170	180	180	180	170	170	180	180	**174,2**
SqlDataReader	140	150	140	140	140	150	140	150	150	150	140	140	**144,2**

DataSet ir SqlDataReader komponentų naudojimo galimybių palyginimo diagrama:

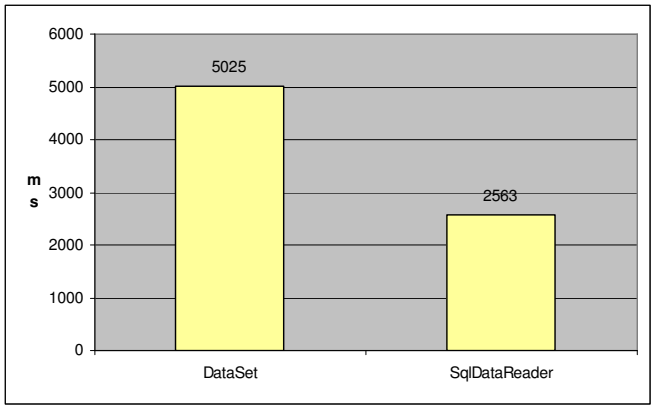
38 pav. Keleto įrašų išrinkimas iš keleto duomenų bazės lentelių

Eksperimento rezultatai, kai iš dviejų duomenų lentelių išrenkami visi įrašai:

12 lentelė. Eksperimento rezultatai, kai iš trijų duomenų lentelių išrenkami visi įrašai

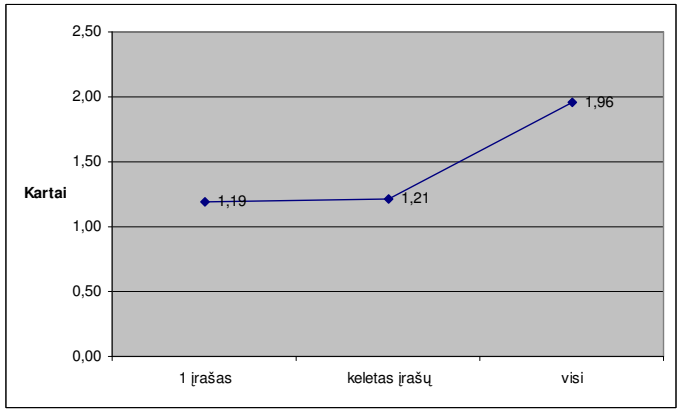
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Vidurkis**
DataSet	5017	4977	5107	5017	5067	4997	4997	5017	5097	5004	4967	5037	**5025,1**
SqlDataReader	2473	2593	2613	2613	2513	2543	2613	2603	2573	2613	2513	2493	**2563,0**

DataSet ir SqlDataReader komponentų naudojimo galimybių palyginimo diagrama:



39 pav. Visų įrašų išrinkimas iš trijų duomenų bazės lentelių

DataSet ir SqlDataReader komponentų santykinių greičių palyginimo diagrama:



40 pav. DataSet ir SqlDataReader santykinio greičio palyginimo diagrama

Apibendrinant šio eksperimento rezultatus, reikėtų išskirti tokius esminius aspektus:

- Visais atvejais užklausos įvykdymo laikas buvo geresnis naudojant SqlDataReader komponentą.
- SqlDataReader efektyvumo koeficientai, išrenkant vieną arba keletą įrašų iš trijų duomenų lentelių, buvo praktiškai identiški koeficientams, gautiems atliekant analogiškus eksperimentus su viena ir dviem lentelėm.
- Išrenkant vieną įrašą iš trijų lentelių, užklausos įvykdymo laikas išaugo 1,17-1,19 karto.
- Išrenkant visus įrašus iš trijų lentelių, užklausos įvykdymo laikas išaugo 1,67, naudojant DataSet komponentą, ir net 2,13 karto, naudojant SqlDataReader komponentą.

5.5. Spartinančiosios atmintinės naudojimo tyrimas

Eksperimento tikslas – išsiaiškinti spartinančiosios atmintinės panaudojimo galimybes, siekiant optimizuoti duomenų prieigą.

Eksperimentas buvo atliekamas su dviem lentelėm. Pirmojoje buvo 100 įrašų, antrojoje - 10 000 įrašų. Duomenų išgavimui buvo panaudoti ADO.NET duomenų prieigos komponentai: DataSet ir SqlDataReader. Tam kad eksperimento rezultatai vizualiai labiau

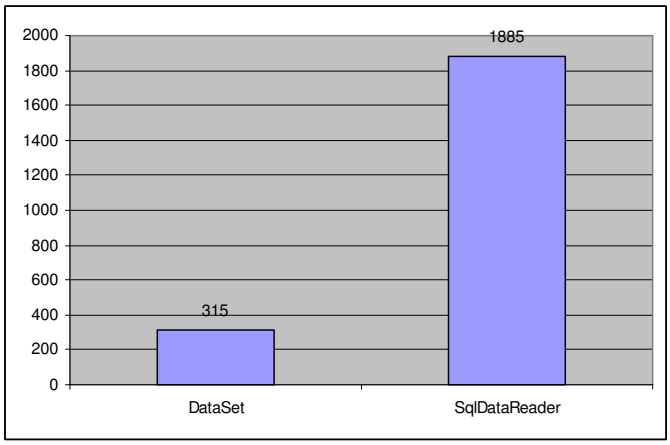
skirtųsi, buvo atliekama 20 iteracijų, t.y. ta pati užklausa vykdoma 20 kartų ir tik po to grąžinamas užklausos įvykdymo laikas milisekundėmis. Tokiu principu kiekvieno eksperimento metu buvo atlikta 12 bandymų, kurių rezultatai užfiksuoti lentelėse. Beje, pirmosios bandymo metu DataSet komponentas buvo patalpintas į spartinančiąją atmintinę, ir vėlesniuose bandymuose duomenys buvo užkraunami į komponentą iš spartinančiosios atmintinės, taip išvengiant pakartotinių kreipinių į duomenų bazę.

13 lentelė. Spartinančiosios atmintinės naudojimo tyrimo rezultatai

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Vidurkis**
Spart. atmintis	3775	0	0	0	0	0	0	0	0	0	0	0	**315**
SqlDataReader	1872	1922	1792	1982	1882	1872	1872	1842	1862	1922	1922	1872	**1885**

Naudojant spartinančiąją atmintinę, tik pirmo bandymo metu buvo kreipiamasi į duomenų bazę. Kitų bandymų metu buvo naudojami duomenys iš spartinančiosios atmintinės, todėl tokios užklausos įvykdymas užtrukdavo labai trumpą laiką, kurio nepavyko užfiksuoti.

Eksperimento rezultatai, kai iš dviejų duomenų lentelių išrenkami visi įrašai naudojant spartinančiąją atmintinę ir SqlDataReader komponentą:



41 pav. Spartinančiosios atminties ir SqlDataReader vykdymo laikų palyginimo diagrama

Diagramoje matyti, jog naudojant DataSet komponento patalpinimo į spartinančiąją atmintinę technologiją, akivaizdžiai pagerėja veikimo charakteristikos. Iki tol visuose eksperimentuose DataSet veikimo charakteristikos buvo prastesnės. Atlikus šį eksperimentą, matome, jog dabar DataSet lenkia SqlDataReader maždaug 5 kartus. Tiesa, reikėtų pabrėžti, jog efektyvumo koeficientas yra tiesiogiai priklausomas nuo kreipinių į duomenų bazę skaičiaus, t. y. kuo daugiau kartų tenka kreiptis į duomenų bazę, tuo efektyvumas yra didesnis.

6. Išvados

1. Siekiant greičiau pacientams suteikti kokybiškas sveikatos priežiūros paslaugas, pagerinti medicinos personalo kompetenciją bei greitą prieigą prie patikimos ir visapusiškos informacijos apie pacientą, pagerinti vadovų ir administratorių galimybes gauti patikimą informaciją valdymui ir planavimui bei išvystyti keitimosi informacija su kitomis įstaigomis sistemą, būtina kompiuterizuoti medicinos veiklos procesus.
2. Kuriant medicinai pritaikytas informacijos sistemas, rekomenduojama naudoti trijų lygių kliento-serverio architektūrą su atskirtais atvaizdavimo, veiklos bei duomenų lygiais. Mažesnės išlaidos techninei įrangai, mažiau sudėtingas tokių sistemų kūrimo ir priežiūros procesas – pagrindiniai trijų lygių architektūros privalumai.
3. Širdies centro kardiologinių tyrimų surinkimo ir analizės programų sistemai realizuoti buvo pasirinkta MS SQL Server duomenų bazių valdymo sistema. Palyginti su Oracle ji yra pigesnė, lengviau įdiegiama bei valdoma, patikimumo ir saugumo lygis pakankamas. Lyginant su MySQL - nemokama duomenų bazių valdymo sistema, veikiančia įvairiose platformose, nereikalaujančia didelių techninės įrangos resursų, MS SQL Server yra aukštesnis patikimumo lygis, lengvesnis diegimas bei valdymas, didesnis papildomų funkcijų kiekis (serverio procedūros, trigeriai, kursoriai), galingesnė užklausų kalba.
4. Programų sistemai realizuoti su prieiga per internetą buvo pasirinkta .NET technologija. Toks sprendimas priimtas todėl, kad .NET naudoja objektinę programavimo kalbą, pasižymi objektams būdingomis savybėmis, yra labai lanksti, nes čia yra galimybė programuoti su C++, C#, VB.NET, J# įrankiais, be to, .NET technologija buvo papildyta keletu objektų, kurie turi teigiamos įtakos programų sistemos nefunkciniams reikalavimams: sistemos greičiui, duomenų pasiekiamumui.
5. Sukurtas produktas – kardiologinių tyrimų surinkimo bei analizės programų sistema. Tai produktas, skirtas organizacijai, kurios pagrindinė veikla – kardiologinių bei kardiochirurginių duomenų kaupimas bei pakartotinis jų panaudojimas. Šio produkto pagalba bus registruojami KMU Širdies centre aptarnaujamų pacientų būklės, jiems atliekamų tyrimų, operacijų, statistinei analizei naudojami CARDS duomenys, iš užregistruotų duomenų formuojamos diagnozės, ataskaitos, vykdomi pacientų perkėlimai iš vienos skyriaus į kitą, išrašymai namo ar į kitą gydymo įstaigą.
6. Informacijos sistemoje registruojama informacija yra labai įvairiapusė, t. y. registruojami dideli informacijos kiekiai. Kadangi sąveika su sistema vyksta

interaktyviai, atsakymai į visas užklausas turi būti gaunami kuo greičiau. Taip iškyla efektyvaus manipuliavimo duomenimis poreikis.

7. Galima skirti tris pagrindinius efektyvaus manipuliavimo duomenimis užtikrinimo principus: serverio procedūrų, ADO.NET duomenų prieigos komponentų ir spartinančiosios atminties panaudojimas.
8. Lyginant serverio procedūras su tiesioginiais SQL kreipiniais, gauti rezultatai rodo, kad užklausos įvykdymo laikas nedaug geresnis naudojant serverio procedūras. Įvertinus rezultatus, gali susidaryti įspūdis, jog nėra jokio skirtumo, kurį iš metodų naudoti, tačiau taip nėra, kadangi naudojant serverio procedūras, duomenų bazė gali optimizuoti duomenų prieigos planą ir naudoti spartinančiąją atmintį pakartotiniam panaudojimui. Be to, serverio procedūrų naudojimas sumažina tinklo užimtumą. Tai turi papildomos teigiamos įtakos sistemos veikimo charakteristikoms.
9. SqlDataReader duomenų prieigos komponento panaudojimas duomenų manipuliacijos procese visais atvejais yra efektyvesnis už DataSet komponento panaudojimą. Ypatingai SqlDataReader komponento pranašumas matomas tada, kai reikia išgauti visus įrašus iš duomenų lentelių. Tuomet SqlDataReader panaudojimas yra maždaug 100-150% efektyvesnis. Visais kitais atvejais SqlDataReader komponento panaudojimas yra maždaug 20% efektyvesnis už DataSet.
10. DataSet komponento talpinimas į spartinančiąją atmintinę akivaizdžiai pagerina sistemos veikimo charakteristikas. Efektyvumo prieaugio koeficientas yra tiesiogiai priklausomas nuo kreipinių į duomenų bazę skaičiaus, t.y. kuo daugiau kartų tenka kreiptis į duomenų bazę, tuo spartinančiosios atmintinės naudojimo efektyvumas yra didesnis. Tiesa, reikėtų pabrėžti tai, jog spartinančiosios atmintinės naudojimas yra efektyvus tik tada, kai operuojama globaliais, retai kintančiais duomenimis. Priešingu atveju sistemos veikimo charakteristikos netgi gali pablogėti.
11. Didėjantis lentelių ir ryšių tarp jų skaičius daro neigiamą poveikį užklausos vykdymo charakteristikoms. Neigiamas poveikis veikimo charakteristikoms auga tiesiškai ir yra labiau pastebimas, naudojant SqlDataReader komponentą.
12. Parengtas ir 11-oje tarpuniversitetinėje doktorantų ir magistrantų konferencijoje “Informacinės technologijos 2006” perskaitytas mokslinis straipsnis tema “Efektyvus manipuliavimas duomenimis medicininėse informacijos sistemose”. Straipsnis taip pat išspausdintas konferencijos pranešimų medžiagoje.

7. Literatūra

1. *.NET Data Access Architecture Guide*. [interaktyvus]. [žiūrėta 2006 03 06]. Prieiga per internetą: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/daag.asp
2. *ADO.NET architecture*. [interaktyvus]. [žiūrėta 2006 03 06]. Prieiga per internetą: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconadonetarchitecture.asp
3. Crocker A., Olsen A., Jezierski E., *Designing Data Tier Components and Passing Data Through Tiers*. [interaktyvus]. [žiūrėta 2006 03 07]. Prieiga per internetą: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/BOAGag.asp
4. *Elektroninės sveikatos strategija 2005 – 2010 m.*. Sveikatos apsaugos ministerija [interaktyvus]. 2004, [žiūrėta 2006 02 24]. Prieiga per internetą: http://sam.lt/images/Dokumentai/eSveikata/esveikata_strategija_web020.doc
5. EMRUpdate. [interaktyvus]. [žiūrėta 2006 03 08]. Prieiga per internetą: http://www.elmr-electronic-medical-records-emr.com/resources/keyrequirements.aspx
6. Giulietti R., Pedrazzini S., *Thin client for web using swing*. TINET SA, 6928 Manno, Switzerland.
7. Howard R.., *10 Tips for Writing High-Performance Web Applications* http://msdn.microsoft.com/msdnmag/issues/05/01/ASPNETPerformance/default.aspx
8. Kahn J., *HIPPA : the critical role of strong authentication*. [interaktyvus]. 2002, balandis [žiūrėta 2006 04 01]. Prieiga per internetą: http://amsys.net/healthcare/pdf/Critical%20Role%20Of%20Strong%20Authentication.pdf
9. Lohman P., *EMR – making the buisness decision for your practise*. Orlando, Florida, U.S., Unified Osteophatic Convention, 2005.
10. Lowber P., *This-client vs Fat-client TCO*. [interaktyvus]. 2001, rugsėjis [žiūrėta 2006 02 23]. Prieiga per internetą: http://h20202.www2.hp.com/Hpsub/downloads/WhitePaper_Gartner_ThinClient_TCO.pdf
11. *Microsoft Data Access Development Overview*. [interaktyvus]. [žiūrėta 2006 03 06]. Prieiga per internetą: http://www.microsoft.com/downloads/details.aspx?familyid=B57B5EC9-9485-4EB6-86EE-A5AC713D3DA9&displaylang=en

12. *N-tier Architecture*. [interaktyvus]. [žiūrėta 2006 03 06]. Prieiga per internetą: http://www.infusionsoft.com/Technology/n-tier.jsp
13. *OLAP duomenų bazės*. [interaktyvus]. [žiūrėta 2006 03 18.]. Prieiga per internetą: http://itekspertas.projektas.lt/index.php?option=com_content&task=view&id=94&Itemid=51
14. Practice Partner. [interaktyvus]. [žiūrėta 2006 03 01]. Prieiga per internetą: http://www.pmsi.com/userprofiles/userprofilecardiology.htm
15. Priya D., *Performance Comparison: Data Access Techniques*. [interaktyvus]. [žiūrėta 2006 03 06]. Prieiga per internetą: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/bdadotnetarch031.asp
16. Wircz N. *Overview of IT-Standards in Healthcare*. Electromedica Journal, 2000 no. 1

8. Santrumpų ir terminų žodynas

Santrumpos

Aspx - Active Server Page Framework (Microsoft)

BLOB – binary large object

CORBA – Common Object Request Broker Architecture

DB - duomenų bazė

DBVS – duomenų bazių valdymo sistema

DICOM - Digital Imaging and Communications in Medicine

EMK – elektroninė medicinos kortelė

ESĮ - elektroninis sveikatos įrašas

IE – Internet Explorer

IKT – informacija, komunikacijos, technologijos

IIS - Internet Information Service

IS - informacijos sistema

HIPPA - Health Insurance Portability and Accountability Act

HL7 – Health Level 7

HTTP - Hyper Text Transfer Protocol

HTTPS - Hyper Text Transfer Protocol Secure sockets

KMU - Kauno medicinos universitetas

KMUK - Kauno medicinos universiteto klinikos

.NET - Microsoft XML Web Services platforma

MS - „Microsoft“

MSDE - Microsoft SQL Desktop Edition

ODBC – Open Database Connectivity

OLE – Object Linking & Embedding

SP – Service Pack

SSL – Secure Sockets Level

SQL - struktūrizuota užklausų kalba

TCP – Transmission Control Protocol

TĮ – techninė įranga

UPS - Uninterruptible Power Supply

WWW - world wide web

XML - eXtensible Markup Language

Terminai

Administratorius - visateisis kompiuterio, kompiuterių tinklo arba kurios nors kitos kompiuterių sistemos tvarkytojas. Rūpinasi naujų programų diegimu ir atnaujinimu, sistemos saugumu, duomenų išsaugojimu, kasdienine sistemos priežiūra.

Antivirusinė programa - programa, kurios paskirtis yra aptikti bei neutralizuoti kompiuterio virusus. Ji peržiūri ir tikrina kompiuterio atmintinėje esančius ir (arba) į kompiuterį patenkančius duomenis.

Internetas - gali būti apibrėžtas kaip plačios aprėpties kompiuterių tinklas, aprėpiantis visą pasaulį.

Intranetas - įmonės vidinė sistema, leidžianti kompanijai elektroninėmis priemonėmis efektyviai valdyti vidinius įmonės procesus.

Meniu - operacijų (funkcijų) pavadinimų sąrašas, iš kurio galima pasirinkti norimą atlikti operaciją ar išskleisti operacijų grupės sąrašą. Yra įvairūs pasirinkimo būdai: 1) atvesti pelės žymeklį ant sąrašo elemento ir spragtelėti pelės klavišu sąrašo elementą; 2) krypčių klavišais nuvesti žymeklį ant meniu elemento pavadinimo ir paspausti įvedimo klavišą; 3) paspausti pabrauktos ar kitaip paryškintos meniu elemento pavadinimo raidės klavišą kartu su klavišu Alt; 4) paspausti kitokią, tam elementui skirtą klavišų kombinaciją;

Serveris - tarnybinės stoties kompiuteris arba kompiuterio programa, aptarnaujanti kitus kompiuterius arba jų programas – klientus pagal jų paraiškas (užklausas);

„Windows“ - „Microsoft” operacinė sistema;