KAUNO TECHNOLOGIJOS UNIVERSITETAS
Informatikos fakultetas
Informacijos sistemų katedra

Saulius Putinas

Duomenų bazių reinžinerijos procesų analizė ir plėtra

Magistro darbas

Darbo vadovas:
doc.dr. **R. Butleris**

KAUNAS
2004

KAUNO TECHNOLOGIJOS UNIVERSITETAS
Informatikos fakultetas
Informacijos sistemų katedra

TVIRTINU
Katedros vedėjas
doc. dr. R.Butleris
2004 05

Duomenų bazių reinžinerijos procesų analizė ir plėtra

Magistro darbas

Kalbos konsultantė
Lietuvių kalbos katedros lektorė
dr. Jurgita Mikelionienė
2004 05

Vadovas
doc.dr. R. Butleris
2004 05

Recenzentas
doc.dr. E.Karčiauskas
2004 05

Atliko
IFM-8/1 gr. stud.
S.Putinas
2004 05

KAUNAS
2004

Turinys

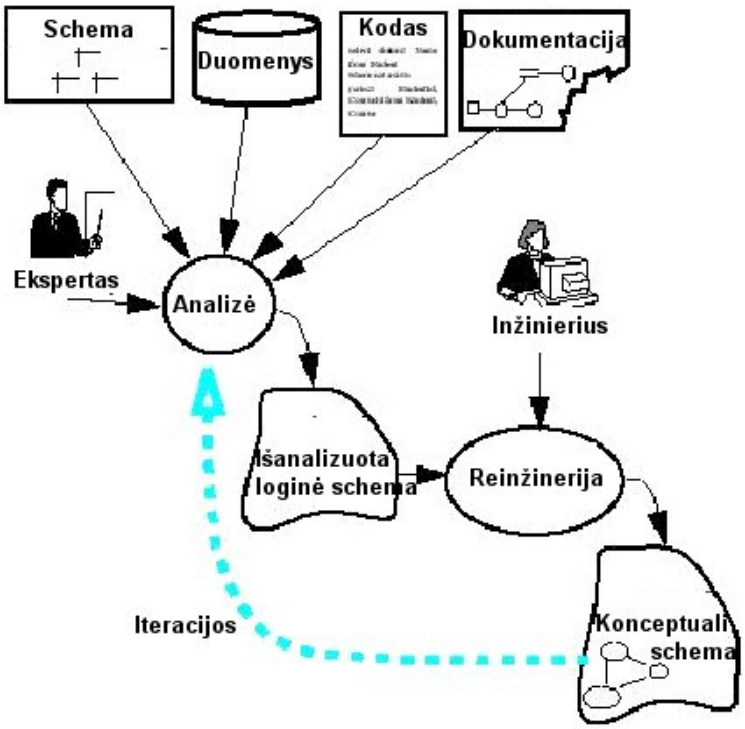
1 ĮVADAS

Naujai atsiradusios technologijos, tokios kaip internetas, objektinis programavimas (OO), kliento/serverio aplikacijos ir atvirų sistemų standartai (pvz., Corba) daro didelę įtaką modernaus verslo plėtrai. Be naujų aplikacijų, pasirodančių elektroninėje rinkoje, taip pat labai išaugo senų, įmonėje naudojamų, informacinių sistemų (IS) panaudojimo ir palaikymo poreikis. Naujos įmonės, žinoma, gali nusipirkti naujas informacines sistemas, kurios naudoja naujausias informacines technologijas. Tačiau pamažu įmonėms plečiantis, duomenų srautai jose neišvengiamai didėja, todėl jos priverstos pildyti ir plėsti senas informacines sistemas. Tai labai nelengvas uždavinys, nes dažniausiai tokios sistemos būna modifikuotos daugelio programuotojų, kurie galbūt jau nebedirba įmonėje, bei dažniausiai tokioms sistemoms dokumentacija būna labai menka, arba jos išvis nebūna. Didelėms organizacijoms būdinga funkcinių reikalavimų ir informacijos poreikių kaita. Taigi šiame darbe bus aptariami metodai, kaip integruoti senų duomenų bazių schemas ir naujų duomenų struktūras, kurios reikalingos pakitusiems informacijos poreikiams tenkinti.

Kompiuterizuotoms informacijos sistemoms tobulinti, arba pakartotinėje inžinerijoje (reinžinerijoje), plačiai naudojami informacijos sistemų programinės įrangos kūrimo automatizavimo įrankiai. Tai spartina projekto įgyvendinimą, mažina pasikartojančių darbų kiekį. CASE priemonėmis aprašomos ir deklaruojamos duomenų bazių (DB) schemos. CASE įrankių kūrimas pagrindžiamas tam tikra metodologija. Plačiausiai taikomos tradicinė struktūrinės analizės metodologija ir objektiškai orientuota metodologija. Taikant struktūrinę analizę pagrindinis dėmesys skiriamas dalykinėje srityje vykstantiems procesams ir jų informacinei sąveikai. Šiai metodologijai būdingas reliacinių duomenų bazių naudojimas. Metodologijos pagrindą sudaro duomenų srautų modelis. Kadangi duomenys migruoja įvairiose aplinkose (žmonių kolektyvuose, skirtingų modelių duomenų bazėse, skirtingų koncepcijų programavimo kalbų aplinkose), duomenų struktūros transformuojamos. Čia duomenų struktūrų transformavimo metodai ir specifikacijos yra ne mažiau svarbūs kaip procesų analizės ir specifikavimo metodai.

2 DUOMENŲ BAZĖS REINŽINERIJOS SAMPRATA

Tradiciškai duomenų bazės loginio lygio (programuotojo arba DBVS lygio) schema išvedama arba generuojama automatiškai iš koncepcinio lygio (vartotojo arba verslo procesų lygio) globalinio duomenų modelio − EER diagramų arba klasių diagramos. Šis globalinis modelis suprojektuojamas iš naujo, pasinaudojant palikuoninės KIS projektu, jeigu jis toks buvo ar išliko, ir įvertinant organizacijos funkcinės veiklos pokyčius. KIS pakartotinę inžineriją galima suprasti kaip nuoseklų perėjimą (2.1 pav.) nuo palikuoninės KIS prie naujos, o nuoseklų perėjimą − kaip nenutrūkstamą perėjimą prie naujos KIS: čia vartotojų seni informaciniai poreikiai ir atitinkami aktualūs uždaviniai paliekami spręsti, tačiau prie jų pridedami ir nauji uždaviniai, kurie reikalingi naujiems poreikiams tenkinti.



2.1 pav. IS pakartotinės inžinerijos procesas

Vienas iš svarbiausių daugumos šiuolaikinių duomenų bazių reinžinerijos paketų trūkumų yra tai, kad jie neįvertina reinžinerijos proceso vystymosi ir pakartotinio tyrinėjimo bei vertinimo ypatumų. Jie naudoja griežtai nuoseklų metodų vykdymą, šis būdas dar kitaip vadinamas krioklio reinžinerijos metodu, kuris neįvertina reinžinerijos iteracijų (2.1 pav.) Tai labai svarbus trūkumas, taikant tuos paketus praktiškai, nes labai dažnai reinžinerijos proceso metu atsiranda iteracijos tarp schemos analizės ir projektavimo

žingsnių. Pavyzdžiui, kai inžinierius-programuotojas projektuodamas naują duomenų bazę netikėtai sužino naujų detalių apie seną DB ir kai tenka pakeisti pradines sąlygas ar prielaidas bei tęsti darbą toliau. Taip pat perėjimas iš senos IS į naują gali užtrukti kelis mėnesius ar net metus. Taigi gali būti, kad staiga atsiras vis naujų ir naujų reikalavimų senajai IS ir duomenų bazei, kol dar jos naudojamos, ir visi tie reikalavimai turi būti įvykdyti naujojoje IS ir duomenų bazėje. Visa tai kaip tik ir reikalauja iteracijų reinžinerijos procese. Šiuo atveju yra prarandama labai daug laiko ir darbo sąnaudų.

Be šios problemos dar yra ir kitos problemos labai susiaurinančios reinžinerijos paketų panaudojimą [2]. Tai programinės įrangos, skirtos DB reinžinerijai, lankstumas ir pritaikymas darbui su iš paprastu žmonių gauta informacija. Lankstumas reikalingas tam, kad skirtingos informacinės sistemos ir BD būna suprojektuotos skirtingomis DBVS bei programavimo kalbomis, taip pat turi skirtingas struktūras, kodavimo principus ir vardų standartus. Taigi reinžinerijos tikslų ratas būna labai platus (nuo 2000-ųjų metų problemos išsprendimo, duomenų papildymo, naujų procedūrų pridėjimo iki visiško architektūros pakeitimo), kurios kaip tik ir turėtų padėti pasiekti šie paketai. Taip pat dauguma informacijos apie egzistuojančią DBVS yra gaunama iš žmonių: ją suprojektavusių projektuotojų, dirbančių su ja operatorių bei administratorių prižiūrinčių pasenusią DBVS. Todėl, kaip jau ir buvo minėta, svarbu, kad vykdant reinžineriją būtų atsižvelgiama ir į jų pateiktas žinias.

Tačiau bet kuriame, reinžinerijai skirtame pakete, yra naudojami panašūs principai. Kai kurioje literatūroje yra išskiriami ir nagrinėjami du veiklos procesų modeliavimo metodų tipai, skirti reinžinerijai :

• veiklos procesų modeliavimas, siekiant suprasti egzistuojančias valdymo funkcijas ir paruošti geresnį KIS projektą; čia nurodomi ir charakterizuojami *UML* pagrįsti metodai, *Provision* ir *Popkin metodikos*; KIS projekto rezultatas yra veikiantys programiniai komponentai, kuriuos vartotojai susieja į duomenų apdorojimo procesą realizavimo stadijoje;

• veiklos procesų modeliavimas, siekiant automatizuoti veiklos proceso valdymą; čia išskiriamas darbų sekų procesų modeliavimas, orientuotas į organizacijos vidinių procesų valdymą, ir išorinių procesų (*B2B - Business to Business* ar *B2C - Business to Customer*) modeliavimas; šiai modeliavimo rūšiai priskiriami *Workflow* ir *DEMO* metodai.

2.2 paveiksle pateiktoje schemoje [2] akcentuojamas esminis požiūrio skirtumas į KIS pakartotinę inžineriją: čia nauja KIS projektuojama iš organizacijos procesų modelio, kuris

orientuotas į organizacijos darbų sekų vidinių ir išorinių procesų modeliavimą. Funkciniai reikalavimai pateikiami darbų sekomis, o funkcinių reikalavimų pokyčiai fiksuojami vartotojų koncepciniame lygyje, koreguojant modelio elementus − nereikalingus pašalinant, įvedant naujus, paliktus ir naujus susiejant tarpusavyje. Čia nuosekliai perduodama informacija apie veiklos funkcijų pokyčius ir naujus funkcinius reikalavimus KIS automatizuoto projektavimo aplinkai − projektavimo žinių bazei.

2.2 pav. IS reinžinerijos procesas, pagrįstas veiklos procesų reinžinerija

Įvedus naujus funkcinius reikalavimus, gali prireikti ir papildomų duomenų struktūrų. Po kiekvieno naujo kompiuterizuoto proceso įvedimo reikia spręsti klausimą, ar pakankami yra eksploatuojami KIS DB informaciniai ištekliai ar nepakankami. Dėl informacinio nepakankamumo tenka projektuoti papildomas duomenų struktūras. Naujų informacinių technologijų ir naujų DBVS naudojimas, viena vertus, priverčia numatyti naujas DB, o palikuoninių duomenų bazių apimtys ir išliekamoji jų vertė verčia naujoje KIS naujas DB eksploatuoti kartu su palikuoninėmis. Palikuoninių duomenų bazių išsaugojimas vertingas ir ta prasme, kad galima palikti eksploatuoti nepakitusius uždavinius. Kelių DBVS (arba tos pačios, bet skirtingų versijų), kurios palaiko tokius pat ar skirtingus duomenų modelius, naudojimas lemia nehomogeninių DB projektavimo metodų atsiradimą. Tiesioginėje KIS inžinerijoje DB nehomogeninių komponentų integravimas įgauna centrinį vaidmenį. Nuolatinės funkcinių reikalavimų kaitos sąlygomis ypač aktualus tampa nehomogeninių DB komponentų projektavimo ir integravimo CASE priemonių sudarymas. Šios klasės CASE priemonės gali būti priskirtos technologinėms priemonėms, skirtoms užtikrinti greitą taikomųjų programų išplėtojimo procesą. Jos naudojamos KIS reinžinerijoje, kai nauji trūkstami ar replikuojami DB komponentai projektuojami ar identifikuojami tiesiogiai panaudojant informaciją apie funkcinių reikalavimų papildymus.

3 DB SCHEMŲ DERINIMAS

Tiesioginės IS reinžinerijos procesus sudaro: kompiuterizuojamų funkcijų išskyrimas; DB schemos papildymas naujais fragmentais; fragmentų sujungimas (schemų integravimas); vartotojų uždavinių papildymas programų komponentais. KIS atvirkštinės inžinerijos esmę sudaro palikuoninės KIS duomenų apdorojimo procesų (palikuoninių DB) transakcijų analizė ir atvaizdavimas globaliu procesų koncepciniu modeliu, kuriame fiksuojami funkcinių reikalavimų pokyčiai.

Informacinės sistemos DB schemos papildymas ir priderinimas yra viena iš pagrindinių daugumos duomenų bazių reinžinerijos ir pritaikymo sričių, tokių kaip duomenų integracija, el. verslo struktūros, duomenų saugojimas, problema. Dabartiniams taikymams DB schemos priderinimas paprastai atliekamas rankiniu būdu, kas smarkiai apriboja panaudojimą. Iš kitos pusės, ankstesni tyrimai siūlo daug metodų, padedančių pasiekti dalinį derinimo operacijų automatizavimą specifinėms pritaikymo sritims. Toliau bus parodyti skirtumai tarp scheminių ir faktinių lygių, elementų ir struktūrinių lygių bei skirtumai tarp derinimo programų, paremtų kalba ar apribojimais. Apžvelgsime ankstesnius derinimo taikymus, tuo pačiu nurodydami, kurią sprendimo dalį jie paaiškina. Bus siekiama apibendrinti ir apžvelgti skirtingus požiūrius į DB schemos reinžineriją bei priderinimą, vystant naujus derinimo algoritmus ir taikant naujus schemos derinimo komponentus.

Viena iš atliekamų operacijų DB schemos reinžinerijoje yra derinimas [1]. Jis apima dvi įvedamas DB schemas ir sutapatina dviejų schemų, kurios semantiškai atitinka viena kitą, elementus. Šiuo metu schemos priderinimas dažniausiai atliekamas rankiniu būdu, greičiausiai remiantis grafine vartotojo aplinka. Akivaizdu, kad rankinis schemos derinimas yra nuobodus, užimantis daug laiko, suteikiantis galimybių klaidoms atsirasti ir, tokiu būdu, brangus procesas. Atsižvelgiant į staigiai didėjantį integruojamų tinklo duomenų šaltinių bei el. verslų skaičių, tai – auganti problema. Taip pat kadangi sistemos gali palaikyti sudėtingesnes duomenų bazes ir aplikacijas, jų schemos tampa didesnės, taigi didėja ir galimų atlikti derinimų skaičius. Pastangų lygis, jeigu reikia įvertinti kiekvieną derinį tarp galimų visų elementų derinių, yra tiesiškai proporcingas ar net didesnis. Reikia greitesnio ir mažiau darbo jėgos reikalaujančio integravimo metodo. Tam reikia automatinės DB schemos derinimo, palaikymo.

3.1 Schemos integravimas

Dauguma darbų apie schemos priderinimą buvo motyvuojami schemos integravimu – problema, kuri tiriama nuo devintojo dešimtmečio pradžios: atsižvelgiant į nepriklausomai viena nuo kitos išvystytas schemas, kuriamas globalinis požiūris.

Kadangi schemos yra nepriklausomai išvystytos, jos dažnai turi skirtingą sandarą ir terminologiją. Tai akivaizdžiai pastebima tada, kai schemos yra iš skirtingų sričių, tokių kaip nekilnojamojo turto schema ir turto mokesčių schema. Tačiau taip pat matyti, kad net ir tos pačios realios pasaulinės srities modeliai skiriasi tik todėl, kad sukurti skirtingų žmonių skirtingame realaus pasaulio kontekste. Todėl pirmas žingsnis integruojant schemas yra šių tarpscheminių ryšių identifikavimas ir apibūdinimas. Tai yra schemos derinimo procesas. Kai tik jie nustatomi, atitinkami elementai gali būti suvienyti į koherentinę, integruotą schemą. Šio integravimo eigoje, o kartais ir atskirai, programos ar užklausos kuriamos taip, kad leistų perduoti duomenis iš originalių schemų į integruotąją.

Schemos integravimo problemos pobūdis kyla dėl atskirai išvystytų schemų integravimo į duotąją konceptualią schemą. Tačiau ir vėl tam reikia suderinti dviejų schemų sandarą bei terminologiją, ką ir apima schemos derinimas.

3.2 Duomenų saugyklos

Duomenų saugykla yra sprendimais paremta duomenų bazė, kuri gaunama iš daugelio duomenų šaltinių. Jos sudarymo procesui reikia duomenų perdavimo iš šaltinio formato į saugyklos formą. Derinimo operacija yra naudinga perdavimo sandaros sudarymui. Atsižvelgiant į duomenų šaltinį, vienas tinkamo perdavimo sukūrimo metodas yra surasti tuos šaltinio elementus, kurie yra ir saugykloje. Tai yra derinimo operacija. Kai tik sukuriama pradinė sandara, saugyklos kūrėjas turi išanalizuoti detalizuotą kiekvieno šaltinio elemento semantiką ir sukurti perdavimą, kuris suderintų jų semantiką su norima išgauti.

Kitas naujo duomenų šaltinio S‘ integravimo metodas yra panaudoti egzistuojančius perdavimus iš šaltinio į saugyklą (S=>W). Pirmiausia randami bendri S‘ ir S elementai (derinimo operacija) ir tuomet šiems bendriems elementams panaudojami S=>W.

3.3 Semantinės užklausos apdorojimas

Schemos integravimas, duomenų saugojimas ir el. verslas yra tarpusavyje panašūs tuo, kad apima schemų kūrimo analizę, kad sudaryti brėžinius, ar net integruotą schemą. Kažkiek skirtingas yra semantinės užklausos apdorojimas – einamojo laiko būdas, kai vartotojas nustato norimą užklausos rezultatą (pvz., SQL sakinys SELECT), o sistema pasirenka, kaip pateikti tą rezultatą (pvz., SQL nustatant FROM ir WHERE sakinius). Vartotojo nurodymas pateikiamas jai suprantamais terminais, kurie gali skirtis nuo elementų pavadinimų, nustatytų duomenų bazėje. Todėl pirmoje užklausos apdorojimo fazėje sistema turi išdėstyti vartotojo užklausos rezultatui nustatytus terminus pagal schemos elementus.

Išdėsčius užklausos rezultatus pagal schemos elementus, sistema turi išvesti apribojimus (pvz., WHERE sakinys), kurie suteikia išdėstymui semantiškumo. Šio apribojimo išvedimo technikos buvo vystomos pastaruosius 20 metų.

3.4 Schemų derinimo įrankis

Norint apibrėžti derinimo įrankį, reikia pasitrinkti įvedamųjų schemų ir rezultatų išdėstymo pateikimo būdą. Panagrinėsime keletą požiūrių į derinimą. Šie požiūriai labai priklauso nuo naudojamos DB schemos informacijos ir kaip ji interpretuojama. Tačiau tai beveik visai nepriklauso nuo informacijos vidinio pateikimo, išskyrus tai, kiek plačiai reikia pateikti dominančią informaciją. Todėl, šio darbo tikslais, mes apibrėžiame *schemą* kaip paprasčiausią *elementų grupę*, susietą kažkokia *struktūra*.

Praktiškai turi būti pasirinktas atskiras aprašymas, toks kaip ER modelis, objektiškai orientuotas (OO) modelis, XML ar tikslūs grafikai[1]. Kiekvienu atveju yra natūralus sudedamųjų pateikimo dalių bei elementų ir struktūros sąvokų atitikimas: esybių ir ryšių ER modelyje, objektiškai orientuotame OO modelyje, XML elementai, subelementai ir IDREF bei taškai ir kampai grafikuose.

Sujungimą apibrėšime kaip *sujungiamų elementų grupę*, kurių kiekvienas nurodo, kad tam tikri S1 elementai atitiktų tam tikrus S2 elementus. Taip pat kiekvienas nagrinėjamas elementas gali turėti su*jungimo išraišką*, kuri nustato, kaip S1 ir S2 elementai yra susiję tarpusavyje. Sujungimo išraiška gali būti tiesioginė, pavyzdžiui, tam tikra S1 elementų funkcija, atitinka S2 elementus, arba gali būti netiesioginė, tai yra ryšys tarp S1 ir S2 elementų kombinacijų. Ji gali naudoti paprastus ryšius tarp skaliarų (pvz., =, ≤), funkcijų (pvz., sudėtis arba tarpusavio sąryšis), ER pobūdžio ryšiai (pvz., yra..., (kažko)

dalis), į grupę orientuoti ryšiai (pvz., perkrovimas, turinys) ar kiti terminai, apibrėžti naudojama išraiškos kalba.

3.1 lentelė Įvedamų schemų pavyzdžiai

S1 elementai	S2 elementai
Klient	Klientas
K#	KlientID
KVardas	Kompanija
Vardas	Kontaktas
Pavardė	Telefonas

1 lentelėje parodytos dvi schemos S1 ir S2, pateikiančios kliento informaciją. Sujungimas tarp S1 ir S2 gali turėti sujungimo elementą, susiejantį Klient.K# atitinkantį Klientas.KlientID. tokios funkcijos išraiška būtų „Klient.K#=Klientas.KlientID“. Elementas su išraiška „Concatenate(Klient.Vardas, Klient.Pavardė) = Klientas.Kontaktas“ aprašo dviejų S1 ir vieno S2 elementų sujungimą.

Apibrėžiame derinimo operaciją kaip funkciją, kuri apima dvi schemas S1 ir S2 kaip įvedamąsias ir pateikia šių dviejų schemų derinį kaip rezultatą, vadintą *derinimo rezultatu*. Kiekvienas derinio elementas derinimo rezultate nusako, kad tam tikri schemos S1 elementai logiškai atitinka, t. y. derinasi su tam tikrais S2 elementus, kur šio atitikimo semantika išreikšta sujungiamų elementų susiejimo išraiška.

Deja, S1 ir S2 elementų derinimo kriterijai paremti euristika, kurią sunku perteikti tikslia matematine formule, kuri padėtų įvykdyti derinimą. Tokiu būdu, siekiant sudaryti sujungimą, kuris atitiktų euristiką bei mūsų suvokimą, rezultatai mūsų netenkina nei praktiškai nei matematiškai.

Čia daugiau skirsime dėmesio derinimo algoritmams, kurie pateikia sujungimą be sudėtingų sujungimo išraiškų. Tuo tikslu pateikiame sujungimą kaip grupių S1 ir S2 naudojant panašumo ryšį **„~“ ,** kur kiekviena pora **„~„** ryšyje pateikia vieną sujungimo elementą. Pavyzdžiui, gautasis derinimo rezultatas iš 1 lentelės DB schemų gali būti „Klient.K#~Klientas.KlientID“, „Klient.Kvardas ~ Klientas.Kompanija“ ir „{Klient.vardas, Klient.Pavardė}~Klientas.Kontaktas“. Pilnas derinimo atnaujinimo nustatytas rezultatas taip pat apimtų kiekvieno elemento sujungimo išraišką, tai yra „Klient.K#=Klientas.KlientID“, „Klient.KVardas=Klientas.Kompanija“ ir „Concatenate(Klient.Vardas, Klient.Pavardė) = Klientas.Kontaktas“. Tai reiškia, kad įtraukus sujungimo išraiškas, mes jas aiškiai pateiksime. Priešingu atveju paprasčiausiai vartosime ~.

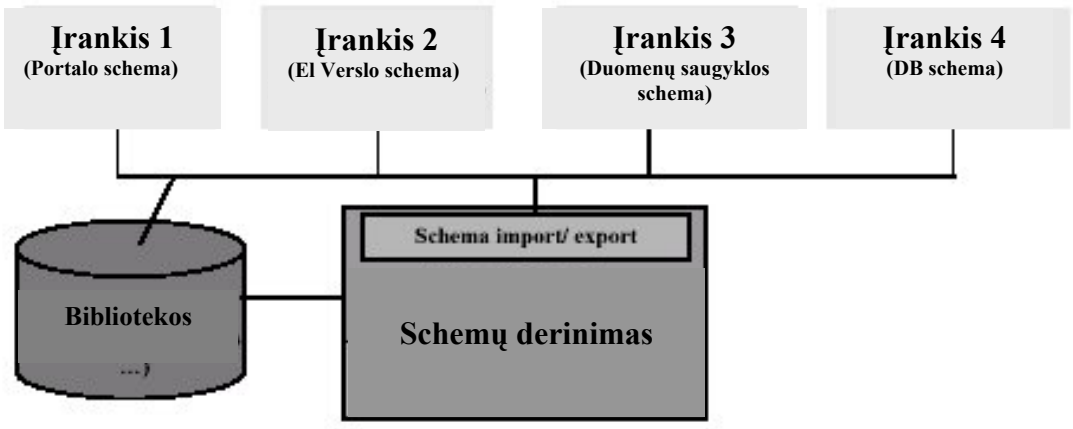
Kaip matysime toliau, kai kurios derinimo realizacijos yra panašios su Sujungimo operacijom duomenų bazėse, kur derinimas ir sujungimas yra dvinarės operacijos, nustatančios atitinkančių elementų poras iš abiejų įvedamų schemų. Žinoma, yra daugybė skirtumų. Derinimas operuoja metaduomenimis (schemos elementais), o Sujungimas – duomenimis (lentelių eilutėmis). Be to, derinimas yra sudėtingesnis nei Sujungimas. Kiekvienas Sujungimo rezultato elementas sujungia tik vieną pirmosios schemos elementą su vienu atitinkamu antrosios schemos elementu, kai derinimo rezultato elementas gali sujungti daugybę elementų iš abiejų DB schemų. Be to, Sujungimo semantika yra nustatoma vienintele palyginimo išraiška (pvz., lygybės sąlyga paprastam jungimui), kuri turi tikti visiems tinkamiems įvedamų schemų elementams. Priešingai, kiekvienas derinimo rezultato elementas gali turėti skirtingą sujungimo išraišką. Iš čia aišku, kad derinimo semantika yra mažiau apribota negu Sujungimo semantika tačiau yra sunkiau nusakoma nuoseklumo požiūriu.

Derinimo ir sujungimo panašumas išsiplečia iki išorinio derinimo operacijų, kurios yra naudingi derinimo atitikmenys, panašiai, kaip Išorinis Jungimas yra Sujungimo atitikmuo. Dešininis (ar kairinis) Išorinis derinimas užtikrina, kad kiekvienas S2 (arba S1) elementas yra sujungtas. Pilnas išorinis derinimas (OuterMatch) užtikrina, kad kiekvienas S1 ir S2 elementas yra sujungtas. Nors išorinio derinimo naudojimas turi kai kurių subtilybių, jo vykdymas yra tiesioginis derinimo išplėtimas: atsižvelgiant į derinimo operacijos algoritmą. Išorinis derinimas gali būti lengvai įvykdomas, pridedant elementus prie pateiktųjų kitaip neapibūdintų S1 ir S2 elementų derinimo rezultatų.

3.5 Bendrojo derinimo architektūra

Apžvelgiant ir lyginant derinimo metodikas, reikia žinoti realizacijos architektūrą. Todėl aprašysime aukšto lygio sandarą bendram derinimo vykdymui.

3.1 paveikslas iliustruoja bendrąją sandarą. Klientai yra su DB schema susiję objektai ir įrankiai iš įvairių sričių, tokių, kaip el. verslas, portalai bei duomenų saugyklos. Kiekvienas klientas naudoja bendrąją derinimo realizaciją, kad automatiškai nustatytų atitikmenis tarp dviejų įeinančių schemų. XML schemos redaktoriai, portalų kūrimo įrankių rinkiniai, duomenų bazių modeliavimo įrankiai ir panašūs gali naudoti bibliotekas, kad pasirinktų iš egzistuojančių schemų, pateiktų 3.1 paveiksle. Derinimo mechanizmas gali taip pat naudotis bibliotekomis ir kita pagalbine informacija, tokia kaip žodynai ir enciklopedijos, kurie padeda atrasti atitikmenis.



3.1 pav. Aukšto lygio architektūros derinimas

Priimsime, kad bendrasis derinimo taikymas pateikia derinamas schemas vienoda vidine išvaizda. Šis vienodas pateikimas žymiai sumažina derinimo sudėtingumą dėl mažesnio skirtingų (heterogeninių) pateiktųjų schemų skaičiaus. Įrankiai, kurie griežtai integruoti į struktūrą, gali tiesiogiai dirbti su vidiniais duomenimis. Kitiems įrankiams reikia importuoti/eksportuoti programas, perduodančias duomenis tarp išeitinės schemos architektūros (tokios, kaip XML, SQL ar UML) ir vidinės architektūros. Semantiką saugantis importavimo įrankis pakeičia įvedamas schemas iš jų prigimtinės struktūros į vidinę struktūrą. Taip pat importavimo įrankis pakeičia iš bendrojo derinimo vykdymo gautus junginius į kiekvienam įrankiui tinkamą formatą. Tai leidžia bendrajam derinimo vykdymui operuoti išskirtinai viduje, nekreipiant dėmesio į išorines duomenų struktūras..

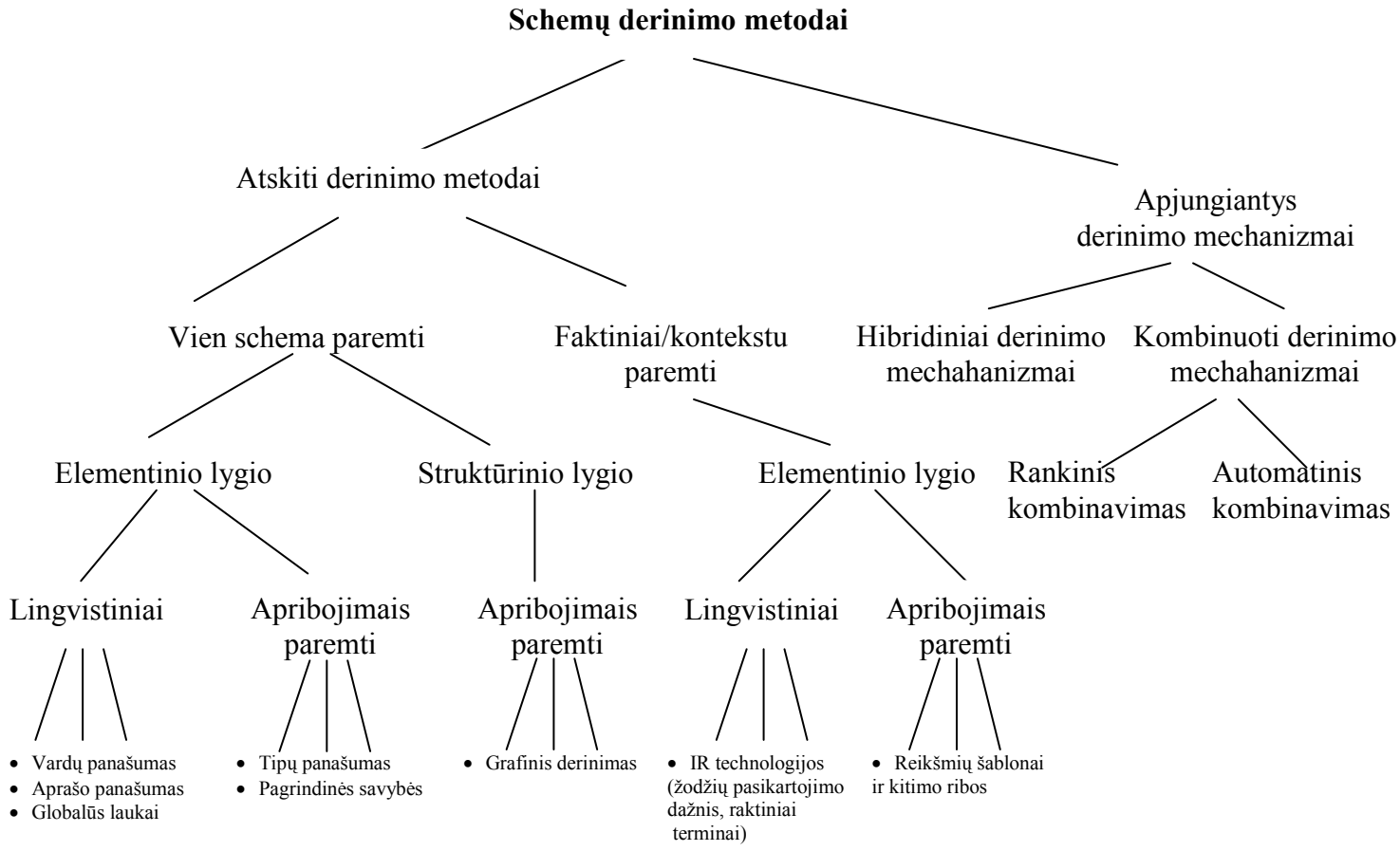
Apibendrinant, neįmanoma visiškai automatiškai nustatyti visus dviejų schemų derinius, pirmiausia dėl to, kad dauguma schemų turi savitą semantiką, kuri veikia derinimo kriterijus, bet formaliai neišreiškiama arba net nenustatoma. Todėl derinimo vykdymas turėtų tiktai nustatyti *galimus derinimus*, kuriuos vartotojas gali priimti, atmesti ar pakeisti. Be to, vartotojas turėtų turėti galimybę nustatyti tuos elementų derinius, kuriems sistema nesugebėjo atrasti patenkinamų galimų derinių.

3.6 DB Schemos derinimo metodų klasifikacija

Šioje dalyje suklasifikuosime pagrindinius metodus, naudojamos schemos derinime. 3.2 paveiksle pavaizduota klasifikacijos schemos dalis kartu su kai kuriais pavyzdžiais.

Derinimo vykdymas gali naudoti daugybę derinimo algoritmų ar *derinimo įrankių*. Tai leidžia pasirinkti derinimo įrankius, priklausančius nuo taikymo srities ir schemos tipo.

Atsižvelgiant į tai, kad norime naudoti daugybę derinimo įrankių, išskirsime dvi problemas. Pirmiausia, galima naudoti atskirus derinimo įrankius, kurių kiekvienas apibrėžia savitą sąsają, paremtą vienu atitinkančiu kriterijumi. Antra, yra atskirų derinimo įrankių kombinacijos, kurios arba naudoja *integruotame hibridiniame derinimo įrankyje* daugybę derinimo kriterijų (pvz., vardo ir tipo lygybė), arba sujungia daugybę derinimo rezultatų, gautų iš skirtingų derinimo algoritmų *sudėtiniame derinimo įrankyje*.



3.2 pav. Schemų derinimo metodų klasifikacija

Atskiriems derinimo įrankiams nagrinėsime tokius daugiausiai ortagonalinius klasifikavimo kriterijus:

- *Faktai − schema:* derinimo metodai gali nagrinėti faktinius duomenis (t. y., duomenų sudėtį) arba tik schemos lygio informaciją.
- *Elementas − struktūrinis derinimas:* derinimas gali būti atliktas atskiriems schemos elementams, tokiems, kaip priedai arba elementų kombinacijoms, tokioms kaip sudėtingos schemų struktūros.

- *Kalba − apribojimai:* derinimo įrankis gali naudoti kalba pagrįstą metodą (pvz., paremtą vardais ar tekstiniais schemos elementų aprašymais) arba konstantomis pagrįstą metodą (pvz., paremtą esme ir ryšiais).
- *Derinimo santykis:* bendras derinimo rezultatas gali susieti vienos schemos vieną ar daugiau elementų su vienu ar daugiau kitos schemos elementų, naudojant ryšius 1:1, 1:n, n:1, n:m. be to, kiekvienas iš siejamų elementų gali sietis su vienu ar daugiau abiejų schemų elementais.
- *Pagalbinė informacija:* dauguma derinimo įrankių pasikliauja ne tik įvedamomis schemomis S1 ir S2, bet taip pat pagalbine informacija, tokia kaip žodynai, globalinės schemos, ankstesni derinimo sprendimai bei vartotojo įvesta informacija.

Reikia atkreipti dėmesį, kad pateiktoji klasifikacija neskiria skirtingų schemų tipų (reliacinių, XML, orientuotų į objektą ir kt.) bei jų vidinio pateikimo todėl, kad algoritmai daugiausiai priklauso nuo informacijos rūšies, kurią jos naudoja, o ne nuo jos pateikimo.

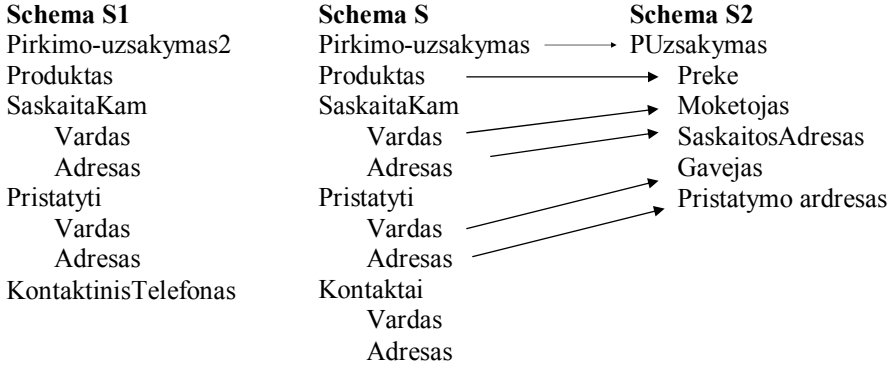
3.7 Pakartotinis schemos ir sujungimo informacijos naudojimas

Jau buvo paminėtas pagalbinės informacijos naudojimas kartu su įvedamomis schemomis. Tai žodynai, enciklopedijos ir vartotojo pateikti deriniai ar netinkama informacija. Kitas būdas yra pagalbinės informacijos naudojimas pagerinti derinimo efektyvumui, pakartotinai naudojant įprastus schemos komponentus ir anksčiau nustatytą sujungimą. Į pakartotinį naudojimą orientuoti metodai yra daug žadantys, kadangi tikimasi, jog schemos, kurias reikia suderinti, gali būti panašios į jau derintas. Pavyzdžiui, el. verslo struktūros pastoviai pasikartoja skirtingose žinutėse (pvz., adreso ar vardo laukuose).

Vardų naudojimas iš vardų lauko ar žodyno naudojimas jau yra orientuoti į pakartotinį naudojimą. Bendresnis metodas yra naudoti ne tik globaliai nustatytus vardus bet ir schemų fragmentus, apimančius tokias savybes, kaip duomenų tipas, raktai ir konstantos. Tai ypač pasiteisina dažnai vartojamiems elementams. Kol pasaulis nesutaria dėl tokių schemų, jos gali būti apibūdintos įmonėms, jų prekybos partneriams ar panašoms organizacijoms, kas sumažintų kaitos lygį. Schemų redaktoriai turėtų prieiti prie tokių bibliotekų, kas paskatintų schemų fragmentų ir nustatytų terminų pakartotinį naudojimą. Tokiu būdu pakartotinai naudojami elementai privalėtų turėti išeitinės bibliotekos ID, kad derinimo įrankis galėtų paprastai nustatyti ir suderinti schemų fragmentus.

Tolimesnis apibendrinantis metodas yra sujungimo pakartotinis naudojimas. Mes norime panaudoti anksčiau nustatytus elementinio lygio derinius, kurie paprasčiausiai

galėtų būti pridėti į žinynus. Taip pat norima pavartoti struktūras, kurios naudingos derinant skirtingas bet panašias schemas į vienodą tikslinę schemą, taip gali atsitikti integruojant naujus šaltinius į duomenų saugyklas ar skaitmenines bibliotekas. Pavyzdžiui, tai naudinga, jei schema S1 turi būti sujungta su schemą S2, ir jau buvo sujungta su kita S schema. Jei S1 yra panašesnė į S, tuomet tai gali supaprastinti automatinį galimų derinių parinkimą panaudojant egzistuojančius rezultatus (S, S2), nors kartais tokio perkėlimo neįmanoma atlikti. Tai taip pat leidžia panaudoti rankiniu būdu nustatytus derinius.



3.3 pav. Egzistuojančio derinimo pakartotinio panaudojimo scenarijus

Tokio pakartotinio naudojimo pavyzdys pateiktas 3.3 paveiksle įsigijimo užsakymo schemoms. Mes jau turime derinio rezultatą tarp S ir S2, parodytą rodyklėmis. Nauja įsigijimo užsakymų schema S1 yra labai panaši S. Tokiu būdu kiekvienas struktūros S1 elementas turi atitinkantį ar visiškai derantį elementą, todėl galime naudoti egzistuojantį S ir S2 sujungimą. Tokiu atveju mes galime panaudoti visus derinius.

Toks ankstesnių derinių pakartotinis naudojimas galimas tik daliai naujos schemos. Iš čia kyla pagrindinė problema, kaip nustatyti, kuri naujos schemos dalis yra panaši į kurią ankstesnės suderintos schemos dalį – tai pati pradinė derinimo problema. Dar daugiau, nustatyti panašumo kriterijai ankstesniam derinimo uždaviniui gali priklausyti nuo pritaikymo srities tiek, kad jų panaudojimas turėtų būti apribotas tik panašiems pritaikymams. Pavyzdžiui, Salary ir Income gali būti laikomos identiškomis pritaikant atsiskaitymams, bet ne mokesčių ataskaitoms. Kiek žinoma literatūroje tie pakartotinio vartojimo metodai dar niekur nebuvo nagrinėti.

3.8 Įvairių derinimo įrankių sujungimas

Apžvelgėme keletą derinimo įrankių tipų ir daugybę jų derinių. Kiekvienas naudoja skirtingą informaciją ir taip turi skirtingą pritaikymą bei vertę derinimo uždaviniui, todėl

derinimo įrankis, kuris naudoja tik vieną metodą, greičiausiai neras tiek daug gerų galimų derinių kaip kelis metodus apimantis derinimo įrankis. Tai gali būti įvykdyta dviem būdais: hibridiniu derinimo įrankiu, kuris integruoja daugybę derinimo kriterijų, ir sudėtiniu derinimo įrankiu, kuris sujungia atskirai veikiančių derinimo įrankių rezultatus. Daugybės derinimo metodų sujungimas taip pat suteikia galimybę juos įvertinti iš karto arba peiliui.

Hibridiniai derinimo įrankiai tiesiogiai sujungia keletą derinimo metodų, paremtų daugybe kriterijų ar informacijos šaltinių (pvz., vardų derinimo ir žinyno sujungimas su duomenų tipo suderinamumu), nustatymui. Jie turėtų pateikti geresnius galimus derinius, kartu ir geresnį rezultatą, negu atskirų derinimo įrankių vykdymas. Gali būti pagerintas efektyvumas, nes menkaverčiai deriniai, derantys tik pagal vieną iš kelių kriterijų, gali būti anksti atmesti, ir tuo būdu išspręsti uždaviniai. Struktūrinio lygio derinimas taip pat pagerėja, sujungus jį su kitais metodais, tokiais, kaip derinimas pagal pavadinimą. Vienas būdų sujungti struktūrinį ir elementinį lygio derinimą yra pirmąjį naudoti dalinio sujungimo gavimui ir antrąjį naudoti jau. pilnam sujungimo sudarymui.

Hibridinis derinimo įrankis gali pasiūlyti geresnį atlikimą negu daugelio derinimo įrankių vykdymas, sumažinant schemos tikrinimų skaičių. Pavyzdžiui, elementinio lygio derinimas su hibridiniu derinimo įrankiu gali patikrinti daugelį kriterijų vienu metu kiekvienam S2 elementui prieš pereidamas prie kito S2 elemento.

Iš kitos pusės *sudėtinis derinimo įrankis*, kuris sujungia kelių atskirai veikiančių derinimo įrankių rezultatus, apima ir hibridinius derinimo įrankius. Tokia galimybė sujungti derinimo įrankius yra lankstesnė negu hibridinio derinimo įrankio. Hibridinis derinimo įrankis dažniausiai naudoja tvirtai nustatytą atskirų derinimo technikų, vykdomų iškart arba paeiliui, sujungimą. Priešingai, sudėtinis derinimo įrankis leidžia pasirinkti iš modulinių derinimo įrankių remiantis, pavyzdžiui, taikymo sritimi ar schemos kalba (pvz., gali būti naudojami skirtingi metodai struktūrizuotoms ir pusiau struktūrizuotoms schemoms). Dar daugiau, sudėtinis derinimo įrankis turėtų leisti lankstų derinimo įrankių eiliškumo parinkimą taip, kad jie veiktų vienu metu ar iš eilės. Pastaruoju atveju, pirmojo derinimo įrankio derinimo rezultatas ir naudojamas ir išplečiamas antrojo derinimo įrankio, kas leidžia pasiekti bendrų derinimo rezultatų pagerinimą.

Derinimo įrankių parinkimas ir jų vykdymo eiliškumo nustatymas bei atskirai nustatytų rezultatų sujungimas gali būti įvykdytas automatiškai naudojant patį derinimo įrankį arba rankiniu būdu. Automatinis metodas sumažina žmogaus darbo kiekį, bet tuomet sunkiai pasiekiamas bendras sprendimas įvairioms pritaikymo sritims (nors metodas ir gali

būti kontroliuojamas keitimo parametrais). Kitas variantas − vartotojas gali tiesiogiai pasirinkti norimas derinimo metodikas, jų vykdymo tvarką ir kaip sujungti rezultatus. Toks rankinis metodas yra lengviau taikomas ir leidžia labiau kontroliuoti proceso eigą. Kaip buvo minėta anksčiau vartotojo sąveika yra būtina bet kuriuo atveju, nes derinimo įrankio taikymas tik nustato galimus derinius, kuriuos vartotojas gali priimti, atmesti ar pakeisti.

Sudėtingų derinimo užduočių vykdymui derinimo taikymas turėtų būti vykdomas lygiagrečiai su vartotojo įsikišimu. Sudėtiniame derinimo metode vartotojo pateikti deriniai gali būti laikomi kaip atskiras derinimo įrankis, kuris pateikia duomenis automatiniams derinimo įrankiams. Tačiau derinimo įrankiai vis tiek turėtų atskirti vartotojo pateiktus derinius ir jų nekeisti bei sutelkti dėmesį į nederintas įvedamos schemos dalis.

4 ITERACINIS REINŽINERIJOS METODAS

Kaip daugeliu atveju būna, vieni ar kiti sistemos pasenimo požymiai verčia perprojektuoti seną sistemą, o ne atsisakyti sistemos, kadangi senoji sistema beveik visada yra suprojektuota ir realizuota per eilę metų, sudėjus ten labai daug žinių. Kaip taisyklė, senosios sistemos perpratimas yra vienas pagrindinių faktorių vedančių prie sėkmingos sistemos reinžinerijos ir tolesnio sklandaus įmonės funkcionavimo. Teisingai pasirinktas reinžinerijos kelias gali pastebimai sumažinti įmonės išlaidas ir be sutrikimų atnaujinti senąją sistemą.

Taip jau nusistovėjo, kad daugelis literatūroje aprašomų sistemų reinžinerijos metodų apima visą programinės įrangos sistemą. Dėl šios priežasties visa sistema turi būti užšaldyta esamoje būsenoje , kol reinžinerijos procesas bus baigtas . Kitaip tariant nieko negalima keisti šio periodo metu. Taigi bendrai paėmus, jeigu bus atliekami senosios sistemos tobulinimo veiksmai, gausis taip, kad senoji ir naujoji sutemos nebeatitiks viena kitos ir programuotojai turės pradėti visą procesą iš naujo. Šioje situacijoje atsiranda uždaras ratas (ciklas) tarp senosios sistemos palaikymo ir reinžinerijos proceso.

Kai kuriais atvejais tai problemai išvengti pakanka interpretuoti, kad senoji informacinė sistema yra juodoji dėžė ir jos nemodifikuojant vykdyti reinžinerijos procesą. Tačiau šis metodas nepasiteisina, kai reinžinerijos procesas užsitęsia ir nebeišeina nekeisti senosios IS.

Labai dažnai senoji sistema yra tokia didelė ir turi labai svarbių funkcionalumo aspektų, kad tiesiog jos pakeitimas naująja yra labai rizikingas. Italijos mokyklos atstovai [1] siūlo iteracinį senosios sistemos reinžinerijos būdą. Kiekvienos iteracijos metu pridedami skirtingo dydžio ir sudėtingumo komponentų patobulinimai, kai tuo tarpu senoji sistema gali dirbti visu savo pajėgumu ir funkcionalumu viso reinžinerijos proceso metu.

Kadangi atkarpa, per kurią perprojektuojamas vienas komponentas, yra sąlyginai trumpa, tai laikas, sugaištas kiekvieną kartą įterpiant naują komponentą, iteracinio reinžinerijos proceso metu yra labai trumpas. Praktikoje, perprojektuojant pastovius komponentus nereikia skubėti, nes tie komponentai nesikeičia naudojimo metu, tačiau nepastovius komponentus reikia apdoroti greitai, nes proceso eigoje jie jau gali pasikeisti. Tokiu atveju geriausias sprendimas yra sumažinti nepastovių komponentų dydžius prieš juos perprojektuojant.

Kita problema, su kuria susiduriama senų informacinių sistemų reinžinerijos metu yra tai, kad pasenusios būna ir duomenų bazių struktūros, todėl taip pat reikia pašalinti tuos

senosios duomenų bazės struktūros trūkumus, kad naujoji sistema veiktų be senosios trūkumų.

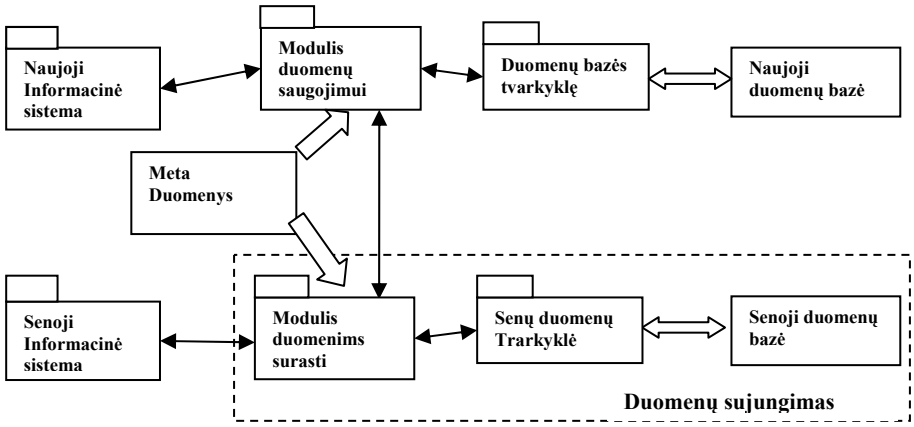
Toliau pasiūlytas metodas aprašo, kaip perprojektuoti duomenų bazės struktūra, nekreipiant dėmesio į aplikacijas, kurios naudojasi ta duomenų baze, taip kad senoji ir naujoji duomenų bazės galėtų egzistuoti kartu, būtų naudojamos senųjų bei naujai sukurtų programų. Reinžinerijos proceso metu senosios programos ir duomenų bazė pamažu nyksta, o naujoji programa tobulėja ir naujoji duomenų bazė didėja.

Kadangi mus labiau domina duomenų bazės reinžinerija šiame etape bus akcentuojami du reinžinerijos aspektai susiję su duomenų reinžinerija.

Remiantis iteraciniu reinžinerijos metodu [2], būtinai reikia užtikrinti senosios informacinės sistemos ir naujosios sistemos lygiagretų egzistavimą viso reinžinerijos proceso metu.

4.1 paveikslėlyje kaip tik ir pavaizduotos senoji informacinė sistema su senąja duomenų baze. Senoji informacinė sistema yra pilnai veikianti programinės įrangos sistema, kuri palaiko kai kurias organizacijos biznio funkcijas. Ši sistema yra svarbus taisyklių ir žinių rinkinys, sukauptas laikui bėgant. Senoji duomenų bazė sudaryta iš duomenų, kuriuos naudoja, pildo, keičia senoji sistema. Tiek sistema tiek senoji duomenų bazė šiuo atveju yra pasenusios, jų palaikymas kainuoja didelius pinigus ir neatsiperka ekonomiškai.

Naujoji duomenų bazė turi visus duomenis , naudojamus įmonės verslo procese.



4.1 pav. Lygiagretus naujosios ir senosios IS egzistavimas reinžinerijos metu

Naujoji duomenų bazių valdymo sistema išvis būtų bevertė, jeigu ji būtų paprasčiausias senosios atkartojimas. Tada įvedamas vadinamasis normalizacijos kriterijus, kuris kaip tik

ir pagerina naujosios sistemos palaikymą ir evoliucijos procesą. Reinžinerijos proceso metu kartais gali reikėti suprojektuoti visai naujas aplikacijas pagrindinės aplikacijos sferoje. Jas galima pavadinti moderniomis aplikacijomis, nes yra toje pačioje sferoje kaip ir senoji sistema, dalinasi tais pačiais duomenimis, ir kartais net praplečia senąją duomenų bazę naujais duomenimis. Tokiu būdu moderniosios aplikacijos duomenų bazė bus suintegruota į naująją duomenų bazę. Duomenų bazės evoliucija visą laiką bus visiškai prieinama tiek senajai informacinei sistemai, tiek naujajai, todėl kad komponentai Data Banker ir Data Locator saugo jos fizinę struktūrą. Tokio reinžinerijos proceso rezultatas bus duomenų bazė, kuri priklausė senajai duomenų bazei. Rekonstruota ir Moderni sistema duomenis pasiekia pateikdamos užklausas Data Banker komponentui. Šios užklausos apibrėžia ne tik reikalinga servisą (read, write, delete,...) , bet taip pat apibrėžia duomenis, su kuriais servisas turėtų operuoti. Data Banker komponentas analizuoja serviso užklausos struktūrą, interpretuoja jos turinį ir realizuoja kreipimąsi į fizinę duomenų bazę, kad pateikti rezultatus užklausai. Pavyzdžiui serviso užklausa yra atidaryti duomenų bazė ir inicializuoti einamo įrašo poziciją: tai atliekama serviso pavadinimu *open_service* pagalba.

Kaip tik tokios užklausos išraiška pavaizduota 4.2 paveiksliuke, kur pasviręs tekstas . prasidedantis žvaigždute(*) yra komentarai.

Open_service= Atidaro nustatytą duomenų saugyklą ir nustato esamą
Access_mode+entity_name+entity_exists+open_mode+optional_clause
Kur:
Access_mode – Aprašo prisijungimo prie saugyklos metodą (dynamic_access/ random_access/sequential_access)
entity_name – Duomenų saugyklos vardas
entity_exists – Požymis, ar yra duomenų saugykla, tikrinamas prieš programos vykdyma.
open_mode – Duomenų saugyklos atidarymo metodas.

4.2 pav. Serviso užklausos pavyzdys

Kai perprojektuotai ar naujai suprojektuotai sistemai reikia atidaryti duomenų bazę, ji tiesiog pateikia užklausą *open_service* , nurodant teisingas parametrų reikšmes, kurios apibrėžia prieigos prie esybės rėžimą, bei jos pavadinimą. Data Banker komponentas gavęs

informaciją, tiesiog atidaro reikiamą lentelę reikiamoje duomenų bazėje. Jei visos lentelės atidaromos korektiškai, tai požymis *file_exists* pereina į būseną *true.*

Taigi, jeigu pasikeitė duomenų bazės struktūra, visą laiką pirmiausia tai atvaizduojama Data locator ir Data Banker komponentų parametruose. Pati aplikacija turi būti keičiama tik tada, kai esybes aplikacijos srities ribose pasikeitė. *Data Mapping* architektūra užtikrina lygiagretų senųjų ir perprojektuotų duomenų egzistavimą. Kai senosios programos komponentai yra perprojektuoti , jie jau nebesikreipia į senąją duomenų bazę, o kreipiasi į naująją, todėl reinžinerijos proceso pabaigoje senoji duomenų bazė išvis nebenaudojama.

Pavyzdžiui 1 lentelėje pavaizduoti duomenys likę senoje duomenų bazėje. Kiekvienas duomuo yra apibrėžtas jo pavadinimu, duomenų failo vardu, kur pirminiai duomenys buvo įrašyti senojoje sistemoje ir duomenų aprašymas, kad paaiškinti identifikatoriaus reikšmę. Šioje išraiškoje duomenys, pavadinimu datfor−arrot−far apibrėžia vaistų tiekėją, duomenys datfor−arrot−par reiškia tą patį, tik tiekėjas netiekia vaistų, o tiekia kitas prekes. Disconto yra nuolaida, suteikiama užsakytoms prekėms.

4.1 lentelė Senų duomenų pavyzdys.

Laukas	Duomenų aprašymas
Datfor−arrot−far	**Požymis, ar užsakymo dydis didesnis už nustatytą kiekį**
Datfor−arrot−par	**Tas pats požymis, tik apie kitą produktų grupę**
Datsconto	**Nuolaidos dydis, jei kiekis viršija nustatytą kiekį**

Kai programai senojoje sistemoje reikia duomenų, ji kreipiasi į senąją duomenų bazę naudodama *Data Locator* komponentą ir funkcionuoja kaip įprastai. *Data Locator* komponentas yra, duomenų sujungimo komponentas, kuris konvertuoja duomenis iš senojo formato į formatą, reikalingą naujajai sistemai ir atvirkščiai. Senoji sistema gali prieiti prie naujosios duomenų struktūros taip pat pasinaudodama *Data Locator* komponentu. *Data Locator* komponentas veikia senosios sistemos aptarnavimui, taip pat kaip *Data Banker* komponentas veikia su naujai perprojektuota sistema. *Data Locator* kaip ir *Data Banker* analizuoja serviso užklausos struktūrą, interpretuoja jos turinį ir kreipiasi į fizinę duomenų bazę, kad įvykdytų serviso užklausą. *Metadata* komponentas, esantis tarp Data Locator ir Data Banker yra duomenų bazė, kurioje saugoma visa informacija apie senąją duomenų bazę ir atitinkamus duomenis naujojoje duomenų struktūroje.

4.1 Iteracinio reinžinerijos proceso metu vykdomų etapų aprašymas

Čia aprašomas iteracinis reinžinerijos procesas siūlo sistemą skaidyti į keletą dalių ir kiekvienai iš jų pritaikyti žemiau išvardintus procesus:

- **Esamų duomenų analizė**

Šio etapo metu atliekamas senos sistemos duomenų identifikacija, analizė ir interpretacija ir tarpusavio ryšiai. Pagal kai kuriuos šaltinius išskiriamos keturios duomenų grupės:

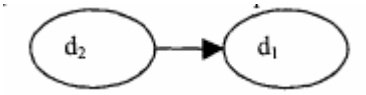
- Konceptualūs duomenys – duomenys , kurie yra specifiniai duotajai sistemai ir aprašo specifines sistemos idėjas. Jų reikšmė gan paprasta aplikacijos vartotojams, bet palyginus sudėtinga reinžinerijos specialistams. Reikalingos geros sistemos žinios, kad teisingai interpretuoti pateiktus duomenis.
- Kontroliniai duomenys − tai duomenys, sukuriami programoje, arba įrašomi atsiradus realaus pasaulio veiksniams ar įvykiams .
- Struktūriniai duomenys − duomenys naudojami organizuoti ir palaikyti sistemos duomenų struktūras.
- Apskaičiuoti duomenys − duomenys gauti po aplikacijos skaičiavimų.

Šios fazės eigoje gali atsirasti duomenų sinonimų. Konceptualūs duomenys bus apibūdinti sekančioje fazėje, o kontroliniai , struktūriniai ir apskaičiuoti duomenys tiesiog perkeliami iš senosios į naująją duomenų bazę.

- **Duomenų perprojektavimas**

Šiame etape perprojektuojama duomenų organizacija, nustatant ryšis tarp jų. Bendru atveju, kaip parodyta 4.3 paveiksle, priimsime, kad *d1* ir *d2* yra pavieniai arba sudėtiniai duomenų laukai. Tada:

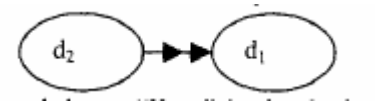
- Yra vienareikšmis ryšys tarp *d2* ir *d1* , jei kiekvienai *d2* reikšmei gaunama viena ir tik viena d1 reikšmė.



4.3 pav. Duomenų vienareikšmis ryšys

Čia d2 yra raktinė aibė, kurioje visos reikšmės yra unikalios ir ne nulinės.

- Taip pat yra daugiareikšmė priklausomybė tarp *d2* ir *d1* , kai kiekvienai *d2* reikšmei gali būti gaunamos viena ar kelios *d1* reikšmės



4.4 pav. Daugiareikšmis duomenų ryšys

Čia *d2* ir *d1* reikšmės turi būti ne nuliai ir unikalios ir kiekviena *d2* ir *d1* kombinacija turi būti unikali. Taip pat kiekviena *d2* reikšmė aprašo *d1* duomenų rinkinį.

Kai kurios funkcijos reikalauja parašymo, kad d1 reikšmės rodo į d2 reikšmes, o kartais būna atvirkščiai d2 rodo į d1 reikšmes. Čia reinžinerijos specialistas turi būti atidus ir pasirinkti reikiamą variantą, pagal esamos sistemos reikalavimus. Ryšiai tarp visų duomenų, kurie turi būti įtraukiami , yra aprašomi ryšių diagramomis. Gauta duomenų bazės forma pagal algoritmus reikia normalizuoti. Normalizacijos proceso metu programinės įrangos inžinieriui gali reikti pridėti papildomus laukus ir raktus duomenų bazėje. Po to gali būti panaudoti duomenys iš senosios duomenų bazės. Kadangi reinžinerijos procesas yra iteratyvus, tai ryšių diagrama, gauta iš senosios duomenų bazės turėtų būti integruota į perprojektuotą ryšių diagramą.

- **Senųjų programų pritaikymas naudoti naują duomenų bazę**

Kiekviena senoji programa, kuri naudos naujos duomenų bazės duomenis, turi būti pritaikyta, kad ji tuos duomenis sėkmingai paimtų ir jeigu reikia, atliktu kitus veiksmus su jais. Tai reiškia, kad kiekviena programa individualiai turi būti peržiūrėta ir pakeistos jungimosi instrukcijos prie naujosios duomenų bazės.

- **Duomenų perkėlimas iš senosios į naująją duomenų bazę**

Senosios duomenų bazės duomenys palaipsniui perkeliami į naująją duomenų bazę. Jeigu nusprendžiama , kad kažkokia duomenų dalis nereikalinga naujajai sistemai, tai tokie duomenys tiesiog paliekami senojoje sistemoje.

- **Sutapimo testas**

Informacinė sistema, praėjusi anksčiau aprašytus veiksmus jau paruošta eksploatacijai. Ji turėtų turėti visą senosios sistemos ir dar naujai įdiegtą funkcionalumą. Tam patikrinti, reikia praleisti testus, palyginančius naujosios ir senosios sistemų vienokiu ar kitokiu atveju.

- **Perdarymas**

Kiekvieną atsiradusią klaidą, aptiktą testuojant reikia nuodugniai išanalizuoti ir naujoji sistema turi būti pataisyta, kad pašalinti tuos trūkumus. Po šio žingsnio vėl reiktų praleisti sutapimo testus ir tol kartoti tuos veiksmus, kol klaidų nebebus pastebėta.

- **Iteracija nuo pirmo žingsnio**

Pabaigai, imamas kitas perprojektuojamos sistemos komponentas ir procesas pradedamas iš naujo pradedant nuo 1 žingsnio. Iteracijos metu, programuotojas gali pastebėti, kad visos duomenų dalys, kuriomis naudojais viena ar kelios programos, jau yra perkeltos, tos programos jau gali būti rekonstruotos. Iteracinį reinžinerijos metodas taip pat įgalina jį taikyti ir procedūroms.

4.2 Pasenusios duomenų bazės simptomų pašalinimas

Čia aprašytas metodas pašalina keletą požymių, pagal kuriuos galima sakyti, kad duomenų bazė jau yra pasenusi [6]. Šie požymiai yra : Duomenų užteršimas, Laikini failai ir patologiniai (nenormalūs) failai.

Duomenų bazės užteršimas atsiranda, kai duomenų bazėje saugomi niekam nebereikalingi seni įrašai ar duomenys, kurie visai nereikalingi programos funkcionavimui. Duomenų perteklius ar dubliavimas gali atsirasti dėl :

Semantiškai perteklinių duomenų, t. y. kai skirtingos programos įrašo tokius pat arba panašius duomenis.

Skaičiavimo perteklinių duomenų t. y. duomenys gali būti išskaičiuoti iš toje pačioje duomenų bazėje esančių duomenų.

Struktūriniai duomenys tai duomenys, kurie buvo naudojami palaikyti pačiai senosios duomenų bazės struktūrai, kurie buvo įdėti pačioje senos DB projektavimo pradžioje ir dabar nereikalingi.

Kontroliniai duomenys tai duomenys, reikalingi senosios sistemos komponentų tarpusavio komunikacijai.

Duomenų užteršimas pablogina sistemos ir jos komponentų bei jų tarpusavio sąveikos supratimą, reinžinerijos proceso metu.

Visi nereikalingi duomenys turėtų būti pašalinti iš duomenų bazės po reinžinerijos. Tačiau kai kurie iš jų turi likti senojoje duomenų bazes struktūroje, kad užtikrinti stabilų senosios duomenų bazės bei informacinės sistemos stabilų veikimą reinžinerijos proceso metu. Kartais ir naujojoje duomenų bazėje pertekliniai duomenys yra naudingi. Nesvarbu, kad jie gal ir sulėtina naujos informacines sistemos darbą, tačiau jų pagalba kartais yra realizuojami kokie nors funkciniai reikalavimai.

Laikini failai yra failai kuruos sistema sukūrė ir skaito, bet neatnaujina. Tokių failų buvimas parodo, kad sistema jau pasenusi.

Tokiu duomenų atsiradimo pavyzdys yra, kai viena programa yra sukuriama ir ji papildo duomenų bazę duomenimis reikalingais jos funkcionavimui. Paskui, tarkim, ta programa yra tobulinama ir ji jau generuoja kitus duomenų rinkinius, kurie pakartotinai įrašomi į duomenų bazę, pirmųjų neištrynus.

Kitas pavyzdys, kaip atsiranda laikini failai, yra, kai sukuriama programa, kuri naudoja duomenis iš duomenų bazės, o po to sukuriamos dar kelios programos, kurios, kad nereiktų visa laiką kreiptis į duomenų bazę, susikuria laikinus duomenų failus. Tos programos nebenaudojamos, o kartais ir ištrinamos, o laikinieji duomenų failai lieka.

4.3 Iteracinio metodo apibendrinimas

Čia buvo aprašytas reinžinerijos metodas, pagal kurį visa IS suskaidoma į dalis ir reinžinerija atliekama vienam ar keletui komponentų vienu metu. Kadangi vienam komponentui suprojektuoti laiko sąnaudos yra gan nedidelės, tai bendru atveju informacinės sistemos reinžinerija šiuo metodu turėtų užimti mažiau laiko.

Antra vertus, iteracinis duomenų bazės reinžinerijos metodas leidžia vienu metu naudoti ir senąją ir naująją duomenų bazes, tuo atveju tereikia keleto pakeitimų senojoje

sistemoje, kad ji galėtų naudoti ir naująją duomenų bazę. Taip pat dar yra galimybė papildyti senąją informacinę sistemą naujais komponentais ir ji galės naudotis naująja duomenų baze.

Kai informacinė sistema galutinai atnaujinama, duomenų bazė turi būti atnaujinta naujais duomenimis, tada nauji duomenys jau rašomi tiesiai į naująją sistemą. Tada jau ir naujosios informacinės sistemos visos funkcijos kreipiasi į vieną duomenų bazę. Naujajai informacinei sistemai visai nereikia žinoti kur yra duomenys, tiesiog tai už ją padaro komponentas *Data Locator* .

Šio metodo privalumai:

Iteracijos panaudojimas reinžinerijos procese, dviejų duomenų bazių, senosios ir naujosios lygiagretus egzistavimas, unikali nauja duomenų bazė, kurios dėka galima kurti naujas programas toje pačioje aplikacijos srityje, visų naujosios duomenų valdymo sistemos servisų eksploatacija, visų duomenų bazės senėjimo požymių pašalinimas.

Metodo trūkumai:

Reikia sukurti ir naudoti komponentą *Data Locator*, kuris nebenaudojamas, kai reinžinerijos procesas baigtas. Trūkumas minimizuojamas, pakartotinai naudojant programas sudėtas į *Data Banker* komponentą . Taip pat reikia palaikyti senąją duomenų bazę iki pat reinžinerijos proceso pabaigos.

4.4 Panašūs į iteracinį metodai

Labiausiai panašūs metodai į anksčiau aprašytąjį yra Chicken Little metodas ir Butterfly metodas[3,4].

Šie metodai, kaip ir aukščiau parašytasis, paremti prielaida, kad duomenys senojoje informacinėje sistemoje yra svarbiausia sistemos dalis ir naujosios sistemos požiūriu duomenys nebus keičiami, tik bus keičiama jų semantika ir schema.

Pagal Chicken Little Metodą senoji sistema gali bendrauti su naująja perkėlimo metu, naudojant tarpinį modulį, vadinamą Gateway. Chicken Little naudoja tiesioginį Gateway nukreipti užklausas į duomenų bazę, bei transformuoti rezultatus, gautus iš duomenų bazės, senajai informacinei sistemai. Šiam sprendimo variantui reikia senąją duomenų bazę duplikuoti ir perprojektuoti, kad ji atitiktu naujosios sistemos reikalavimus. Kreipiantis į duomenų bazę pagal šį metodą reikia kreiptis programa turi turėti priėjima prie abiejų duomenų bazių. Tačiau daug modernių savybių, tokių kaip ryšiai, trigeriai, vientisumas, nuoseklumas nepalaikomi senose duomenų bazėse, todėl negali būti naudojami, todėl

sunku atnaujinti sistemos vientisumą. Kinų autorių pasiūlytas metodas [1,5,10] tiesiog perkelia senus duomenis į naująją duomenų bazę, nenaudojant duplikavimo. Šio atveju vienas komponentas nusako iš kur duomenys buvo paimti, iš senosios duomenų bazės ar iš naujai suprojektuotos. Kadangi duomenų bazės nėra duplikuojamos, atkrenta duomenų sudvejinimo problema ir klausimas kaip nustatyti iš kurios duomenų bazės tie duomenys paimti.

Butterfly metodologijoje tiek senoji, tiek naujoji duomenų bazės gali egzistuoti, bet jos nesidalina duomenimis tarpusavyje. Butterfly metodologija koncentruojasi į senų duomenų perkėlimą ir kuriama nauja sistema traktuojama kaip visiškai atskiras procesas. Senoji duomenų bazė perkeliama į naująją ir tada senoji užšaldoma ir naudojama tik skaitymui. Visi duomenų pasikeitimai saugomi laikinoje pagalbinėje saugykloje. Taigi kiekvieną kartą, kai reikia duomenų, sistema turi kreiptis į abi duomenų bazes ir dar patikrinti laikinojoje pagalbinėje saugykloje, ar jie nepasikeitę. Tokiu atveju atsiranda rizika, kad reinžinerijos proceso metu ieškant duomenų vykdoma daug transakcijų ir tai gali labai pailginti paieškos laiką. Tada laikinoji duomenų saugykla turės tendenciją didėti diena iš dienos.

Šio metodo privalumas yra tas, kad duomenų struktūra yra perprojektuojama, vietoj to, kad tiesiog būtų perrašyta pasinaudojant kitais metodais. Taip pat išvengiama pablogėjusios ir pasenusios duomenų bazės struktūros simptomų perkėlimo į naująją duomenų bazę, o jeigu ji būtų tiesiog perrašyta pasinaudojant paprastesniais metodais, šios ydos liktų.

5 KONCEPCINIS DUOMENŲ BAZĖS REINŽINERIJOS MODELIS

Atsižvelgiant į išnagrinėtus duomenų bazių reinžinerijos metodus galima gauti vieną modelį, kuris gal ne toks sudėtingas, kaip anksčiau aprašyti, bet realizuoja visus keliamus reikalavimus vidutinės duomenų bazės reinžinerijai atlikti. Bet koks pasenusios duomenų bazės reinžinerijos procesas susideda iš dviejų stambių etapų: atvirkštines inžinerijos ir tiesioginės inžinerijos šiuo atveju: sugeneruotos schemos realizavimas naujojoje DBVS. Taigi pirmiausia aprašysime atvirkštinės duomenų bazės inžinerijos metodą.

Beveik visi atvirkštinės inžinerijos metodai yra apriboti tam tikromis būtinomis sąlygomis, todėl ir mes jas iškelsime:

- visos konceptualiosios specifikacijos turi būti transliuojamos į duomenų struktūras ir apribojimus, pavyzdžiui, esant reliacinei duomenų bazei visos poaibio priklausomybės ir pirminiai raktai turi būti aiškiai ir tiksliai apibrėžti;
- pradinė schema negali turėti painių objektų ir ryšių struktūrų;
- pagal tam tikras taisykles parinkti vardai (pavyzdžiui, išorinio rakto ir rakto, į kurį tas išorinis raktas turi nuorodą, vardai turi sutapti);
- tariama, kad schema nebuvo optimizuota ar blogai suprojektuota.

Taip pat atliekant duomenų bazės reinžineriją reikia užsiduoti kažkokius kriterijus, tikslus, siekius, kurių bus siekiama viso reinžinerijos proceso metu. Pagrindinis atvirkštinės inžinerijos tikslas yra gauti geros kokybės duomenų bazę. Schemos transformavimo kokybė skiriasi nuo schemų ar duomenų modeliavimo kokybės, kuri dažniausiai remiasi intuityviomis sąvokomis – *skaitomumas, aiškumas ar išraiškingumas*. Šios sąvokos svarbios konceptualiajame duomenų bazių projektavime. Atvirkštinei duomenų inžinerijai būdingi kiti kriterijai, kuriuos ir išskirsime.

5.1 Reinžinerijos tikslai ir siekiai

- ***Teisingumas***: schema yra *sintaksiškai teisinga*, jei visi schemos konceptai yra tinkamai apibrėžti. Schema yra *semantiškai teisinga*, jei visi jos konceptai vartojami pagal jų apibrėžimą, pavyzdžiui, asociacija nėra modeliuojama kaip apibendrinimas ar atvirkščiai.

- ***Primityvumas***: schema yra *primityvi*, jei visos schemos struktūros priskirtos ne daugiau kaip vienai realaus pasaulio abstrakcijai.

- ***Minimalumas***: schema yra *minimali*, jei negalima pašalinti nė vieno schemos elemento, nepraradus informacijos. Schemos, kurios nėra minimalios, dažnai yra daug sunkiau suprantamos.
- ***Tinkamumas*****:** schema yra *tinkama* (angl. *relevant*), jei visi jos objektai ir ryšiai yra aiškiai ir tiksliai atvaizduoti duomenų bazės schemoje, naudojant atitinkamo modelio konceptus.
- ***Normiškumas***: duomenų bazės schema turi tenkinti tam tikrą norminę formą.

Svarbiausias kokybės kriterijus yra semantinis teisingumas. Minimalumo ir primityvumo kriterijai yra glaudžiai susiję su semantinės abstrakcijos lygiu, kadangi pertekliškumas ir paslėpti objektai mažina abstrakcijos lygį. Tinkamumo kriterijus svarbus tuo, kad leidžia visus ryšius ir objektus tiksliai aprašyti modeliavimo konceptais, o ne paslėptais ryšiais ar loginiu suvokimu.

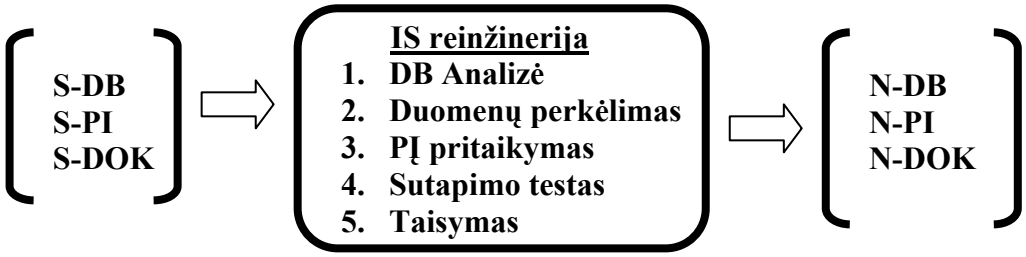
Be struktūros transformacijų, atvirkštinė inžinerija yra susijusi ir su egzempliorių bei taikomųjų programų pakeitimu. Dėl to reikia atsižvelgti į formalias transformacijos savybes. Ypač svarbus yra duomenų bazės turinys – galimų duomenų egzempliorių aibė, susieta su duomenų bazės schema. Idealiu atveju informacijos turinys transformuojamas nepakinta.

5.2 Reinžinerijos etapai

Remiantis aprašytomis metodikomis suformuosime etapus ir jų seką:

1. Išanalizuoti duotą duomenų bazės struktūrą.
2. Duomenų perkėlimas iš senosios duomenų bazės į naująją
3. Pritaikyti senąsias programas naudoti naują duomenų bazę
4. Sutapimo testas
5. Taisymas

Bendru atveju duomenų bazės reinžinerijos procesas atvaizduojamas, kaip parodyta 5.1 paveiksle.



5.1 pav. IS reinžinerijos schema

Čia **DB = < T,K,R,I,W,SP,F >** ,

kur $T=\{t_1,t_2,t_3,\ldots\}$ – duomenų bazės lentelių aibė;

$K=\{k_1,k_2,k_3,\ldots\}$ – duomenų bazės lentelių raktinių laukų aibė;

$R=\{r_1,r_2,r_3,...\}$ – duomenų bazės ryšių aibė;

$I=\{i_1,i_2,i_3,\ldots\}$ – duomenų bazės indeksų aibė;

$W=\{w_1,w_2,w_3,...\}$ – duomenų bazės vaizdų aibė;

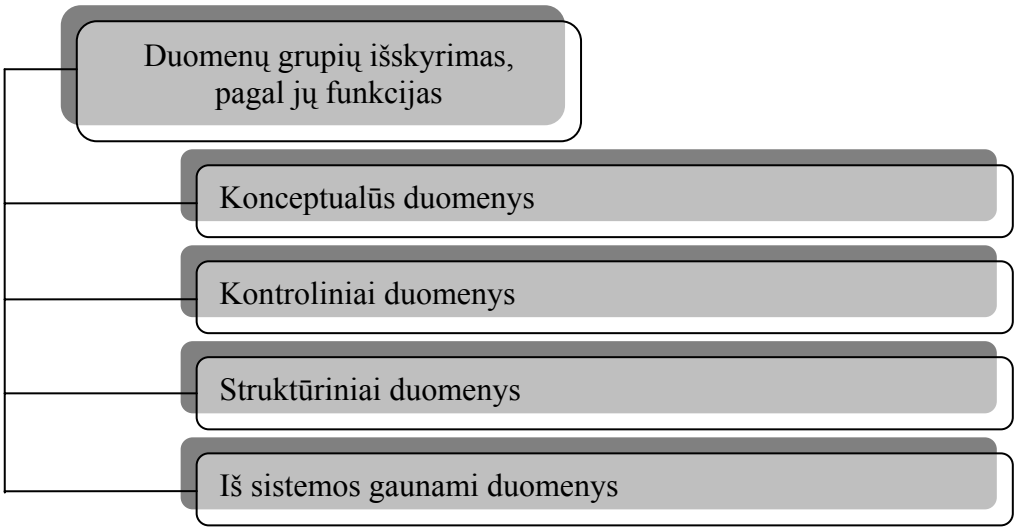
$SP=\{sp_1,sp_2,sp_3,\ldots\}$ – duomenų bazės procedūrų aibė;

$F=\{f_1,f_2,f_3,...\}$ – duomenų bazės funkcijų aibė;

S-DB - senoji duomenų bazė, N-DB – Naujpji duomenų bazė, gauta po reinžinerijos proceso.

5.2.1 Duomenų bazės analizė

Pirmiausia , kaip jau buvo minėta, reikia išanalizuoti esamus duomenis, jų struktūrą, tarpusavio ryšiai.



5.2 pav. Duomenų grupavimas

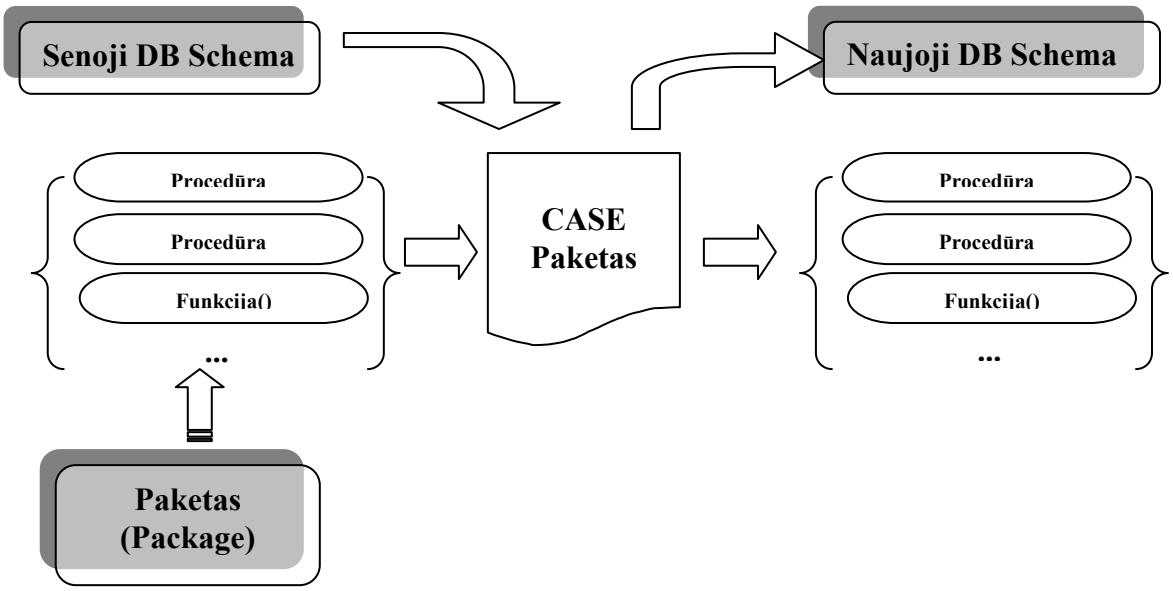
Pirmiausia iš eilės, kaip pavaizduota 5.2 pav., reikia išanalizuoti pačius duomenis susijusius su sistemos funkcionavimu. Kartais jie gali atrodyti paprasti sistemos vartotojui, tačiau sistemos reinžinerijos specialistui gali iškilti visokiu neaiškumų. Taigi pirmiausia reikia labai įsigilinti į dalykinę sritį.

Toliau iš eilės reikia įvertinti kontrolinius duomenis, tai yra duomenis, kurie įrašomi atsiradus realaus pasaulio veiksniams, tokie kaip sistemos vartotojų teisės, vartotojų atliktų operacijų fiksavimas (logs). Pirmiausia reikia paimti iš senosios sistemos visus kontrolinių duomenų fiksavimo principus ir numatyti jų papildymą naujojoje sistemoje taip, kad būtų gauti visi kontroliniai duomenys, kurių pageidauja naujos informacinės sistemos užsakovai. Dažniausiai tokiu atveju atsiranda daug poreikių, bet tai nėra blogai , nes analizuojant kontrolinius duomenis galima gauti daug vertingos informacijos, tokios kaip pavyzdžiui: kada vartotojas buvo prisijungęs prie duomenų bazės ar kada buvo ištrintas vienas ar kitas įrašas.Tada dar lieka struktūriniai duomenys ir apskaičiuoti duomenys. Struktūriniai duomenys tai duomenys skirti organizuoti ir palaikyti duomenų struktūras, tokie kaip sekos, trigeriai ir t.t. O apskaičiuojami duomenys, tai programoje skaičiuojami ir generuojami duomenys.

Bendrai duomenų analizės etapo metu gali būti aptikti sinoniminiai duomenys. Tokiu atveju jie turėtų būti pervadinti ir išsaugoti arba jeigu tai besidubliuojantys duomenys, šie duomenys turėtų būti pašalinti.

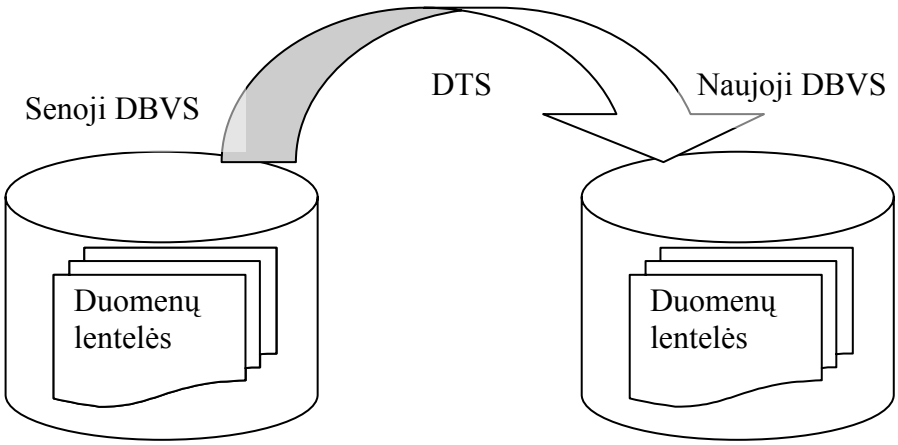
5.2.2 Duomenų perkėlimas

Išanalizavus duomenis juos perkeliame etapais. Pirmiausia turėtumėme kelti informacinėje sistemoje tiesiogiai naudojamus duomenis ir tik po to sisteminius bei struktūrinius duomenis. Duomenų bazės schemos perkėlimui gali būti naudojami įvairūs duomenų projektavimo paketai. Vienas iš populiariausių ko gero butų „Visio“ paketas. Šio paketo naudojimas leidžia perkelti ne tik Duomenų bazės schemas, bet ir stored procedūras ar funkcijas. Tačiau čia kartais susiduriame su nesuderinamumais. Pavyzdžiui Oracle DBVS yra tokie objektai, vadinami „Package“, o Microsoft SQL DBVS gali būti kuriamos tik Saugomos procedūros ir Funkcijos. Oracle paketo (angl. Package) viduje gali būti aprašytos ir realizuotis saugomos procedūros ir funkcijos. Tokių objektų perkėlimo automatizuoti nepavyks, todėl reikia imtis priemonių, kurios tai įgalintų padaryti su mažiausiomis resursų sąnaudomis. Šiuo atveju, ko gero tektų transformuoti Oracle paketus į atskiras saugomas procedūras ir funkcijas, kaip parodyta 5.3 paveiksle, ir tada jas perkėlinėti su tam turimais įrankiais.



5.3 pav. Duomenų perkėlimas Visio paketo pagalba

Patys duomenys gali būti perkeliami tokiu patogiu Data Transformation Service (DTS) įrankiu, kuris įeina į Microsoft SQL Server 2000 Enterprise Manager paketą, kaip parodyta 5.4 paveiksle.



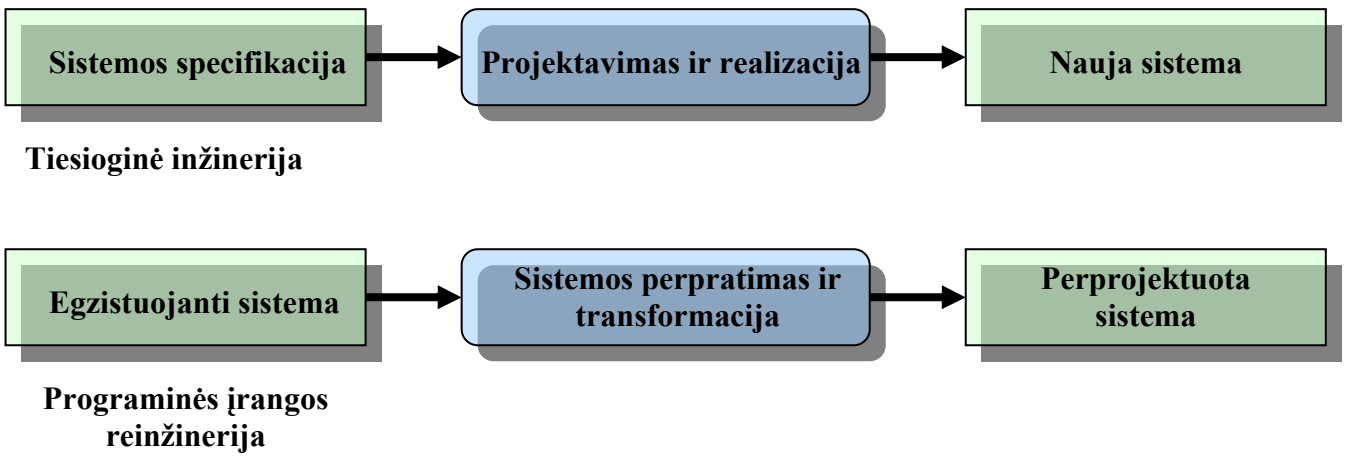
5.4 pav. Duomenų perkėlimas DTS paketo pagalba

Visi likę perkėlimo darbai negali būti atliekami automatiškai, todėl tiek ryšiai tarp naujosios duomenų bazės lentelių, tiek vartotojų teisės turėtų būti perkeltos rankomis.

5.2.3 Programinės įrangos pritaikymas

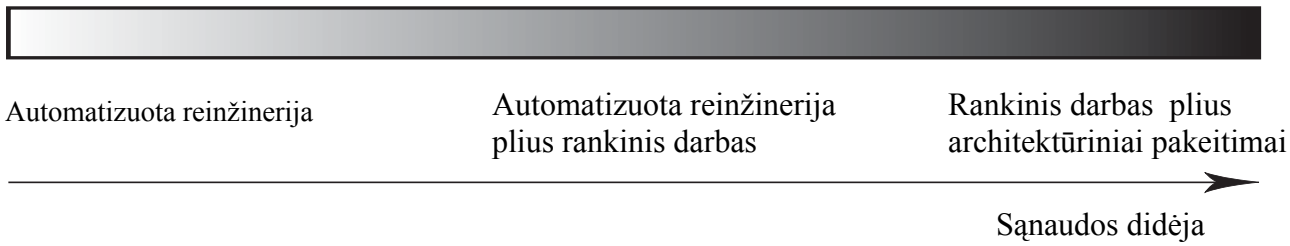
Kiekvienai senajai programai, kuri naudos naujos duomenų bazės duomenis, taip pat turi būti atlikta reinžinerija. Tai reikia atlikti tam, kad ji duomenis sėkmingai paimtų, įrašytų ir, jeigu reikia, atliktų kitus veiksmus su jais. Tai reiškia, kad kiekviena programa individualiai turi būti peržiūrėta ir pakeistos jos dalys, kuriose nurodytos jungimosi instrukcijos prie naujosios duomenų bazės, arba perprojektuota visa programa bendrai. Tuo pačiu metu, žinoma galima įvesti patobulinimus programinėje įrangoje, kurie reikalingi, kad ji atitiktu naujas verslo taisykles ar tiesiog tam tikslui, kad programinę įrangą būtų lengviau palaikyti. Žinoma jei įmanoma, pirmiausia reikia naudoti automatinį programinės įrangos perprojektavimo būdą, ir tik po to, jei kuriems nors komponentams jis negali būti pritaikytas, tada reikia taikyti rankinį programų ar atskirų paketų perprogramavimą.

Pagrindiniai privalumai naudojant senų programų reinžineriją lyginant su naujai programų kūrimu yra savaime suprantami: pirmiausia senos programinės įrangos reinžinerijos procesas yra mažiau imlus laikui ir pinigams palyginus su naujai daromu aplikacijų programavimu, antra − projektuojant naują programinę įrangą visada atsiranda klaidų tikimybės. Taip pat tuo atveju gali iškilti projektavimo, programavimo bei specifikacijos nesklandumų. Esminiai skirtumai tarp jau esamos programinės įrangos reinžinerijos ir naujai projektuojamos programinės įrangos pavaizduoti 5.5 paveiksle.



5.5 pav. Tiesioginė inžinerija ir reinžinerija

Vykdant programų reinžineriją svarbiausia naudoti kuo mažiau rankinio darbo. Kuo daugiau darbo turėtų būti atliekama naudojant automatizavimo įrankius, tokius kaip CASE priemones (Visio, Rational Rose). Žemiau pateiktame paveikslėlyje, kaip tik ir vaizduojama tiesiogine laiko ir pinigų sąnaudų priklausomybė nuo rankinio darbo kiekio.



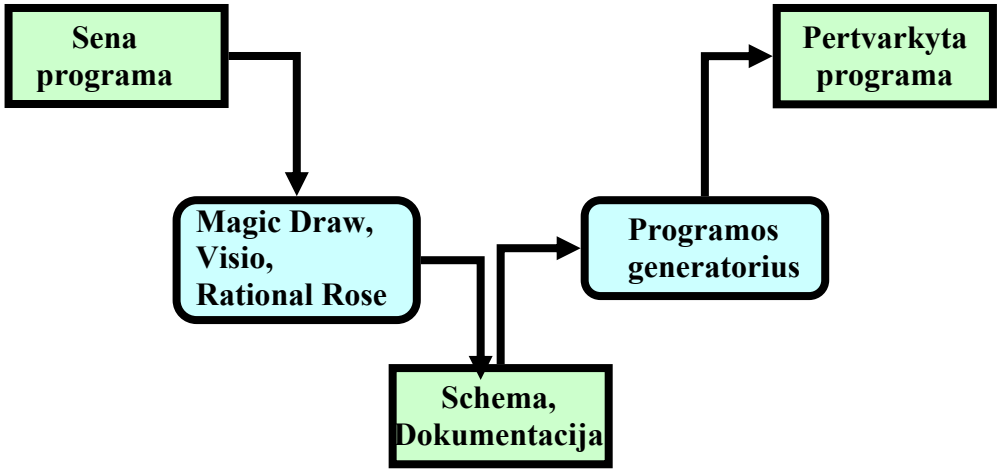
5.6 pav. Darbo ir laiko sąnaudos

Taip pat galimas atvejis, kad programinės įrangos reinžinerijos metu gali reikti pakeisti netgi iš vienos programavimo kalbos į kitą. Taip dažniausiai atsitinka tokiais atvejais:

- ✓ Kai pasikeičia aparatūrinė dalis
- ✓ Kai specialistų įgūdžiai per menki atnaujinti senąją programinę įrangą
- ✓ Kai pasikeičia organizacijos ar įmonės politika

Bet kokiu atveju toks reinžinerijos modelis naudojamas itin retai.

Automatinio programinės įrangos reinžinerijos proceso schema atrodys kaip pavaizduota senkančiame paveiksle.



5.7 pav. Automatinis programos reinžinerijos procesas

Tačiau taikant automatinius programinės įrangos metodus atsiranda ir didesnių ar mažesnių trūkumų, kuriuos reikia įvertinti:

- ✓ Komentarų, esančių programoje, praradimas.
- ✓ Dokumentacijos visiškas arba dalinis praradimas ir poreikis ją rašyti iš naujo.
- ✓ Skaičiavimo resursų poreikis
- ✓ Perprogramuotos programinės įrangos suprantamumas gali pablogėti

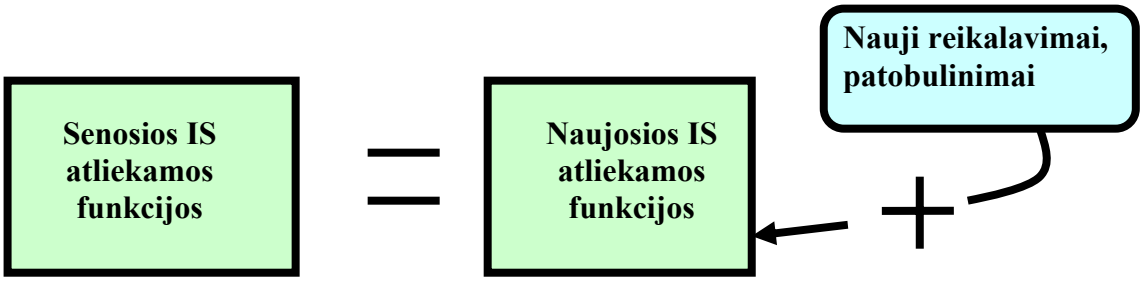
Primityviausias ir daugiausia resursų reikalaujantis programinės įrangos reinžinerijos metodas yra rankinis.

5.2.4 Sutapimo testas

Nuo šio etapo reinžinerijos procesas beveik baigtas, liko tik analizė ir klaidų paieška. Klaidų paieška išskiriama į dvi dalis pagal tiriamąjį objektą:

- ✓ Sutapimo testas atliekamas IS funkcijų lygyje
- ✓ Sutapimo testas atliekamas duomenų sutapimo lygyje

Pirmoji dalis – IS atliekamų funkcijų tikrinimas. Čia reikia įsitikinti, kad naujoji IS atlieka ir teisingai atlieka visas senosios IS funkcijas, bei naujas funkcijas, kurių poreikis iškilo reinžinerijos proceso metu. Schematiškai tai pavaizduota 5.8 paveiksle.



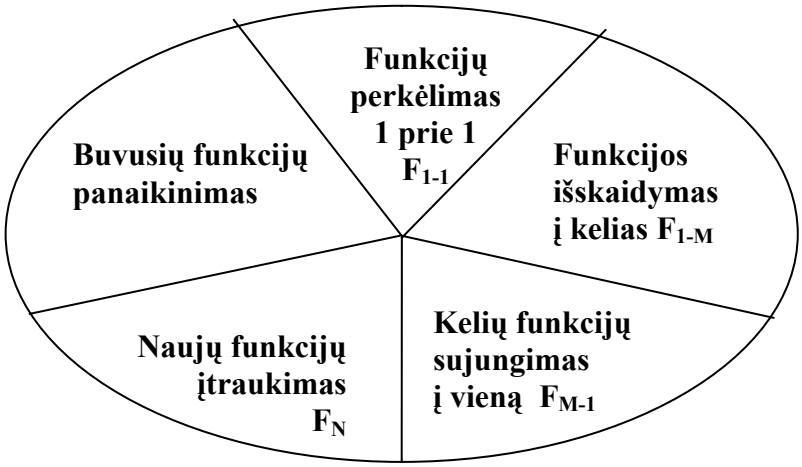
5.8 pav. Naujosios ir senosios IS atliekamų funkcijų sulyginimas

Taigi tikrinama ne tik ar naujoji IS atlieka visas tas pačias f-jas, kaip ir senoji IS, bet ji turi grąžinti tokius pat rezultatus. Realiame pasaulyje reinžinerijos metu gali būti ne tik perkeliamos funkcijos iš naujosios sistemos į senąją, bet senosios apjungiamos arba išskaidomos bei įdedamos naujosios.

$$F=F_{1\text{-}1} \vee F_{1\text{-}M} \vee F_{M\text{-}1} \vee F_N \qquad (1)$$

čia: $F_{1\text{-}1}$ − funkcijos perkeltos vienas prie vieno; $F_{1\text{-}M}$ − funkcijos, kurios perkeliant išskaidytos į keletą funkcijų; $F_{M\text{-}1}$ − funkcijos, kurios perkeliant apjungtos į vieną funkciją; $F_{1\text{-}1}$ − naujai sukurtos funkcijos;

Grafiškai tai atvaizduota 5.9 paveiksle.



5.9 pav. IS funkcijų perkėlimas

Funkcijų perkėlimo realizavimas aprašomas šiomis taisyklėmis naudojantis predikatų logiką:

1 taisyklė: Kiekviena senosios sistemos funkcija yra perkeliama į naująją sistemą ir perkeltoji funkcija atitinka senajai.

$$\forall f_i\,(\text{senos\_sistemos\_funkcija}(f_j) \rightarrow (\exists !\, f_j\,(\text{Naujos\_sistemos\_funkcija}(f_j)) \wedge (\text{Atitinka}(f_j,f_i))) \quad (2)$$

2 taisyklė: Senosios sistemos funkcija yra išskaidoma į kelias funkcijas ir jos perkeliamos į naująją sistemą.

$$\forall f_i\,(\text{senos\_sistemos\_funkcija}(f_j) \rightarrow (\exists\, f_j\,(\text{Naujos\_sistemos\_funkcija}(f_j)) \wedge (\text{Apima}(F_{M-1},f_i))) \quad (3)$$

3 taisyklė: Senosios sistemos kelios funkcijos apjungiamos ir perkeliamos į naująją sistemą kaip viena funkcija.

$$\forall f_i\,(\text{senos\_sistemos\_funkcija}(f_j) \rightarrow (\exists\, f_j\,(\text{Naujos\_sistemos\_funkcija}(f_j)) \wedge (\text{Įeina\_į\_sudėtį}(F_{1-M},f_i))) \quad (4)$$

Antroji dalis – Duomenų sutapimo testas . Jos metu turi būti patikrinta

- ✓ Lentelių, esančių duomenų bazėje, kiekių sutapimas
- ✓ Indeksų, trigerių ir kitų duomenų bazės elementų sutapimas
- ✓ Įrašų kiekiai kiekvienoje lentelėje
- ✓ Atsitiktinai apimtos ir sulygintos įrašų poros iš senosios ir naujosios DB

Visos klaidos turi būti dokumentuotos ir ištaisytos žemiau aprašytame redagavimo etape

5.2.5 Redagavimas

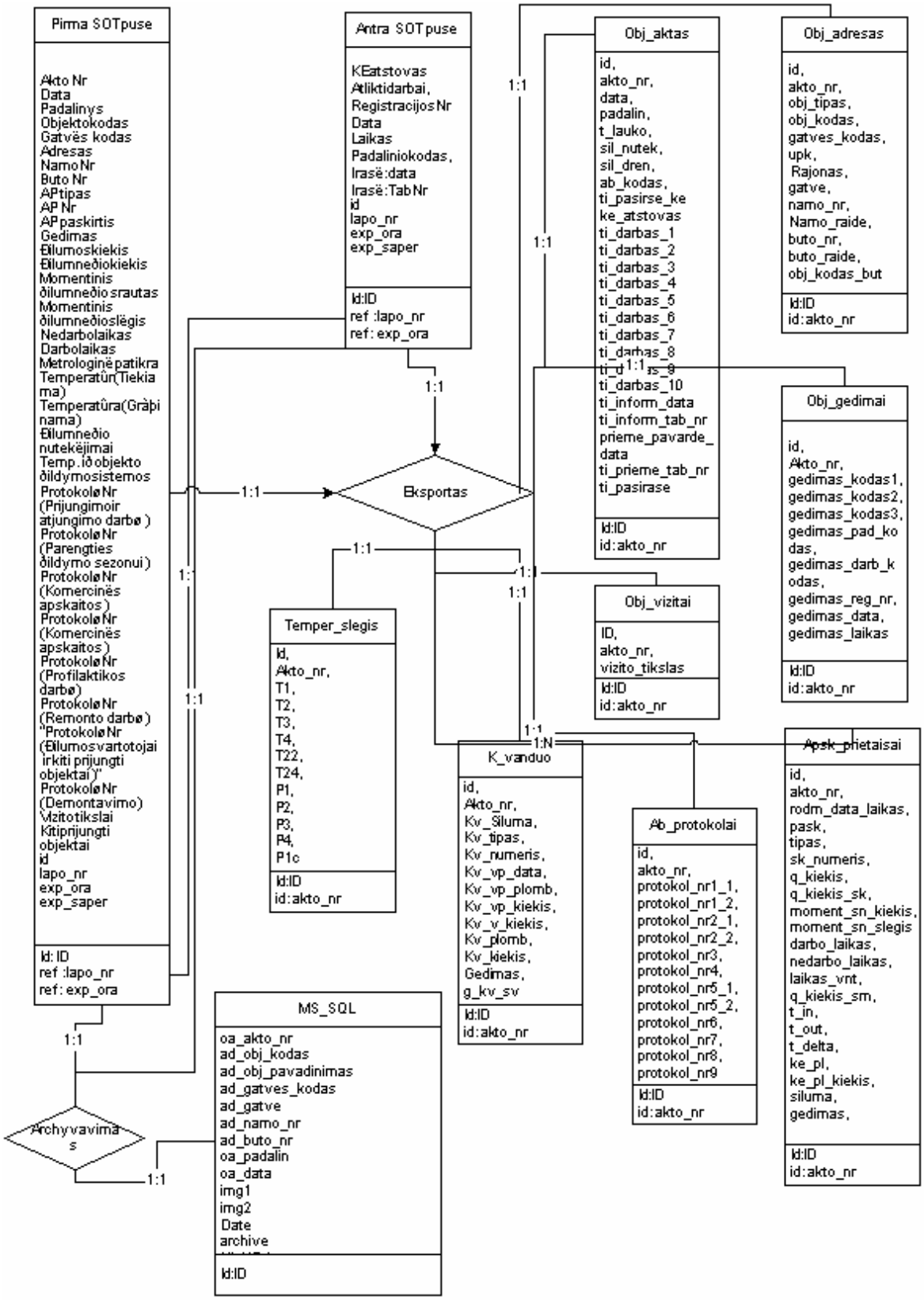
Kiekvieną atsiradusią klaidą, aptiktą testuojant reikia nuodugniai išanalizuoti ir naujoji sistema turi būti pataisyta, kad pašalinti tuos trūkumus. Po šio žingsnio vėl reiktų praleisti sutapimo testus ir tol kartoti tuos veiksmus, kol klaidų nebebus pastebėta. Šiuos veiksmus turi atlikti specialiai apmokyti darbuotojai, bet ne programuotojai.

6 DUOMENŲ BAZĖS REINŽINERIJOS METODO EKSPERIMENTINIS TYRIMAS

Šio eksperimento objektas yra Oracle DB ir MSSQL Duomenų bazių valdymo sistemos. Šiuo atveju atliekama informacinės sistemos Oracle duomenų bazės pagrindu reinžinerija. Oracle duomenų bazės pagrindu yra realizuota realiai veikianti informacinė sistema – „Šilumos objektų tikrinimo aktų skenavimas“. Visas ŠOT Akto apdorojimo seka pavaizduota žemiau esančiose schemose. Remiantis aukščiau aprašytais ir pasiūlytais etapais atliktas eksperimentinis tyrimas.

6.1 Duomenų bazės struktūros analizė

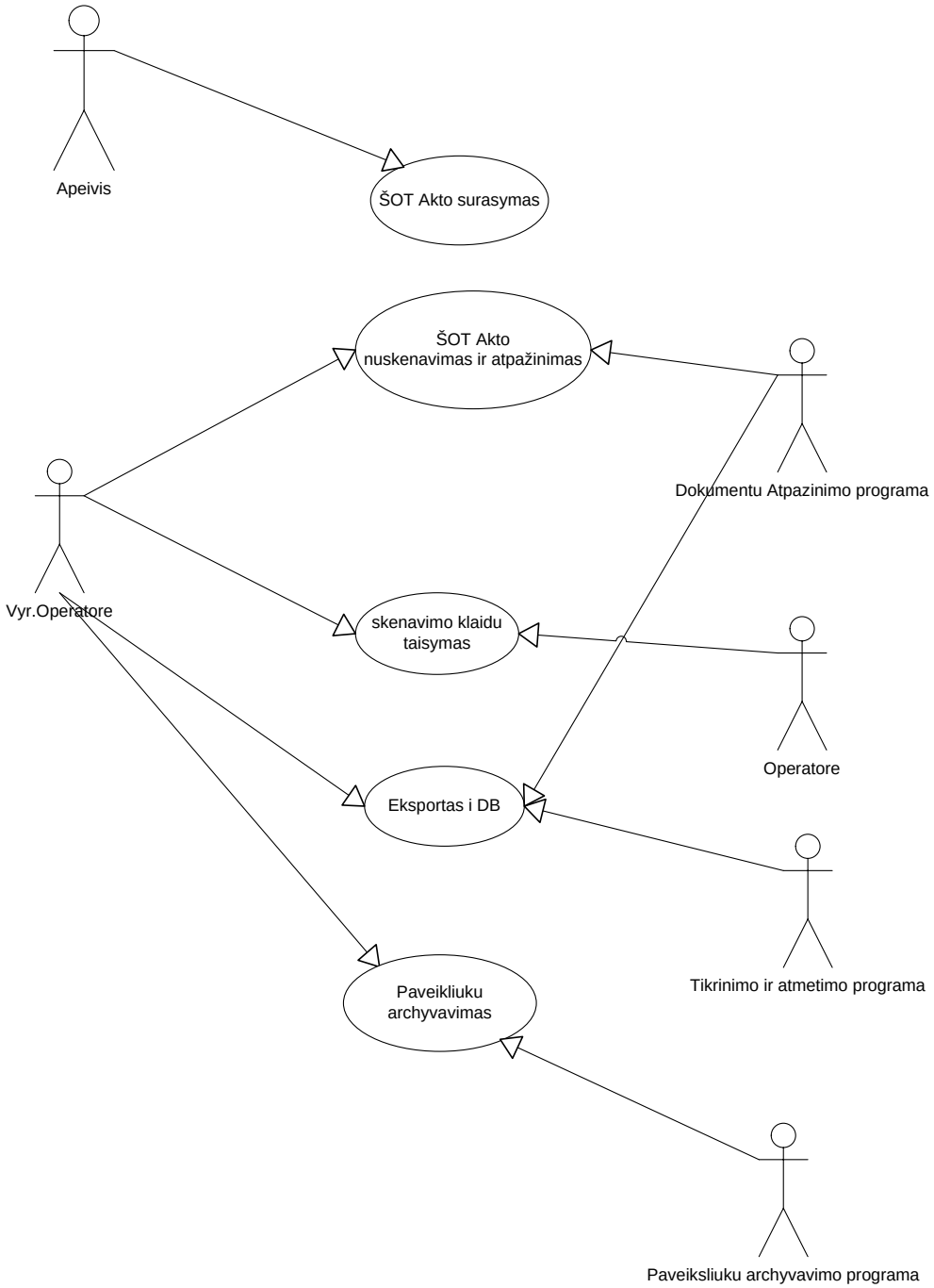
Pagal siūlomą koncepcinį duomenų bazės reinžinerijos modelį, pirmiausia atliekama duomenų struktūrų analizė. Žemiau pavaizduota realiai egzistuojančios DB schema. Visas procesas vyksta taip: pirmiausia darbuotojai, vadinami apeiviais, šilumos objektuose surašo ŠOT Aktą ir jis parnešamas į skaičiavimo centrą. Tada operatorės juos nuskenuoja, apdoroja spec. programa − Abby Form Reader ir duomenys iš ŠOT Akto perkeliami į Oracle DBVS. Ten atliekamas, vadinamas eksportas, t. y. atsiųsti duomenys dar kartą patikrinami, sulyginamas objekto kodas su duomenų bazėje esančiu kodu, Objekto adresas, objekte esantys skaitikliai. Jei kokie nors duomenys neatitinka tikrovės, ŠOT Aktas pažymimas kaip neteisingas, ir apie tai informuojama operatorė. Ji atitinkamai imasi veiksmų: arba randa ir ištaiso skenavimo klaidą arba ŠOT Aktas grąžinamas jį surašiusiam meistrui perrašyti ir po to iš naujo skenuojamas. Po sėkmingos akto skenavimo procedūros, toliau seka importuotų aktų „archyvavimas“, t. y. nuskenuotų aktų paveiksliukai yra perkeliami į WEB serverį, kartu įrašant į to serverio duomenų bazę pagrindinius duomenis reikalingus to paveiksliuko paieškai. Tai yra objekto kodas, objekto adresas akto surašymo data, surašiusio asmens padalinys.



6.1 pav. EER diagrama

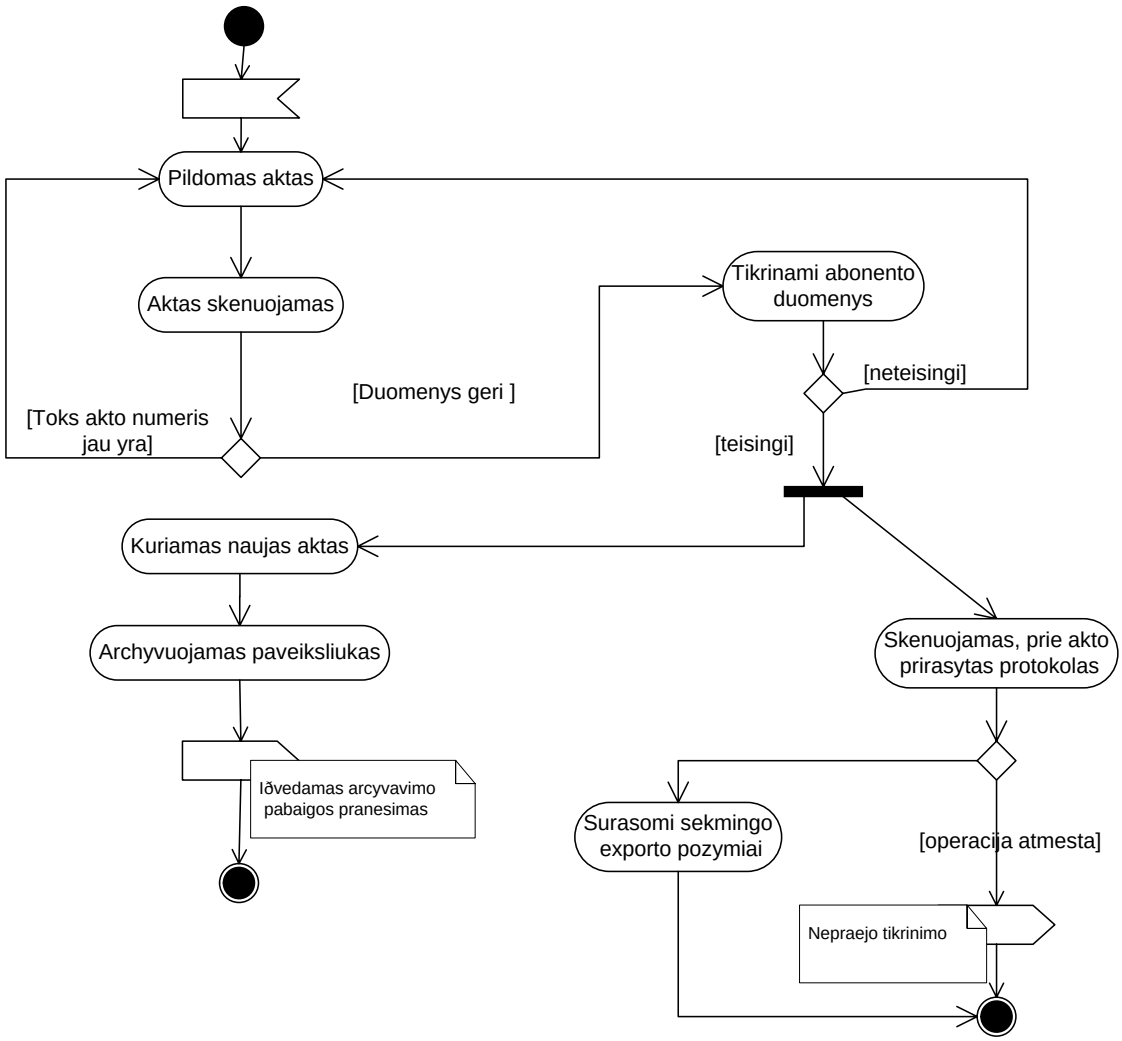
Šioje diagramoje pavaizduota reliacinės duomenų bazės EER diagrama su generuota iš esamos duomenų bazės. Eksportą realizuoja paketas ir jo rezultate duomens iš pirmos

ir antros pusės lentelių sukilnojami į visas kitas lenteles. Šios operacijos metu patikrinamas duomenų korektiškumas. Funkcija archyvavimas šiuo atveju atlieka programa parašyta Visual Basic kalba. Jos rezultatas – dar karta perrašyti pagrindiniai akto duomenys, bet jau į kitą duomenų bazę ir išsaugotas to akto nuskenuotas paveiksliukas. Jo vieta serveryje yra nurodoma toje pačioje MS_SQL duomenų bazėje viename iš laukų.



6.2 pav. Panaudojimo atvejų diagrama

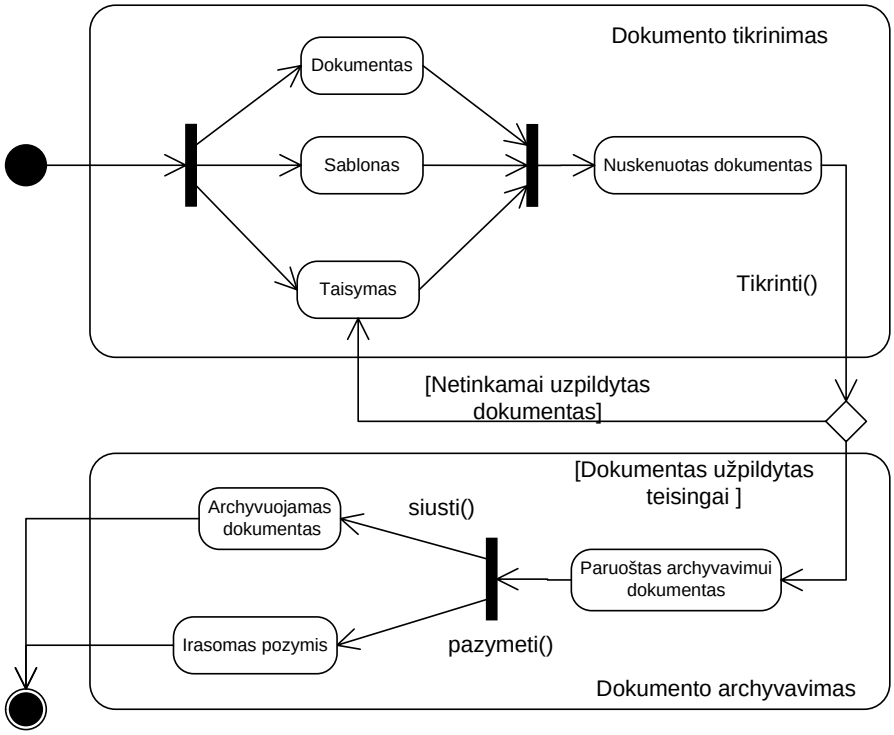
6.2 paveikslėlyje pavaizduotoje panaudojimo atvejų diagramoje pavaizduota, kad pirmiausia šilumos tinklų darbuotojas surašo šilumos objekto tikrinimo aktą. Po to vyr. operatorė jį nuskenuoja ir atpažįsta su dokumentų atpažinimo programa, operatorės atpažintus duomenis ištaiso, jei jie pasitaiko skenavimo klaidų ir nuskenuoti duomenys neatitinka surašytų akte duomenų. Tada vyr. operatore sueksportuoja nuskenuoto dokumento duomenis į duomenų bazę ir suarchyvuoja dokumento vaizdą į paveiksliukų saugyklą.



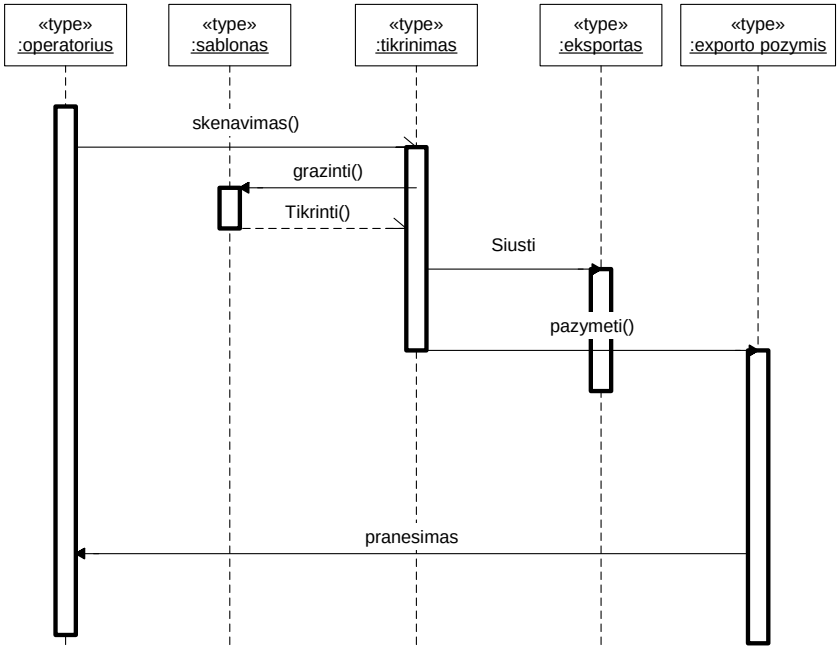
6.3 pav. Veiklos diagrama. Akto apdorojimo scenarijus

Šioje akų apdorojimo scenarijaus diagramoje (6.3 pav.) matome , kokie veiksmai atliekami nuo pat pradžių aktą užkildžius iki jo suarchyvavimo.

6.4 paveikslėlyje pavaizduota Būsenų diagrama apdorojant aktą. Čia matome kaip apdorojamas aktas keičia savo būsenas iš nuskenuoto dokumento į sueksportuotą dokumentą ir po to į suarchyvuotą dokumentą



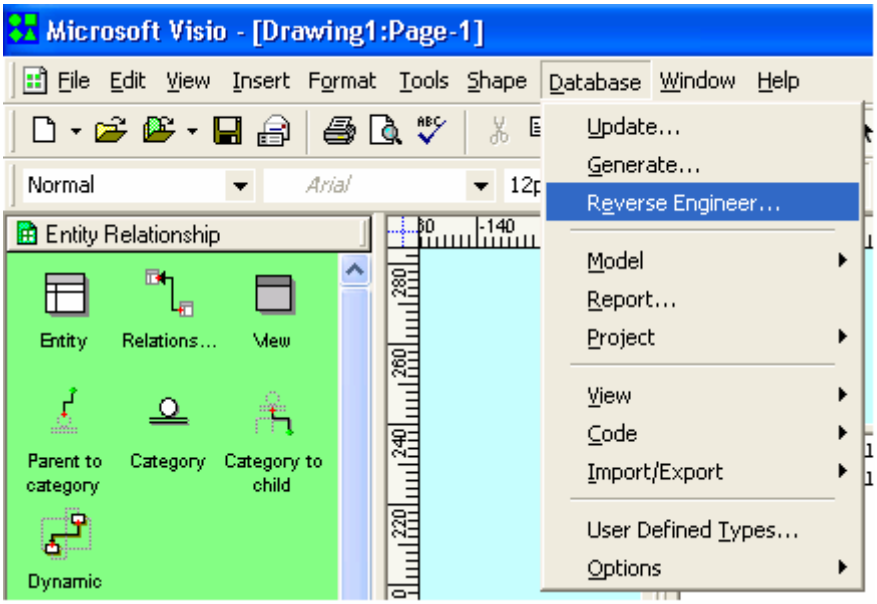
6.4 pav. Būsenų kaita apdorojant aktą



6.5 pav. Sekų diagrama

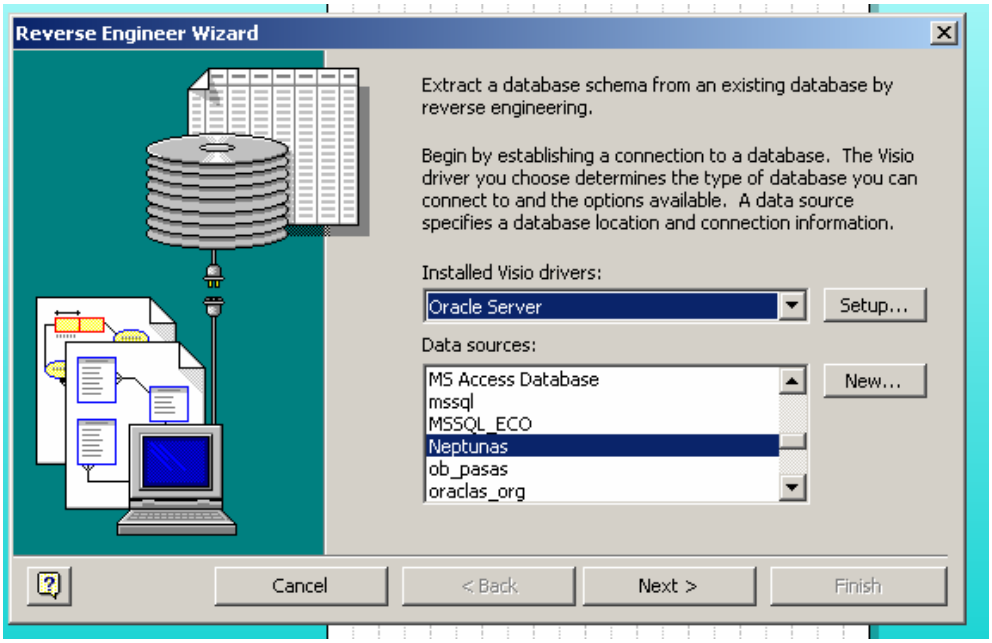
6.2 Duomenų perkėlimas

Duomenų bazės schema buvo perkelta Visio 2000 Enterprise Edition paketo pagalba sugeneruojant schemą ir ją su visais elementais (trigeriai, vaizdai, indeksai, saugomos procedūros, funkcijos) perkeliant į SQL Duomenų bazę. Pats perkėlimo procesas pavaizduotas sekančiuose paveikslėliuose.



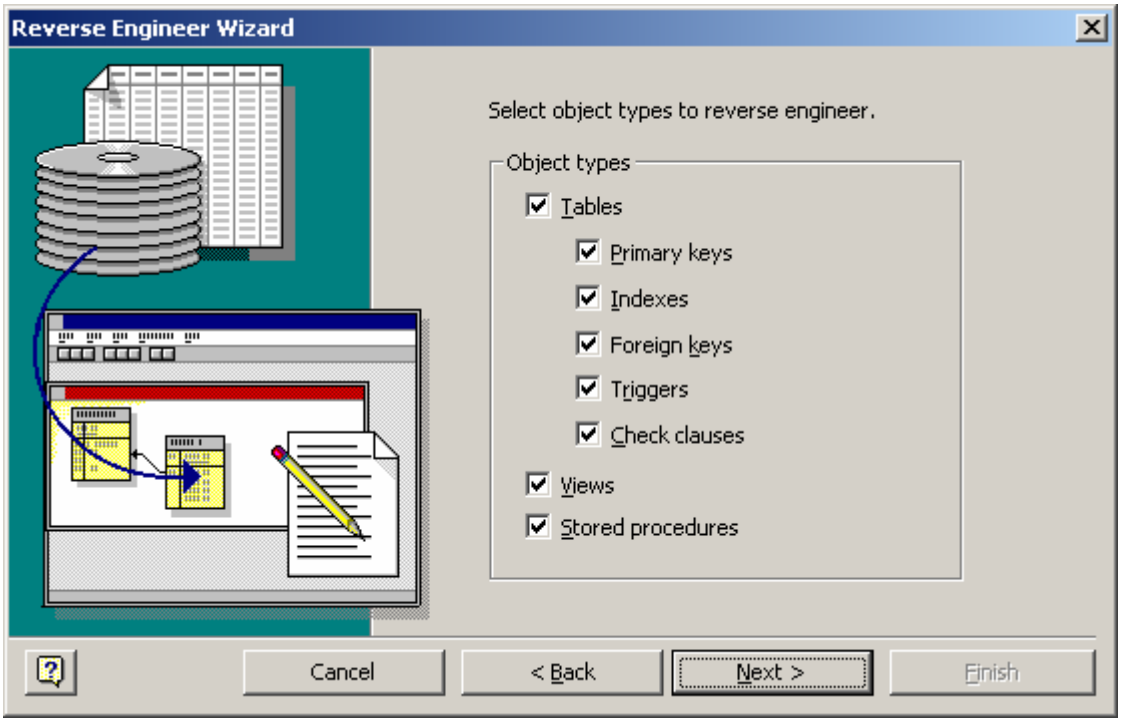
6.6 pav. Atgalinės reinžinerijos panaudojimas Visio pakete

Pirmiausia reikia pasirinkti iš kokios DBVS reikia generuoti BD schema



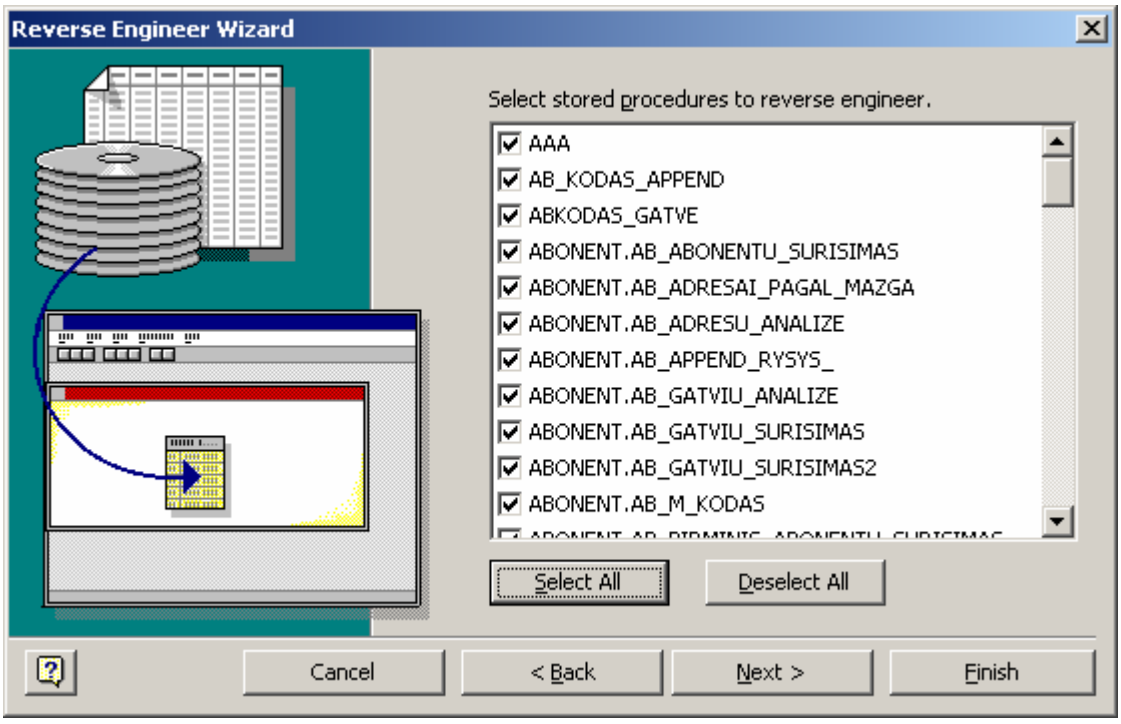
6.7 pav. Pradinės DB pasirinkimas Visio pakete.

Pasirinkus pradinę DB nurodome kuriuos komponentus eksportuosime į Visio modelį



6.8 pav. Eksportuojamų objektų pasirinkimas

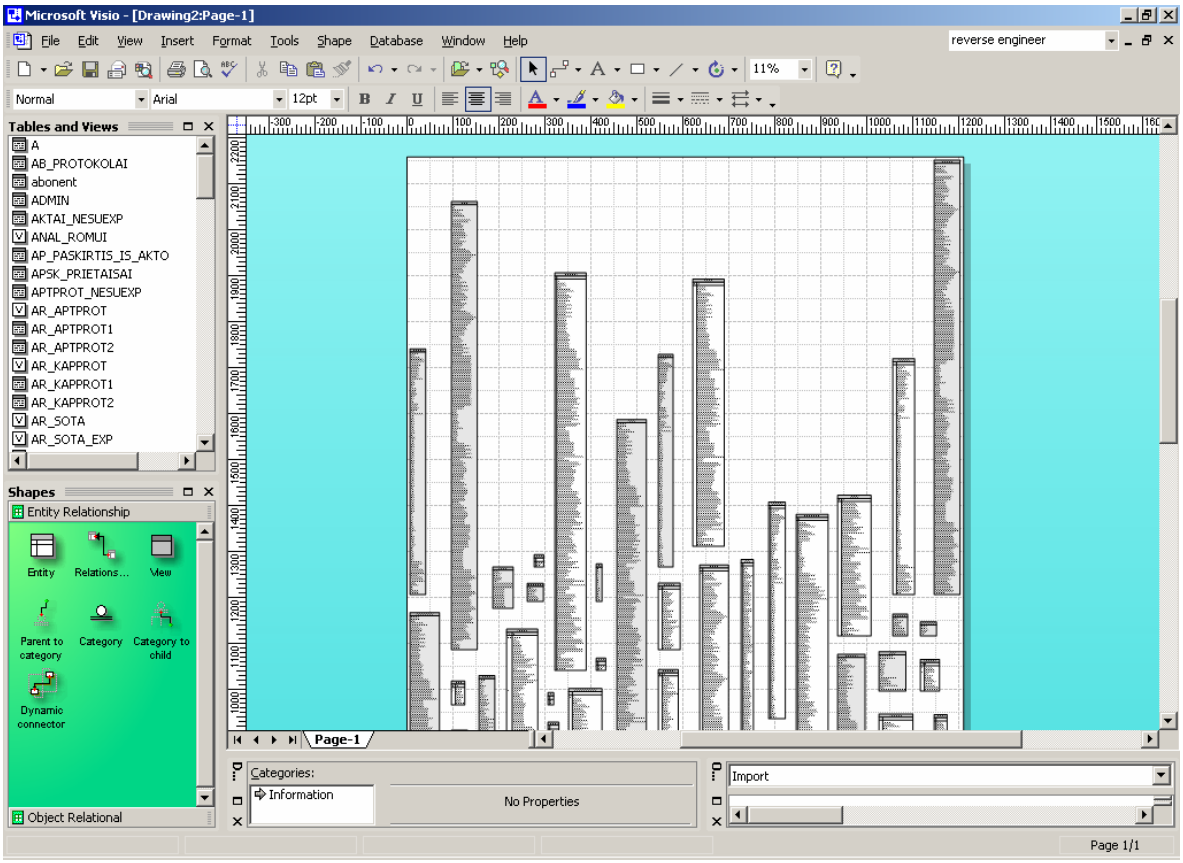
Pasirenkant punktą „Stored Procedures“ kartu yra parenkamos serveryje esančios funkcijos. O toliau pasirenkami konkretūs objektai.



6.9 pav. Eksportuojamų procedūrų pasirinkimo langas Visio pakete

Perkeliant duomenų bazės komponentus buvo susidurta su sunkumais kaip perkelti paketus iš Oracle DBVS į MS SQL DBVS, nes MS SQL struktūrose iš viso tokio objekto nėra. Todėl buvo rastas sprendimas − išskaidyti Oracle paketus į atskiras procedūras ir funkcijas.

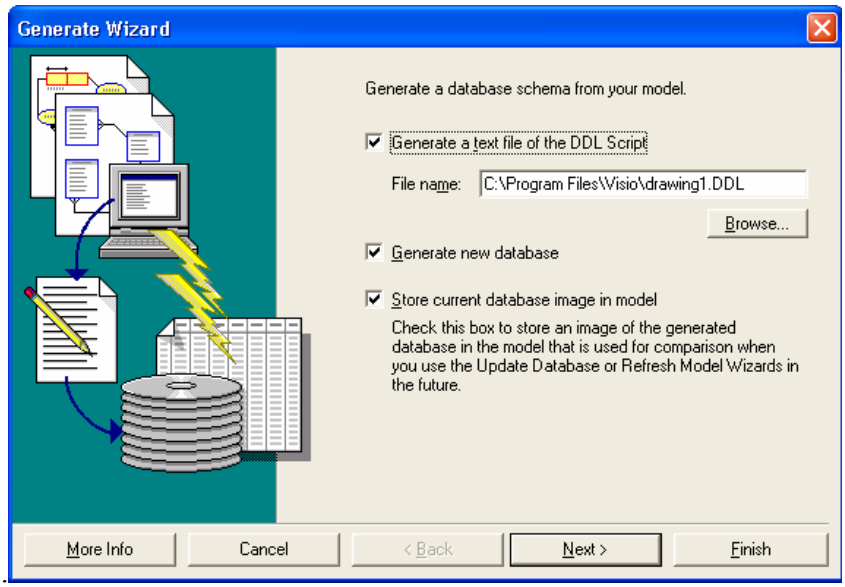
Analogiškas langas yra atidaromas norimų lentelių eksportui į Visio generuojamą modelį. Pasirinkus visus norimus objektus ir paspaudus „Finish“ mygtuką gaunamas užduotos duomenų bazės modelis, kurio pavyzdys pavaizduotas 6.10 paveiksle.



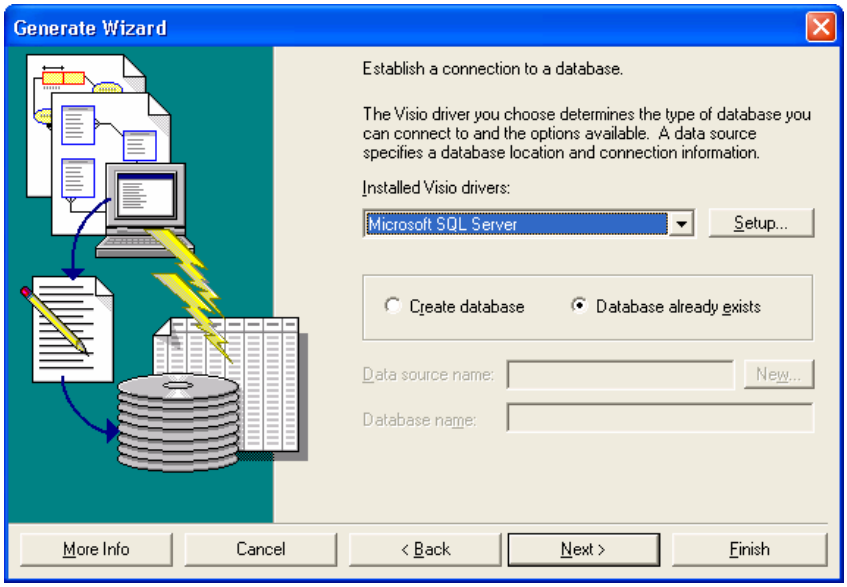
6.10 pav. Gautas DB modelis Visio pakete

Toliau jį reikia sueksportuoti į norimą gauti DBVS. Būtent ši funkcija veikia tik Visio 2000 Enterprise Edition. Buvo išanalizuotos funkcijos, kurias atlieka paketas Visio 2002 Professional, tačiau būtent ta versija nepalaiko gauto DB modelio sueksportavimo i norimą DB.

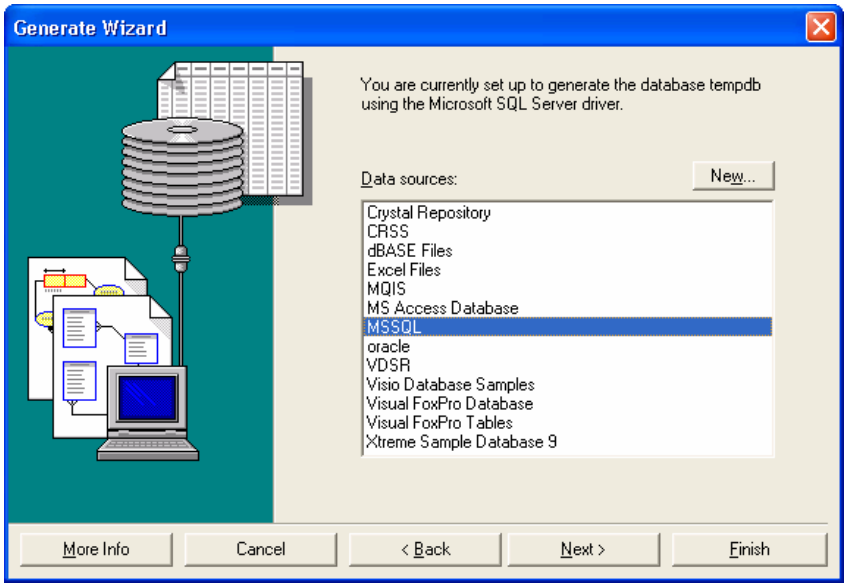
Taigi gavus modelį reikia atlikti tiesioginę inžineriją tai yra - pasirinkus DBVS, į kurią reikia perkelti visą DB, sugeneruotą modelį perkelti. Tai pavaizduota sekančiuose paveiksliukuose.



6.11 pav. Galutinės vietos pasirinkimas

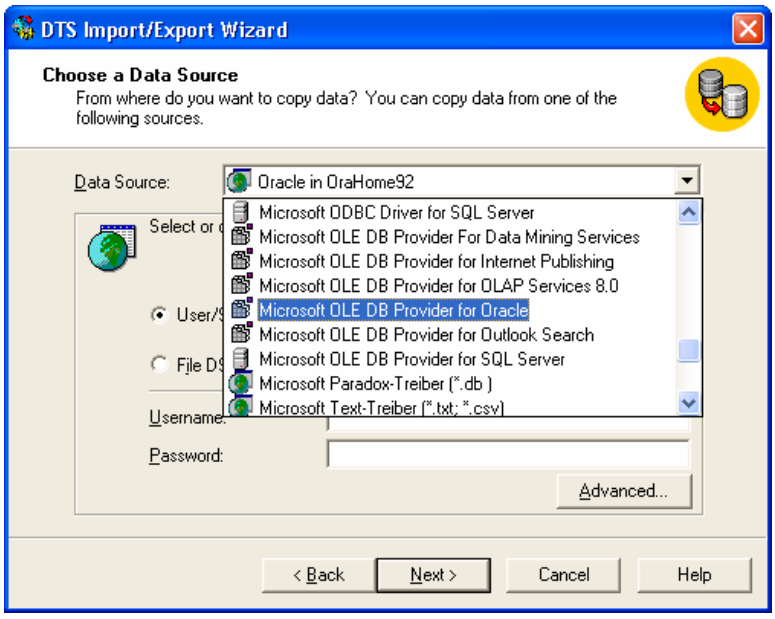


6.12 pav. DBVS tipo pasirinkimas

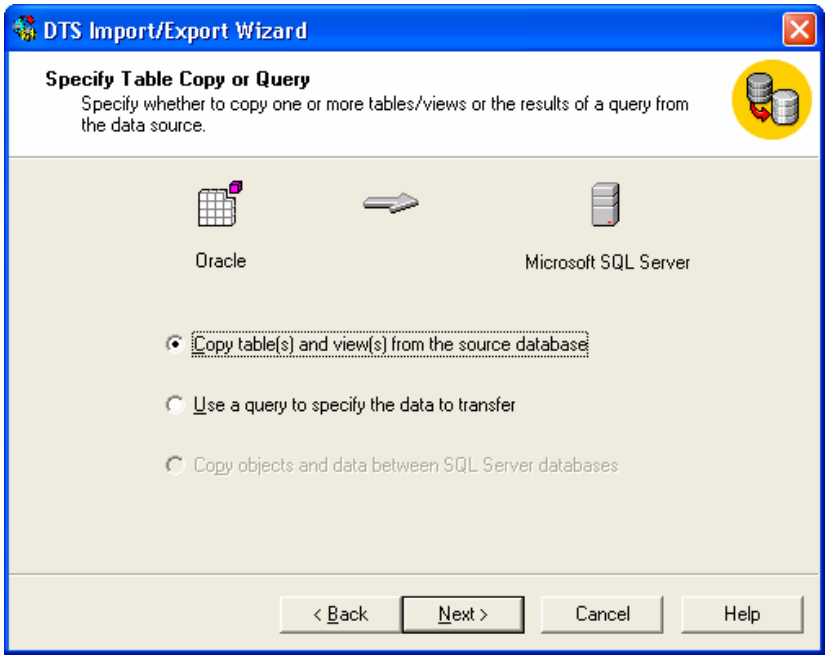


6.13 pav. DBVS tvarkyklės pasirinkimas

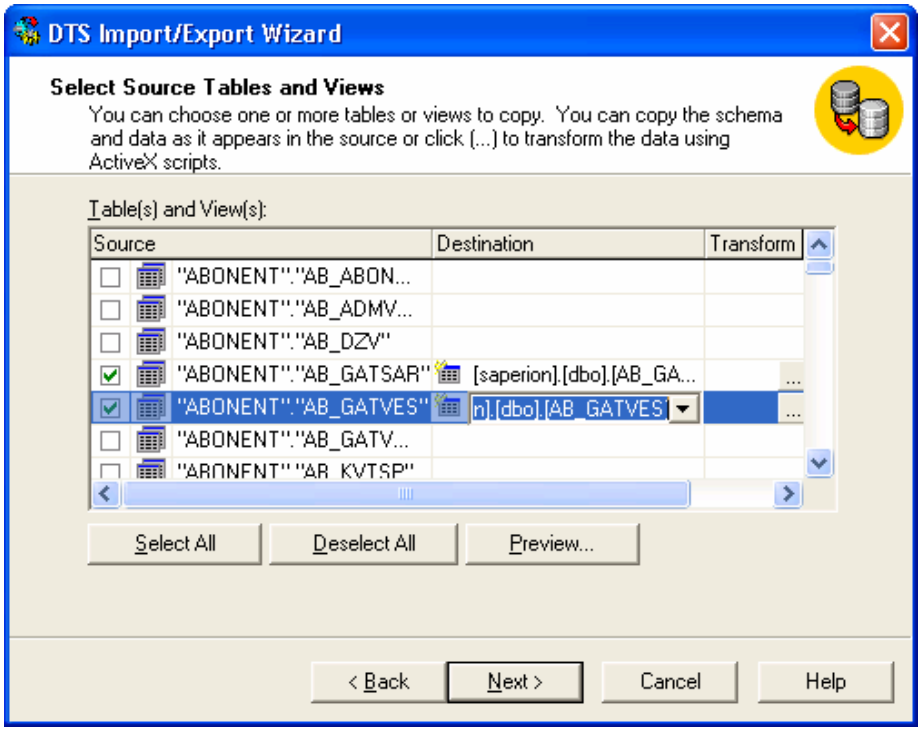
Taip pat naudojamas MS SQL Enterprise manager DTS funkcija lentelių duomenims perkopijuoti. Tai atliekama tiesiog pasirinkus MS SQL serverio Import-Export vedlį. Kaip parodyta žemiau esančiuose paveiksliukuose, tiesiog pasirenkame DB, iš kurios kopijuosime duomenis, tada analogiškai pasirenkame DB, į kurią kopijuosime duomenis, tada nurodome kokių lentelių duomenis kopijuosime ir į kurias lenteles tuos duomenis surašysim, prieš tai dar, jei reikia, pakoregavus lentelių laukų formatus.



6.14 pav. DBVS pasirinkimas iš kurios (į kurią) bus kopijuojami duomenys



6.15 pav. DBVS parinktos



6.16 pav. Kopijuojamų lentelių parinkimas

6.3 Senųjų informacinių sistemų pritaikymas prie naujosios DB

Senosios informacinės sistemos realizuotos įvairiomis kalbomis, tokiomis, kaip Visual Basic, Borland C++, PHP bei ASP. Taip pat yra programų paketas Abby Form Reader, kuris nusiunčia nuskenuotus duomenis iškart į nurodytos duomenų bazės atitinkamas lenteles. Primityviausias ir daugiausia resursų reikalaujantis programinės įrangos reinžinerijos metodas yra rankinis. Kadangi objektinis programavimas minėtose programose nenaudojamas, tai visos šios aplikacijos perdarytos rankiniu būdu pakeičiant prisijungimo prie DB instrukcijas, bei kai kuriuos parametrus, tokius kaip lentelių pavadinimus ar tipus. Žemiau pateikti pavyzdžiai, kaip jungiamasi prie vienos ar kitos duomenų bazės programose.

Prisijungimas prie Oracle serverio su php programavimo kalba

```
$conn = ora_logon("username@neptunas", "passw");
$curs = ora_open($conn);
ora_commitoff($conn);
$query = "select * from automat.dbf where kodas like '$kodas"._."'
and paskirtis='$paskirtis'";
ora_parse($curs,$query );
ora_exec($curs);
ora_fetch_into($curs,$row);
```

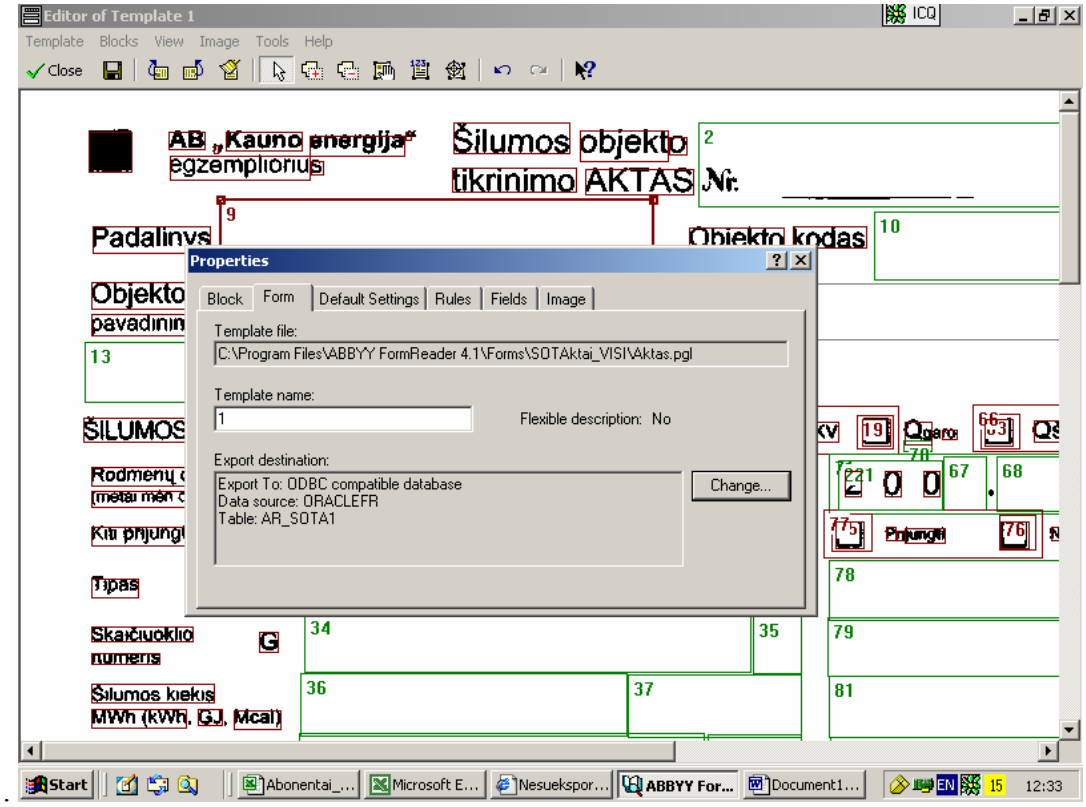
Prisijungimas prie Microsoft SQL Server 2000 su php programavimo kalba

```
MSSQL_CONNECT($hostname,$username,$password) or DIE("DATABASE FAILED
TO RESPOND.");
mssql_select_db($dbName) or DIE("Nerasta lentele");
$query = "select meter.meterid, complex.complexnumber, meter.name
from complex
inner join meter on meter.complexid=complex.complexid";
$result = MSSQL_QUERY($query);
$number = MSSQL_NUM_ROWS($result);
$name =  mssql fetch row ($result);
```

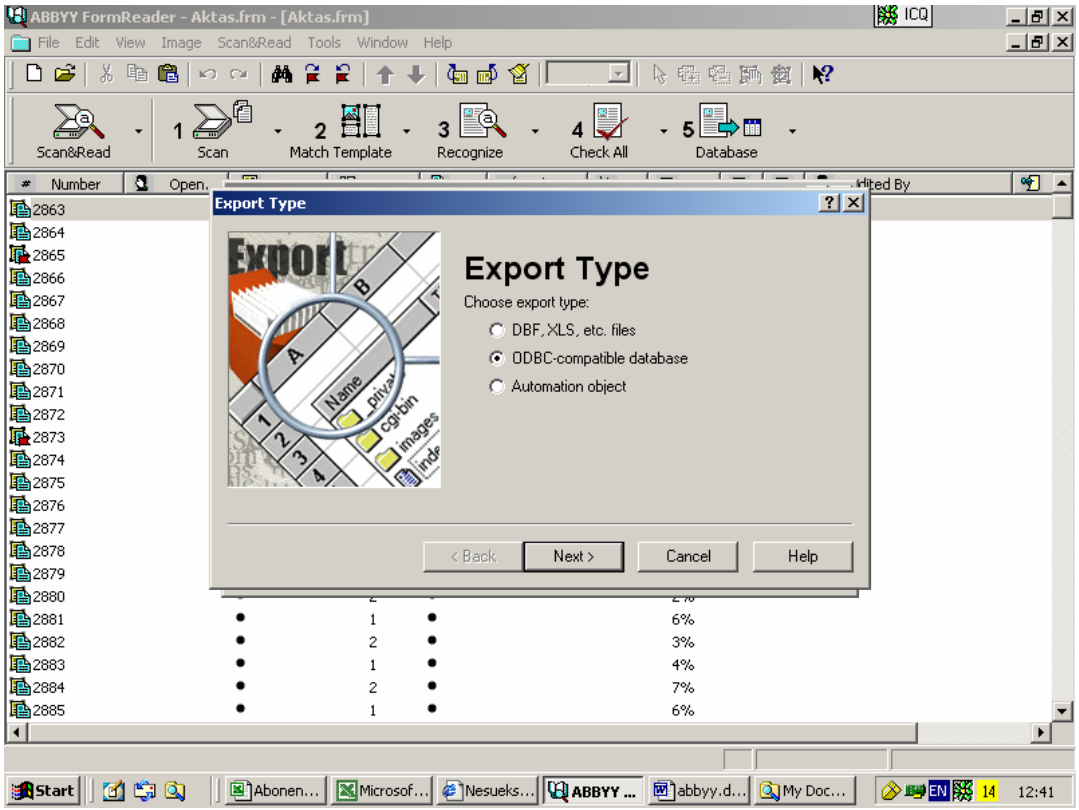
5.17 pav. Programos kodo ištraukos, jungiantis prie Oracle ir SQL Server 2000 DBVS

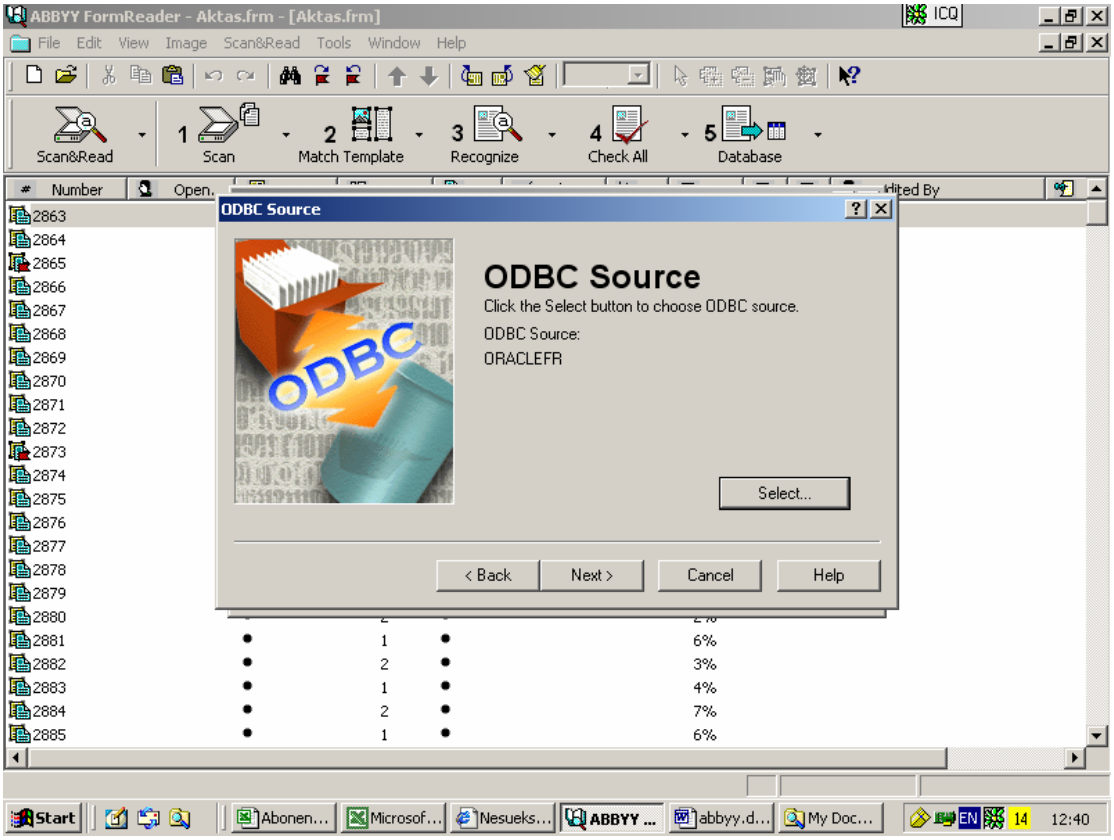
Kaip galima pastebėti iš pateiktų pavyzdžių, skiriasi tiktai prisijungimo eilutė ir joje paduodamų parametrų kiekis bei eiliškumas, visa kita beveik identiška. Kintamajam `$query` priskiriamas užklausos tekstas, kuri perduodama serveriui. Tiek vienu, tiek kitu atveju, kaip rezultatas yra grąžinamas objektas - kursorius. Norint gauti rezultatą iš kursoriaus ištraukiama (`fetch`) po vieną užklausos grąžintą įrašą, o toliau su juo gali būti atliekamos reikalingos operacijos.

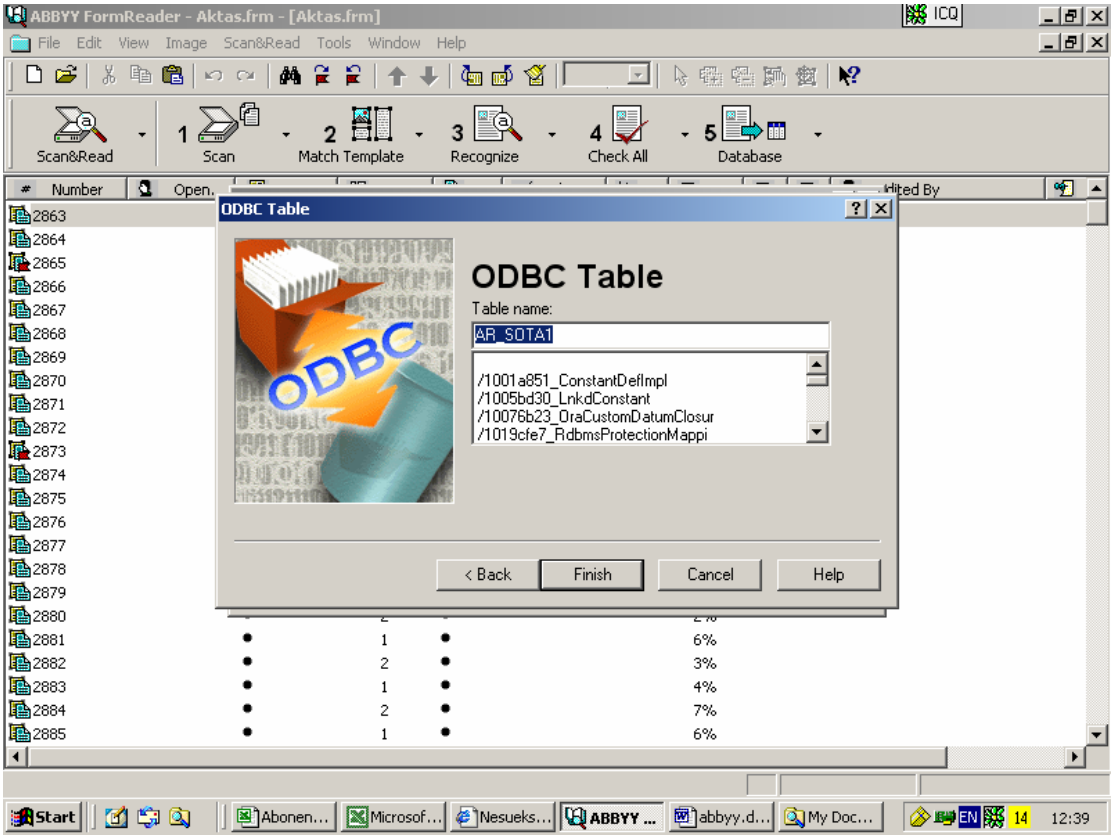
Taip pakeičiamos prisijungimo instrukcijos FormReader programai, kad ji siūstų duomenis nebe į Oracle DBVS, o į MS SQL DBVS. Darbų eiga pavaizduota sekančiuose paveikslėliuose.



6.18 pav. FormReader šablono savybių pasirinkimas







6.19 pav.FormReader šablonoODBC šaltinio ir lentelių pasirinkimas

6.4 Duomenų ir programų veikimo sutapimo testas

Duomenų ir programų veikimo sutapimas patikrintas įvertinant senosios ir naujosios IS atliekamų funkcijų sutapimą. Kaip ir aprašyta konceptualiame modelyje, pirmiausia buvo atliktas duomenų sutapimo testas. Pagal siūlomą metodą buvo atliktas lentelių kiekio sutapomo patikrinimas. Įsitikinus kad perkeltos visos lentelės, patikrinta ar perkelti ir sukurti visi reikalingi indeksai, trigeriai ir lentelėse atitinka įrašų kiekiai. Atlikus išvardintus etapus atsitiktinio pasirinkimo metodu buvo parinkti įrašai ir patikrina ar jie sutampa senojoje ir naujojoje duomenų bazėje.

Toliau atliktas sekantis etapas - funkcijų veikimo testas, išsiaiškinta ar visos funkcijos naujojoje sistemoje veikia taip pat kaip senojoje. Tiesiog senojoje ir naujojoje informacinėje sistemoje buvo atlikti veiksmai, tokie kaip:

- ✓ duomenų siuntimas iš FormReader programos į DB
- ✓ duomenų eksportas, sulyginant
- ✓ archyvavimas
- ✓ pateiktos užklausos ataskaitoms gauti

Tada buvo vykdomas taisymas, kol įsitikinta, kad senosios ir naujosios IS atlieka tokias pat funkcijas ir jas atlieka teisingai, t.y.:

- ✓ duomenų siuntimas iš FormReader programos pavyko,
- ✓ duomenų eksportas praėjo be klaidų ir buvo grąžinti nepraėję eksporto įrašai,
- ✓ visi paveiksliukai suarchyvuoti ir įrašai įrašyti į MS SQL duomenų bazę, kur saugomi skenuotų dokumentų paveiksliukai.
- ✓ Programoms pateiktos užklausos grąžino norimus rezultatus.

Atsiradus neatitikimams vykdomas sekantis aprašytas etapas.

6.5 Programų ir DB struktūros bei duomenų taisymas

Šiame etape taisomos klaidos, rastos klaidų paieškos etape, taip pat tobulinama programinė įranga, atsižvelgiant į vartotojų reikalavimus. Procesas yra iteratyvus, nes praktika rodo, kad klaidų bet kokiame programavimo ir projektavimo etape visą laiką pasitaiko.

7 IŠVADOS

- ✓ Šiame darbe išanalizuoti literatūroje pateikti ir praktikoje naudojami duomenų bazių reinžinerijos metodai, schemų derinimas, iteracinis reinžinerijos metodas, Gateway bei Chicken Little duomenų bazių reinžinerijos metodai.
- ✓ Analizės metu nustatyta, kad analizuotas schemų derinimo metodas naudotinas tada, kai jau suprojektuotos tiek naujoji, tiek senoji duomenų bazės ir šis metodas taikomas automatizuoti senosios DB struktūrą priderinti prie naujosios, naudojant automatinius arba kombinuotis derinimo įrankius bei bibliotekas.
- ✓ Apibendrinus likusius analizuotus metodus (iteracinis, gateway, Chicken Little) pastebėta, kad jie naudoja „data locator“, “gateway“ ir kitus specifinius komponentus, kuriuos reikia projektuoti ir palaikyti, šie metodai skirti tik objektiškai orientuotų sistemų reinžinerijai, taip pat jie neįvardina konkrečių etapų sudėties bei įrankių.
- ✓ Remiantis analizės rezultatais, pasiūlytas duomenų bazių reinžinerijos metodas aprašant konkrečius veiksmus kiekviename siūlomame etape, atsisakant papildomų komponentų, naudojamų išanalizuotuose metoduose.
- ✓ Sudaryta metodika tenkina praktinės duomenų bazių reinžinerijos reikalavimus, nes joje naudojamos standartinės modeliavimo priemonės ir įrankiai.
- ✓ Darbe pristatytas duomenų bazių reinžinerijos metodas aprašo Oracle DBVS paketų perkėlimą į Microsoft SQL serverio platformą, siūlomas duomenų ir programinės įrangos sutapimo testas, optimizuotas pagal konkrečius funkcinius reikalavimus.
- ✓ Atliekant darbą taip pat buvo įvykdytas realus eksperimentas su realiai egzistuojančiomis ir funkcionuojančiomis Oracle ir Microsoft SQL Server DBVS perkeliant šilumos objektų tikrinimo aktų duomenų bazę ir pritaikant php, asp, Visual Basic ir kitomis programavimo kalbomis sukurtą programinę įrangą naudoti naująją duomenų bazę.
- ✓ Eksperimentas parodė, kad šios metodikos reinžinerijos procesui leidžia palengvinti projektuotojų bei inžinierių-programuotojų darbą taikant papildomas CASE priemones, bei gauti geresnės kokybės duomenų bazių reinžinerijos proceso rezultatus, nes laiko ir darbo sąnaudos šiam procesui

gaunamos mažesnės, negu literatūroje parašomuose metoduose, nes nereikia kurti bei palaikyti sudėtingų komponentų, kurie aprašomi literatūroje.

- ✓ Magistrinio darbo tematika konferencijoje „Informacinė Visuomenė ir Universitetinės Studijos 2004“ (2004 04 15) buvo perskaitytas pranešimas. Straipsnis atspausdintas konferencijos pranešimų medžiagoje. (žr. priedus).

8 LITERATŪRA

1. Bianchi A., Caivano D, Visaggio G. Method and Process for Iterative Reengineering of Data in a Legacy System, 2000, *IEEE Software*, *IEEE* Comp. Soc.

2. Paradauskas B. Duomenų semantiniai modeliai, Paskaitų konspektai, Kaunas, 2004

3. Jahnke J. H.; Wadsack J. P. Varlet: Human − Centered Tool Support for Database Reengineering, 2002.12.03 [žiūrėta 2003-12-12]. Prieiga per internetą: http://www.uni-koblenz.de/~ist/RWS99/beitraege/JahnkeWadsack.pdf ,

4. Jahnke J. H.; Schäfer W.; Wadsack J.; Zündorf A. Managing Inconsistency in Evolutionary Database Reengineering Processes, 2002.12.03 03 [žiūrėta 2003-12-05]. Prieiga per internetą: http://www.uni-paderborn.de/fachbereich/AG/schaefer/Personen/Ehemalige/Zuendorf/socp.pdf

5. Shi-Ming Huang, Systematic Approach for Information Systems Reengineering, 2002.11.09 [žiūrėta 2004-02-05], Prieiga per internetą: http://cmca.mis.ccu.edu.tw/AE/material/isr/slidec/ch5.ppt

6. Paradauskas B. Pakartotinės inžinerijos uždaviniai kompiuterizuotose informacijos sistemose su kintamais funkciniais reikalavimais, 2002.11.12 [žiūrėta 2004-01-05] Prieiga per internetą: http://www.leidykla.vu.lt/inetleid/inf-mok/21/str9.html

7. Rahm E.; Bernstein P. A survey of approaches to automatic schema matching, 2001.10.09 [žiūrėta 2004-01-05] Prieiga per internetą: http://research.microsoft.com/~philbe/VLDBJ-Dec2001.pdf

8. Shanmugasundaram J.; Shekita E.; Barr R.; Carey M.; Lindsay B.; Pirahesh H.; Reinwald B. Efficiently publishing relational data as XML documents, 2001.04.15 The VLDB Journal The International Journal on Very Large Data Bases, Volume 10, Numbers 2-3 (September 2001) [žiūrėta 2004-01-25] Prieiga per internetą: http://content.ebsco.com/fulltext.asp?wasp=f0169mxvxpcqx2xydff7&ext=.pdf

9. Hainaut J-L; Englebert V.; Henrard J.; Hick J-M, Roland D. Evolution of database Applications: the DB-MAIN Approach , 1994.12 [žiūrėta 2004-01-15] Prieiga per internetą: http://www.fundp.ac.be/recherche/publications/fr/39257.html

10. Jahnke J.-H., Zündorf A.: Applying Graph Transformations To Database Re-Engineering 1999.10 [žiūrėta 2003-10-25] Prieiga per internetą: http://www.uni-paderborn.de/fachbereich/AG/schaefer/ag_dt/PG/Varlet/docs.html

11. Richard C., Ph.D. Incremental Migration from Enscribe to SQL: The Database Gateway Solution, 2004.05.01 [žiūrėta 2003-11-05] Prieiga per internetą: http://www.carrscott.com/ITUGArt.pdf .

12. Brodie M.; Stonebraker M. Migrating Legacy Systems: Gateways, Interfaces & the Incremental Approach, Morgan Kaufmann, San Francisco, 1995

13. Mossienko M., Khaschansky O., Antonov D., Smirnov O., Gubanov A. Towards managing environment dependence during legacy systems renovation and maintenance 2004.05.01 [žiūrėta 2003-11-12] Prieiga per internetą: http://www.iti.spbu.ru/publications/eng/pdf/MossienkoM_Managing.pdf

14. Oracle Migration Workbench White Paper 2004.05.01 [žiūrėta 2004-01-19] Prieiga per internetą: http://otn.oracle.com/tech/migration/workbench/pdf/omwb_wp.pdf

15. Putinas S. Duomenų bazių reinžinerijos procesų analizė ir plėtra// Informacinė visuomenė ir universitetinės studijos : 9 −oji magistrantų ir doktorantų konferencija [Kaunas, 2004 m. balandžio 15 d.]. Kaunas, 2004, p. 52−58.

9 SUMMARY

In this work current database reengineering methods (automatic schema matching, iterative reengineering, butterfly, Chicken Little) are evaluated. Rules and principles of these methods were generalized and represented via algorithmic structures. Also it highlights main advantages and disadvantages of these reengineering techniques. New database reengineering method is proposed describing specific actions for each step during this process. Evaluation has shown that proposed method ensures the higher quality and faster database reengineering process. Also an experimental part of this method is described, transferring data from really functioning Oracle database management system to Microsoft SQL Server DBMS platform. Experimental part completed successfully including software reengineering and external tools readjustment which interacts with reengineered DBMS.

10 PRIEDAI

DUOMENŲ BAZIŲ REINŽINERIJOS PROCESŲ ANALIZĖ IR PLĖTRA

Saulius Putinas
Kauno technologijos universiteta
Informacijos sistemų katedra

Šiame straipsnyje analizuojami esami duomenų bazių reinžinerijos metodai. Išskiriami analizuojamų metodų bei technologinių sprendimų privalumai bei trūkumai. Pasiūlomas duomenų bazės reinžinerijos metodas aprašant konkrečius veiksmus kiekviename šio proceso etape. Aprašomas eksperimentas, atliktas su realia duomenų baze, funkcionuojančia Oracle duomenų bazių valdymo sistemos pagrindu ir perkeliant ją į Microsoft SQL Server duomenų bazių valdymo sistemos platformą.

1 Įvadas

Paskutiniuosius keletą metų pastebimas ypač didelis susidomėjimas senų duomenų bazių atnaujinimu. Praktiškai šiuo momentu veikiančios senos sistemos yra tokios svarbios, kad tiek mokslininkai, tiek industrinė visuomenė bei programuotojai turi atkreipti dėmesį į dabar veikiančių sistemų problemas verslo srities aplikacijose. Yra paskaičiuota, kad dabar egzistuoja apie 100 milijardų programinio kodo eilučių senose informacinėse sistemose, kurias siekiant perprojektuoti konkuruoja keletas pagrindinių programų reinžinerijos paketų.

Taigi vieni ar kiti sistemos pasenimo požymiai verčia perprojektuoti visą sistemą, kad ją būtų galima naudoti toliau , o ne atsisakyti visos sistemos. Kadangi senoji sistema beveik visada yra suprojektuota ir realizuota per eilę metų, sudėjus ten labai daug žinių , todėl senosios sistemos perpratimas yra vienas pagrindinių faktorių vedančių prie sėkmingos sistemos reinžinerijos ir tolesnio sklandaus įmonės funkcionavimo. Teisingai pasirinktas reinžinerijos kelias gali pastebimai sumažinti įmonės išlaidas ir be sutrikimų atnaujinti senąją sistemą.

2 Esamų reinžinerijos metodų analizė

Daugelis literatūroje aprašomų sistemų reinžinerijos metodų apima visą programinės įrangos sistemą. Dėl šios priežasties visa sistema turi būti užšaldyta esamoje būsenoje , kol reinžinerijos procesas bus baigtas . Kitaip tariant nieko negalima keisti šio periodo metu. Taigi bendrai paėmus, jeigu bus atliekami senosios sistemos tobulinimo veiksmai, gausis taip, kad senoji ir naujoji sutemos nebeatitiks viena kitos ir programuotojai turės pradėti visą procesą iš naujo. Šioje situacijoje atsiranda uždaras ratas (ciklas) tarp senosios sistemos palaikymo ir reinžinerijos proceso.

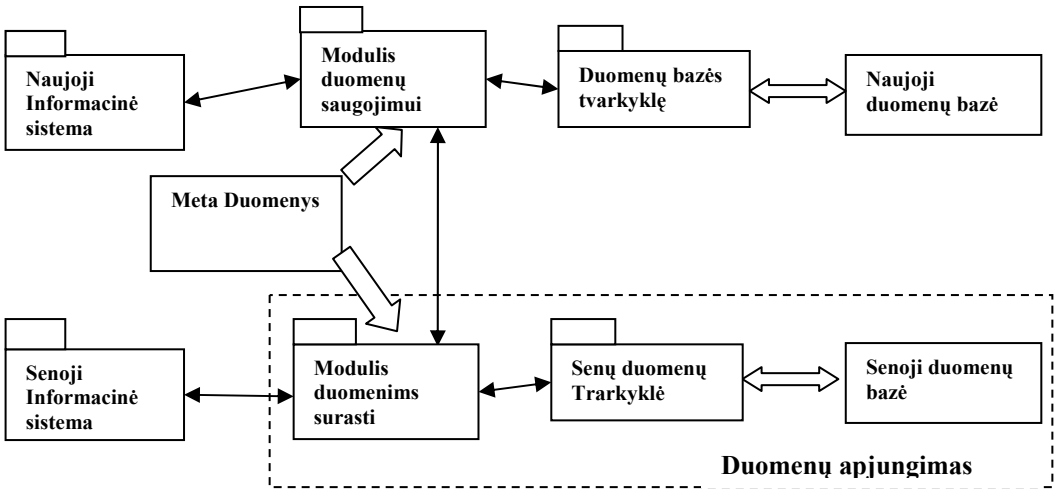
Kai kuriais atvejais tai problemai išvengti pakanka interpretuoti, kad senoji informacinė sistema yra juodoji dėžė ir jos nemodifikuojant reikia vykdyti reinžinerijos procesą. Tačiau šis metodas nepasiteisina, kai reinžinerijos procesas užsitęsia ir senąją IS keisti būtina.

Kita problema, su kuria susiduriama senų informacinių sistemų reinžinerijos metu yra tai, kad pasenusios būna ir duomenų bazių struktūros, todėl tenka pašalinti tuos senosios duomenų bazės struktūros trūkumus, kad naujoji sistema veiktų be senosios trūkumų.

Labai dažnai senoji sistema yra tokia didelė ir turi labai svarbių funkcionalumo aspektų, kad tiesiog jos pakeitimas naująja yra labai rizikingas. Tokiam atvejui autoriai Alessandro Bianchi, Danilo Caivano ir Giuseppe Vissagio siūlo iteracinį senosios sistemos reinžinerijos būdą[1]. Kiekvienos iteracijos metu pridedami skirtingo dydžio ir sudėtingumo komponentų patobulinimai, kai tuo tarpu senoji sistema gali dirbti visu savo pajėgumu ir funkcionalumu viso reinžinerijos proceso metu.

Remiantis iteraciniu reinžinerijos metodu, būtinai reikia užtikrinti senosios informacinės sistemos ir naujosios sistemos lygiagretų egzistavimą viso reinžinerijos proceso metu.

1 paveikslėlyje kaip tik ir pavaizduotos senoji informacinė sistema su senąja duomenų baze. Senoji informacinė sistema yra pilnai veikianti programinės įrangos sistema, kuri palaiko kai kurias organizacijos verslo funkcijas. Ši sistema yra svarbus taisyklių ir žinių rinkinys, sukauptas laikui bėgant. Senoji duomenų bazė sudaryta iš duomenų, kuriuos naudoja, pildo bei keičia senoji sistema. Tiek sistema tiek senoji duomenų bazė šiuo atveju yra pasenusios, jų palaikymas daug kainuoja ir neatsiperka ekonomiškai.



1 pav. Paralelinis naujosios ir senosios IS egzistavimas reinžinerijos metu

Naujoji duomenų bazių valdymo sistema išvis būtų bevertė, jeigu ji būtų paprasčiausias senosios atkartojimas. Tada įvedamas normalizacijos kriterijus, kuris kaip tik ir pagerina naujosios sistemos palaikymą, ir evoliucijos procesą. Duomenų bazės evoliucija visą laiką bus visiškai prieinama tiek senajai informacinei sistemai, tiek naujajai, todėl kad komponentai duomenų saugojimui („Data Banker“) ir duomenų paieškai („Data Locator“) [1] saugo jos fizinę struktūrą. Tokio reinžinerijos proceso rezultatas bus duomenų bazė, kuri priklausė senajai duomenų bazei. Rekonstruota ir moderni sistema duomenis pasiekia pateikdamos užklausas duomenų saugojimo („Data Banker“) komponentui. Šios užklausos apibrėžia ne tik reikalinga servisą (read, write, delete,...), bet taip pat apibrėžia duomenis, su kuriais servisas turėtų operuoti. Duomenų saugojimo komponentas analizuoja serviso užklausos struktūrą, interpretuoja jos turinį ir realizuoja kreipimąsi į fizinę duomenų bazę, kad pateikti rezultatus užklausai.. Pavyzdžiui serviso užklausa yra atidaryti duomenų bazę ir inicializuoti einamo įrašo poziciją: tai atliekama serviso pavadinimu open_service pagalba.[1]
Taigi , jeigu pasikeitė duomenų bazės struktūra, visą laiką pirmiausia tai atvaizduojama duomenų saugojimo ir duomenų paieškos komponentų parametruose. Pati aplikacija turi būti keičiama tik tada, kai esybės aplikacijos srities ribose pasikeitė. Duomenų apjungimas („Data Mapping“) architektūra užtikrina lygiagretų senųjų ir perprojektuotų duomenų egzistavimą. Kai senosios programos komponentai yra perprojektuoti, jie jau nebesikreipia į senąją duomenų bazę, o kreipiasi į naująją, todėl reinžinerijos proceso pabaigoje senoji duomenų bazė išvis nebenaudojama.

Kai programai senojoje sistemoje reikia duomenų, ji kreipiasi į senąją duomenų bazę naudodama *Duomenų paieškos* komponentą ir funkcionuoja kaip įprastai. *Duomenų paieškos* komponentas yra duomenų apjungimo komponentas, kuris konvertuoja duomenis iš senojo formato, į formatą, reikalingą naujajai sistemai ir atvirkščiai. Senoji sistema gali prieiti prie naujosios duomenų struktūros taip pat pasinaudodama *Duomenų paieškos* komponentu. *Duomenų paieškos* komponentas veikia senosios sistemos aptarnavimui, taip pat kaip *Duomenų saugojimo* komponentas veikia su naujai perprojektuota sistema. *Duomenų paieškos* kaip ir *Duomenų saugojimo komponentas* analizuoja serviso užklausos struktūrą, interpretuoja jos turinį ir kreipiasi į fizinę duomenų bazę, kad įvykdytų serviso užklausą. *Meta duomenų* komponentas, esantis tarp *Duomenų paieškos* kaip ir *Duomenų saugojimo* modulių yra duomenų bazė, kurioje saugoma visa informacija apie senąją duomenų bazę ir atitinkamus duomenis naujojoje duomenų struktūroje.

Pagrindinis iteracinio reinžinerijos proceso privalumas yra tai, kad tik vienas komponentas yra užšaldomas kiekvienu momentu. Kadangi atkarpa, per kurią perprojektuojamas vienas komponentas, yra sąlyginai trumpa, tai laikas, sugaištas kiekvieną kartą įterpiant naują komponentą, iteracinio reinžinerijos proceso metu yra labai trumpas. Praktikoje, perprojektuojant pastovius komponentus nereikia skubėti, nes tie komponentai nesikeičia naudojimo metu, tačiau nepastovius komponentus reikia apdoroti greitai, nes proceso eigoje jie jau gali pasikeisti. Tokiu atveju geriausias sprendimas yra sumažinti nepastovių komponentų dydžius prieš juos perprojektuojant.

Dar literatūroje pastebėti ir aprašomi panašūs metodai į anksčiau aprašytąjį yra Chicken Little Metodas ir Butterfly metodas[4][6]

Šie metodai, kaip ir aukščiau aprašytasis, paremti prielaida, kad duomenys senojoje informacinėje sistemoje yra svarbiausia sistemos dalis ir naujosios sistemos požiūriu duomenys nebus keičiami, tik bus keičiama jų semantika ir schema.

Pagal Chicken Little Metodą[4] senoji sistema gali bendrauti su naująja perkėlimo metu, naudojant tarpinį modulį, vadinamą „Gateway“. Chicken Little naudoja tiesioginį Gateway nukreipti užklausas į duomenų bazę, bei transformuoti rezultatus, gautus iš duomenų bazės, senajai informacinei sistemai. Šiam sprendimo variantui reikia senąją duomenų bazę duplikuoti ir perprojektuoti, kad ji atitiktu naujosios sistemos reikalavimus. Kreipiantis į duomenų bazę pagal šį metodą reikia kreiptis programa, kuri turi turėti priėjima prie abiejų duomenų bazių. Tačiau daug modernių savybių, tokių kaip ryšiai, trigeriai,vientisumas, nuoseklumas nepalaikomi senose duomenų bazėse, todėl negali būti naudojami, todėl sunku atnaujinti sistemos vientisumą.

Iteracinės reinžinerijos metodas tiesiog perkelia senus duomenis į naująją duomenų bazę, nenaudojant duplikavimo. Šio atveju vienas Duomenų paieškos komponentas nusako iš kur duomenys buvo paimti - iš senosios duomenų bazės ar iš naujai suprojektuotos. Kadangi duomenų bazės nėra duplikuojamos, atkrenta duomenų sudvejinimo problema ir problema kaip nustatyti iš kurios duomenų bazės tie duomenys paimti .

Butterfly metodologijoje tiek senoji tiek naujoji duomenų bazės gali egzistuoti, bet jos nesidalina duomenimis tarpusavyje. Ši metodologija orientuota į senų duomenų perkėlimą, o kuriama nauja informacinė sistema traktuojama kaip visiškai atskiras procesas. Senoji duomenų bazė perkeliama į naująją, tada senoji užšaldoma ir naudojama tik skaitymui. Visi duomenų pasikeitimai saugomi laikinoje pagalbinėje saugykloje. Taigi kiekvieną kartą, kai reikia duomenų, sistema turi kreiptis į abi duomenų bazes ir dar patikrinti laikinojoje pagalbinėje saugykloje, ar jie nepasikeitę. Tokiu atveju atsiranda rizika, kad reinžinerijos proceso metu ieškant duomenų vykdoma daug transakcijų ir tai gali labai pailginti paieškos laiką. Tada laikinoji duomenų saugykla turės tendenciją didėti diena iš dienos.

2 Siūlomas koncepcinis duomenų bazės reinžinerijos modelis

Atsižvelgiant į išnagrinėtus duomenų bazių reinžinerijos metodus galima gauti vieną modelį, kuris realizuoja visus keliamus reikalavimus duomenų bazės, kuri neįvertina paskirstytų duomenų bazių, reinžinerijai atlikti. Bet koks pasenusios duomenų bazės reinžinerijos procesas susideda iš dviejų stambių etapų: atvirkštines inžinerijos ir tiesioginės inžinerijos [2] - šiuo atveju: sugeneruotos schemos realizavimas naujojoje DBVS. Taigi pirmiausia bus aprašytas atvirkštinės duomenų bazės inžinerijos metodą.

Beveik visi atvirkštinės inžinerijos metodai yra apriboti tam tikromis būtinomis sąlygomis, kurios šiuo atveju yra tokios:

- visos konceptualiosios specifikacijos turi būti transliuojamos į duomenų struktūras ir apribojimus, pavyzdžiui, esant reliacinei duomenų bazei visos poaibio priklausomybės ir pirminiai raktai turi būti aiškiai ir tiksliai apibrėžti;
- pradinė schema negali turėti painių objektų ir ryšių struktūrų;
- pagal tam tikras taisykles parinkti vardai (pavyzdžiui, išorinio rakto ir rakto, į kurį tas išorinis raktas turi nuorodą, vardai turi sutapti);
- tariama, kad schema nebuvo optimizuota ar blogai suprojektuota.

Taip pat atliekant duomenų bazės reinžineriją reikia užsiduoti kažkokius kriterijus, tikslus, siekius, kurių bus siekiama viso reinžinerijos proceso metu. Pagrindinis atvirkštinės inžinerijos tikslas yra gauti geros kokybės duomenų bazę. Schemos transformavimo kokybė skiriasi nuo schemų ar duomenų modeliavimo kokybės, kuri dažniausiai remiasi intuityviomis sąvokomis – *skaitomumas, aiškumas ar išraiškingumas*. Šios sąvokos svarbios konceptualiajame duomenų bazių projektavime. Atvirkštinei duomenų inžinerijai būdingi kiti kriterijai, kuriuos ir išskirsime.

2.1 Reinžinerijos tikslai ir siekiai

Bendru atveju išskiriami tokie reinžinerijos tikslai ir siekiai[3]:

- *Teisingumas*: schema yra sintaksiškai teisinga, jei visi schemos konceptai yra tinkamai apibrėžti. Schema yra semantiškai teisinga, jei visi jos konceptai vartojami pagal jų apibrėžimą, pavyzdžiui, asociacija nėra modeliuojama kaip apibendrinimas ar atvirkščiai.
- *Primityvumas*: schema yra primityvi, jei visos schemos struktūros priskirtos ne daugiau kaip vienai realaus pasaulio abstrakcijai.
- *Minimalumas*: schema yra minimali, jei negalima pašalinti nė vieno schemos elemento, nepraradus informacijos. Schemos, kurios nėra minimalios, dažnai yra daug sunkiau suprantamos.
- *Tinkamumas***:** schema yra tinkama (angl. relevant), jei visi jos objektai ir ryšiai yra aiškiai ir tiksliai atvaizduoti duomenų bazės schemoje, naudojant atitinkamo modelio konceptus.
- *Normiškumas*: duomenų bazės schema turi tenkinti tam tikrą norminę formą.

Svarbiausias kokybės kriterijus yra semantinis teisingumas. Minimalumo ir primityvumo kriterijai yra glaudžiai susiję su semantinės abstrakcijos lygiu, kadangi pertekliškumas ir paslėpti objektai mažina abstrakcijos lygį. Tinkamumo kriterijus svarbus tuo, kad leidžia visus ryšius ir objektus tiksliai aprašyti modeliavimo konceptais, o ne paslėptais ryšiais ar loginiu suvokimu.

Be struktūros transformacijų, atvirkštinė inžinerija yra susijusi ir su egzempliorių bei taikomųjų programų pakeitimu. Dėl to reikia atsižvelgti į formalias transformacijos savybes. Ypač svarbus yra duomenų bazės turinys – galimų duomenų egzempliorių aibė, susieta su duomenų bazės schema. Idealiu atveju informacijos turinys transformuojamas nepakinta.

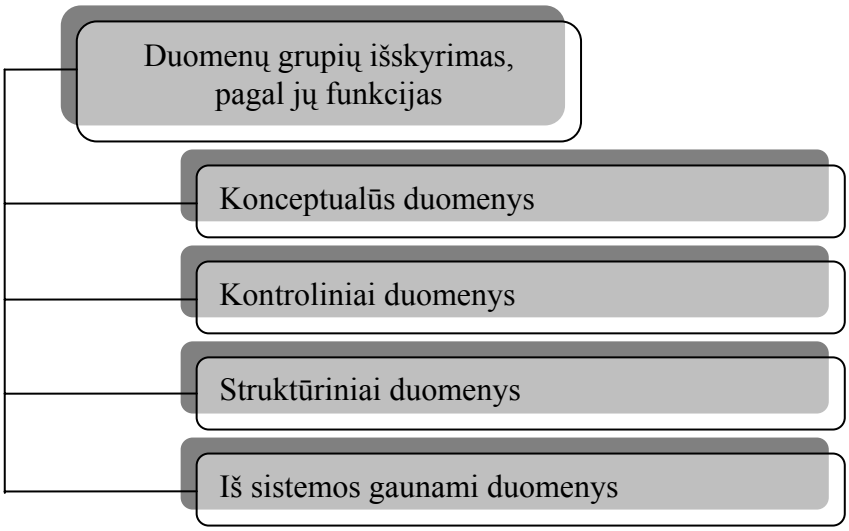
2.2 Reinžinerijos etapai

Remiantis aprašytomis metodikomis suformuosime etapus ir jų seką

6. Išanalizuoti duotą duomenų bazės struktūrą.
7. Duomenų perkėlimas iš senosios duomenų bazės į naująją
8. Pritaikyti senąsias programas naudoti naują duomenų bazę
9. Sutapimo testas
10. Taisymas

2.2.1 Duomenų bazės analizė

Pirmiausia , kaip jau buvo minėta, reikia išanalizuoti esamus duomenis, jų struktūrą, tarpusavio ryšiai.



2 pav. Duomenų grupavimas

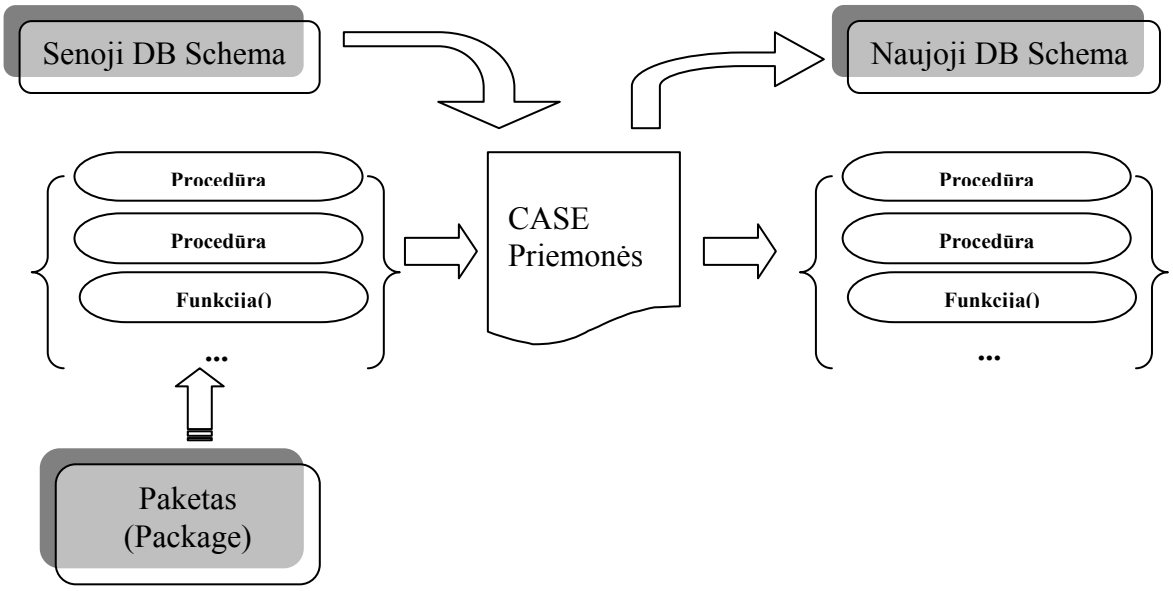
Kaip parodyta 2 paveiksle, reikia išanalizuoti pačius duomenis, susijusius su sistemos funkcionavimu. Kartais jie gali atrodyti paprasti sistemos vartotojui, tačiau sistemos reinžinerijos specialistui gali iškilti visokiu neaiškumų. Taigi pirmiausia reikia labai įsigilinti į dalykinę sritį.

Toliau iš eiles reikia įvertinti kontrolinius duomenis, tai yra , duomenis , kurie įrašomi atsiradus realaus pasaulio veiksniams, tokie kaip sistemos vartotojų teisės, vartotojų atliktų operacijų fiksavimas (logs). Tuo tikslu iš senosios sistemos reikia paimti visus kontrolinių duomenų fiksavimo principus ir numatyti jų papildymą naujojoje sistemoje taip, kad būtų gauti visi kontroliniai duomenys, kurių pageidauja naujos informacinės sistemos užsakovai. Dažniausiai tokiu atveju atsiranda daug poreikių, bet tai nėra blogai , nes analizuojant kontrolinius duomenis galima gauti daug vertingos informacijos, tokios kaip pavyzdžiui : kada vartotojas buvo prisijungęs prie duomenų bazės, ar kada buvo ištrintas vienas ar kitas įrašas. Tada dar lieka struktūriniai duomenys ir apskaičiuoti duomenys . Struktūriniai duomenys tai duomenys skirti organizuoti ir palaikyti duomenų struktūras, tokie kaip sekos, funkcijos ir t.t. O apskaičiuojami duomenys, tai programoje skaičiuojami ir generuojami duomenys.

Bendrai duomenų analizės etapo metu gali būti aptikti sinoniminiai duomenys. Tokiu atveju jie turėtų būti pervadinti ir išsaugoti, arba jeigu tai besidubliuojantys duomenys, tokie įrašai turėtų būti pašalinti.

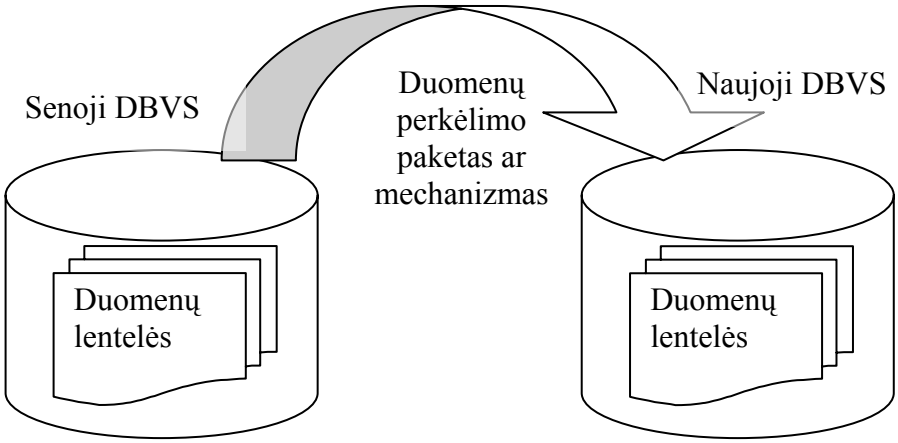
2.2.2 Duomenų perkėlimas

Išanalizavus duomenis juos perkeliame etapais. Pirmiausia turėtumėme kelti informacinėje sistemoje tiesiogiai naudojamus duomenis ir tik po to sisteminius bei struktūrinius duomenis. Duomenų bazės schemos perkėlimui gali būti naudojami įvairios case priemones ar paketai. Kaip pavyzdi galima paminėti Visio, Rational Rose paketai. Šio paketo pagalba galima būtų perkelti ne tik Duomenų bazės schemas, bet ir Saugomas procedūras ar funkcijas. Tačiau čia kartais susiduriame su nesuderinamumais. Pavyzdžiui Oracle DBVS yra tokie objektai, vadinami „Package“, o Microsoft SQL DBVS gali būti kuriamos tik Saugomos procedūros ir funkcijos. Oracle Paketo (angl. Package) viduje gali būti aprašytos ir realizuotis Saugomos procedūros ir funkcijos. Tokių objektų perkėlimo automatizuoti nepavyks, todėl reikia imtis priemonių, kurios tai įgalintų padaryti su mažiausiomis resursų sąnaudomis. Šiuo atveju, ko gero tektų transformuoti Oracle paketus į atskiras Saugomas procedūras ir funkcijas , kaip parodyta 3 paveiksle, ir tada jas perkėlinėti su Case įrankiais .



3 pav. Duomenų perkėlimas Visio paketo pagalba

Patys duomenys gali būti perkeliami naudojantis bet kokiu patogiu įrankiu, kuris įeina į vieną ar kitą paketą, kaip parodyta 4 paveiksle. Svarbu atkreipti dėmesį į duomenų integralumo užtikrinimą ir keliamų duomenų teisingumą.

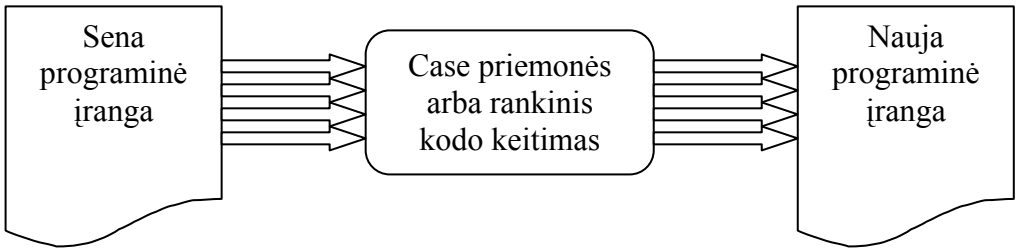


4 pav. Duomenų perkėlimas

Visi likę perkėlimo darbai negali būti atliekami automatiškai, todėl tiek ryšiai tarp naujosios duomenų bazės lentelių, tiek vartotojų teisės turėtų būti perkeltos rankomis.

2.2.3 Programinės įrangos pritaikymas

Kiekviena senoji programa, kuri naudos naujos duomenų bazės duomenis, turi būti pritaikyta, kad ji tuos duomenis sėkmingai paimtų ir, jeigu reikia, atliktu kitus veiksmus su jais. Tai reiškia, kad kiekviena programa individualiai turi būti peržiūrėta ir pakeistos jungimosi instrukcijos prie naujosios duomenų bazės. Tokiu atveju, jei jungiamasi iš tos pačios aplikacijos, tai tiesiog reikia pakeisti prisijungimo instrukcijas arba naudojant vieną ar kitą įrankį arba rankiniu būdu, jei case priemonės neužtikrina galimybės parinki specifikaciją arba išvis nėra tinkamų įrankių . Tais pačiais metodais galima programas perdaryti, jei naudojamos skirtingos programavimo kalbos tiesiog reikia perrašyti programos kodą kita kalba, įvertinant prisijungimo parametrus. Schema, pagal kurią galima pritaikyti aplikacijas naujajai DBVS, pavaizduota 5 paveiksle.



5 pav. Programos kodo keitimas, jungiantis prie naujos DBVS

Konkrečiai realizuotas pavyzdys pateiktas aprašant metodo praktinę realizaciją.

2.2.4 Sutapimo testas

Nuo šio etapo reinžinerijos procesas beveik baigtas , liko tik analizė ir klaidų paieška. Taigi šiame etape yra lyginamos senosios ir naujosios programos veikimas, naujosios programos išduodami rezultatai bei jų korektiškumas.

2.2.5 Redagavimas

Kiekvieną atsiradusią klaidą, aptiktą testuojant reikia nuodugniai išanalizuoti ir naujoji sistema turi būti pataisyta, kad pašalinti tuos trūkumus. Po šio žingsnio vėl reiktų praleisti sutapimo testus ir tol kartoti tuos veiksmus, kol klaidų nebebus pastebėta.

3 Reinžinerijos modelio eksperimentinis tyrimas

Remiantis aukščiau aprašytais ir pasiūlytais reinžinerijos etapais, galima aprašyti praktinius šio reinžinerijos modelio žingsnius. Pirmiausia pagal planą turi būti išanalizuoti ir keliami duomenys. Pagal aukščiau pateiktus kriterijus jie turi būti išanalizuoti ir sugrupuoti. Sekančiame etape pagal duomenų svarbą ir funkcinę priklausomybę jie keliami pasinaudojant vienu ar kitu įrankiu. Šiuo atveju keliant duomenis iš Oracle DBVS į Microsoft SQL serverį naudojamas SQL serveryje esantis DTS paketas. Jo vedlių pagalba nesunkiai galima perkelti lentelių duomenis, prieš tai jas sukūrus ir perkėlus tokius elementus kaip trigeriai, kurie užtikrina keliamų duomenų teisingumą ir tikslumą. Kartais, jei kitaip negalima, trigerius galima laikinai išjungti. Tokie objektai, kaip vartotojų teisės ir slaptažodžiai keliami tiesiog rankomis, atsižvelgiant į poreikius. Kadangi senojoje duomenų bazės struktūroje nebuvo ryšių, tai juos reikia sukurti rankomis.

Pagal pasiūlyta modelį toliau turi sekti programinės įrangos pritaikymas darbui su nauja DBVS.Žemiau pateiktas pavyzdys, kaip jungiamasi prie vienos ar kitos duomenų bazės iš PHP programavimo kalbos.

Prisijungimas prie Oracle DBVS su PHP programavimo kalba

```
$conn = ora_logon("username@neptunas", "passw");
$curs = ora_open($conn);
ora_commitoff($conn);
$query = "select * from automat.dbf where kodas like '$kodas"._."'
and paskirtis='$paskirtis'";
ora_parse($curs,$query );
ora_exec($curs);
ora_fetch_into($curs,$row);
```

Prisijungimas prie Microsoft SQL Server 2000 su PHP programavimo kalba

```
MSSQL_CONNECT($hostname,$username,$password) or DIE("DATABASE FAILED
TO RESPOND.");
mssql_select_db($dbName) or DIE("Nerasta lentele");
$query = "select meter.meterid, complex.complexnumber, meter.name
from complex
inner join meter on meter.complexid=complex.complexid";
$result = MSSQL_QUERY($query);
$number = MSSQL_NUM_ROWS($result);
$name =  mssql fetch row ($result);
```

6 pav. Programos kodo ištraukos, jungiantis prie Oracle ir SQL Server 2000 DBVS

Kaip galima pastebėti iš pateiktų pavyzdžių, skiriasi tiktai prisijungimo eilutė ir ją sudarančių parametrų kiekis bei eiliškumas, visa kita dalis lieka identiška. Kintamajam `$query` priskiriamas užklausos tekstas, kuris perduodamas serveriui. Tiek vienu, tiek kitu atveju , kaip rezultatas yra grąžinamas kursorius. Norint gauti rezultatą iš kursoriaus nuskaitoma (`fetch`) po vieną užklausos grąžintą įrašą, o toliau su juo gali būti atliekamos reikalingos operacijos.

Kaip visi iki šiol aprašyti etapai atlikti lieka iteracinis procesas- tai programos testavimas pagal sudarytą planą ir aptiktų klaidų ar netikslumų taisymas. Vienas iš plano variantų yra išbandyti visas IS atliekamas funkcijas su keliais duomenų rinkiniais. Šio etapo metu turi būti nesunaikinta senoji IS, kad būtų galima palyginti rezultatus

5 Išvados

Šiame straipsnyje išanalizuoti literatūroje pateikti ir praktikoje naudojami duomenų bazių reinžinerijos metodai. Pasiūlytas duomenų bazės reinžinerijos modelis aprašant konkrečius veiksmus kiekviename šio

proceso etape. Aprašomas eksperimentas, kuris buvo atliktas realia duomenų baze, funkcionuojančia Oracle duomenų bazių valdymo sistemos pagrindu ir perkeliant ją ant Microsoft SQL Server duomenų bazių valdymo sistemos platformos. Ateityje reikia patobulinti pasiūlytą reinžinerijos metodą bei eksperimentinę darbo dalį.

Literatūros sąrašas

[1] [**A. Bianchi, D. Caivano, G. Visaggio**].[Method anr Process for Iterative Reengineering of Data in a Legacy System].[*IEEE Software, IEEE Comp. Soc*], [2000] [*http://cmca.mis.ccu.edu.tw/AE/material/isr/slidec/ch5.ppt*] ,[2002.11.9]

[2] [**B. Paradauskas**].[Duomenų semantiniai modeliai].[*Paskaitų konspektai*],[2003]

[3] [**B.Paradauskas**].[Pakartotinės inžinerijos uždaviniai kompiuterizuotose informacijos sistemose su kintamais funkciniais reikalavimais].[*http://www.leidykla.vu.lt/inetleid/inf-mok/21/str9.html*],[2002.11.12]

[4] [**E. Rahm, P-A. Bernstein**].[A survey of approaches to automatic schema matching]. [*http://research.microsoft.com/~philbe/VLDBJ-Dec2001.pdf*],[2001.10.09]

[5] [**J-H. Jahnke, J-P.Wadsack**].[Varlet: Human-Centered Tool Support for Database Reengineering] . [*http://www.uni-koblenz.de/~ist/RWS99/beitraege/JahnkeWadsack.pdf*], [2002.12.03]

[6] [**J-H. Jahnke, W. Schäfer, J. Wadsack, A. Zündorf**].[Managing Inconsistency in Evolutionary Database Reengineering Processes]. [*http://www.un-paderborn.de/fachbereich/AG/schaefer/Personen/Ehemalige/Zuendorf/socp.pdf*],[2002.12.03]

[7] [**J.-H. Jahnke and A. Zündorf**].[Applying Graph Transformations To Database Re-Engineering]. [*http://www.uni-paderborn.de/fachbereich/AG/schaefer/ag_dt/PG/Varlet/docs.html*],[1999.10]

[8] [**J. Shanmugasundaram, E. Shekita, R. Barr, M.Carey, B. Lindsay,H. Pirahesh, B. Reinwald**].[Efficiently publishing relational data as XML documents]. [*The VLDB Journal The International Journal on Very Large Data Bases*] .[2001.04.15, Tomas 10]

[9] [**J-L Hainaut, V. Englebert, J. Henrard, J-M Hick, D. Roland**].[*Evolution of database Applications: the DB-MAIN Approach*].*[http://www.fundp.ac.be/recherche/publications/fr/39257.html*],[1994.12]

[10] [**S-M Huang**]. [Systematic Approach for Information Systems Reengineering].

Database reengineering process analysis and development

In this article, current database reengineering methods are evaluated. Also it highlights main advantages and disadvantages of these reengineering techniques. New database reengineering method is proposed describing specific actions for each step during this process. Also an experimental part of this method is described, transferring data from really functioning Oracle database management system to Microsoft SQL Server DBMS platform.