KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Aleksas Kekys

Duomenų bazės schemos SQL aprašo generavimas funkcinių reikalavimų specifikacijos pagrindu

Magistro darbas

Darbo vadovas
doc. dr. R. Butleris

Kaunas, 2005

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. R. Butleris
2005 05 06

Duomenų bazės schemos SQL aprašo generavimas funkcinių reikalavimų specifikacijos pagrindu

Magistro darbas

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

Darbo vadovas
doc. dr. R. Butleris

2005 05 23

Konsultantas
Informatikos fakulteto j.m.d.
T. Danikauskas
2005 05 23

Recenzentas
dr. doc. A. Lenkevičius
2005 05 23

Darbą atliko
IFM9/2 gr. stud.
Aleksas Kekys
2005 05 23

Kaunas, 2005

Turinys

Paveikslų sąrašas

Lentelių sąrašas

1 Įvadas

Sparčiai besivystant programinės įrangos kūrimo technologijoms ir reikalavimams programinei įrangai, sunku įsivaizduoti didelės apimties programinės įrangos sistemą funkcionuojančią be duomenų saugyklos. Daugelyje projektavimo metodologijų, duomenų bazės lygio projektavimas egzistuoja kaip atskiras ir nepriklausomas etapas[7], kurio sėkmingumas priklauso nuo duomenų bazės modeliuotojo žinių lygio bei intuityvumo. Taigi natūralu, kad projektuojant dideles informacines sistemas (IS), pavyzdžiui, atliekančias įmonių kompiuterizuotų funkcijų valdymą, apskaitos vedimą ar realizuojant didelės įmonės turinio valdymo sistemą, dažnai neišvengiama duomenų bazių analitiko klaidų.

Šio tipo klaidoms išvengti, vertėtų duomenų modelį įtraukti į bendrą PĮ kūrimo procesą, ne kaip atskirą menkai su bendru procesu susijusį etapą, o kaip vieną iš fundamentalių etapų, nuo kurių priklauso projekto sėkmė. Idealiu atveju, duomenų bazės konceptualusis modelis galėtų būti sukuriamas identifikavus pagrindinius objektus bei duomenų srautus funkciniuose reikalavimuose keliamuose sistemai. Tada, duomenų bazių projektuotojas galėtų įvesti tinkamus pakeitimus, jau galutinei siūlomai duomenų bazės schemai.

Vienas būdų šiai problemai spręsti yra CASE metodai. Automatizuotas programinės įrangos projektavimas (CASE – computer-aided software engineering) yra vienas iš metodų, leidžiantis užtikrinti kokybišką duomenų bazių modelių sukūrimą. Šios sistemos priartina informacijos sistemos (IS) kūrimo procesą prie vartotojo lygmens. CASE priemonėmis yra organizuojamas ir valdomas programinės įrangos kūrimas ir tai ypač naudinga dideliuose projektuose, kadangi tai padeda susisteminti IS kūrimo procesą, įvesti griežtesnę jo kontrolę. Kauno technologijos universiteto Informacijos sistemų katedroje plėtojamas *funkcinių reikalavimų specifikavimo metodas* bei jį realizuojantis CASE įrankis. Šio metodo esmė – kurti sistemą remiantis pirminiais vartotojo funkciniais reikalavimais sistemai.

Šio darbo tikslas – išanalizuoti projektuojamos informacinės sistemos duomenų saugyklos loginio duomenų modelio pavaizdavimo esybių ryšių diagrama bei konvertavimo į SQL DDL aprašą (reliacinę schemą) principus ir apribojimus. Suprojektuoti loginio duomenų modelio, suformuoto funkcinių reikalavimų specifikacijos pagrindu, vaizdavimo esybių ryšių diagrama bei konvertavimo į SQL DDL aprašą programinį modulį ir patikrinti CASE įrankyje realizuoto modelio teisingumą eksperimentiniu būdu. Sukurtą CASE įrankio modulį palyginti kitų duomenų modeliavimo sistemų kontekste.

Analizės dalyje išnagrinėti duomenų modelių sudarymo, konvertavimo metodai, keletas CASE metodų, pateiktas šių metodų įvertinimas pagal pasirinktus kriterijus. Kadangi nagrinėjamas modelis yra funkcinių reikalavimų specifikacijos metodo sudedamoji dalis, pateiktas trumpas šio metodo aprašymas ir jo pagrindiniai privalumai.

Duomenų modelio sudarymo etapo dalyje aprašyti siūlomi CASE duomenų metamodelio papildymai. Taipogi pasiūlytas duomenų modelio saugomo CASE įrankio duomenų saugykloje analizės bei SQL DDL aprašo (reliacinės schemos) generavimo procesas, jo sudarymo principai ir taisyklės, vieta funkcinių reikalavimų specifikavimo metodo kontekste.

Duomenų bazės schemos SQL aprašo generavimo modulio realizacijos skyriuje pateikti fragmentai iš programinio produkto kūrimo metu sudarytų dokumentų. Plačiau pateiktos reikalavimų ir architektūros specifikacijos, kurios sudarytos reikalavimų išgavimo bei sistemos projektavimo metu.

Eksperimentinėje dalyje pateiktas realizuotos programinės įrangos kokybės įvertinimas, bei palyginimas su kitomis panašaus pobūdžio sistemomis.

Prieduose pateikiama straipsnio, paskelbto konferencijos pranešimo medžiagoje, kopija, funkcinių reikalavimų specifikavimui naudojama metaduomenų bazė ir jos aprašas, realizuoto CASE įrankio modulio vartotojo dokumentacija.

2 Bendra duomenų modelių sudarymo metodų bei įrankių analizė

CASE įrankių naudojimas, pakartotinio panaudojimo programinės įrangos, objektiškai – orientuoti įrankiai, prototipų kūrimas ir ketvirtos kartos generatoriai padeda IS kūrėjams sukurti veikiančią sistemą daug greičiau, nei taikant tradicinius struktūrinius metodus. Šis procesas yra apibrėžiamas terminu greitas taikomųjų programų vystymas (angl. Rapid Aplication Developement, sutrumpintai RAD). Žemiau pateikiami keli IS kūrimo metodai, kurių kiekvienas atitinka tam tikrą metodologiją.

2.1 Oracle CASE metodas

Oracle CASE metodas yra struktūrinis automatizuotas IS vystymo metodas. Šį metodą 1992 metais aprašė R.Baker ir C.Longman. Pagal šį metodą sistemos gyvavimo ciklas susideda iš septynių etapų. Kiekvienas etapas susideda iš užduočių, kurių metu gaunamas vienas ar daugiau rezultatų. Kiekviena užduotis dar smulkinama į veiksmus. Konkrečiam projekte nebūtina vykdyti visus veiksmus. Dalis veiksmų yra skirti valdyti, kontroliuoti procesą, kiti – praktiniams rezultatams gauti. Sistemos gyvavimo ciklo etapai ir pagrindiniai jų rezultatai pateikiami 1 pav. Strategijos (planavimo) etapas. Šio etapo tikslas yra sukurti nagrinėjamos organizacijos srities bendrą veiklos modelį bei sudaryti ir patvirtinti informacijos sistemos kūrimo planą. Etapo išskiriamos taikomosios sistemos.[9] Remiantis veiklos prioritetais, tikslais, veiklos funkcijų tarpusavio priklausomybėmis, poreikiais naudoti informaciją bei turimais ištekliai sudaromas taikomųjų sistemų kūrimo planas. Esminiai strategijos (planavimo) etapo rezultatai :

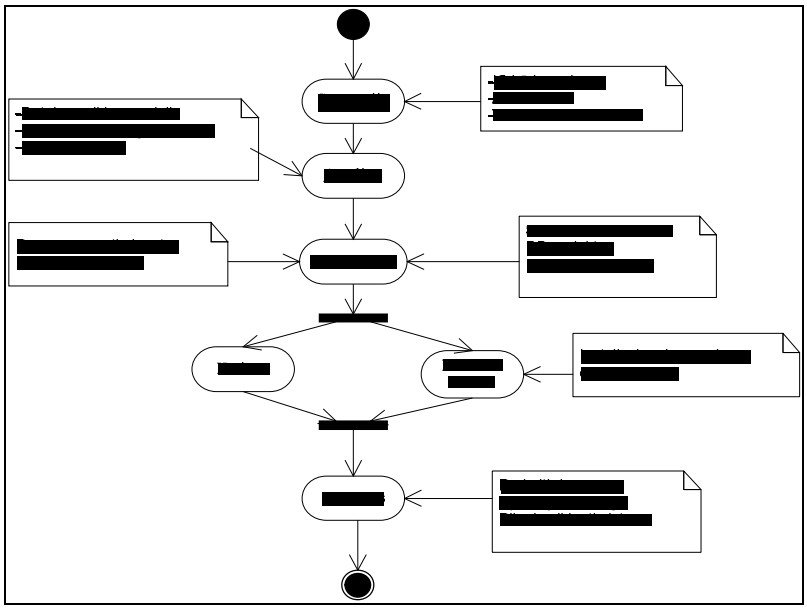
- veiklos krypties apibūdinimas, *bendras esybių ryšių modelis*, bendra veiklos funkcijų hierarchija, kuriamos sistemos ribų, apibrėžimas, galima sistemos architektūra, sistemos kūrimo planas.

Analizės etapas. Etapo tikslas – išplėsti ir detalizuoti strategijos etapo rezultatus tiek, kad veikla būtų visiškai sumodeliuota, nuosekliai aprašyta, modeli praktiškai pritaikomas bei užtikrinti visi duomenys, reikalingi IS projektavimui. Pagrindiniai analizės etapo rezultatai :

- detalus esybių – ryšių modelis, detalus veiklos funkcijų hierarchijos modelis, detali veiklos funkcijų – esybių sąveikos matrica, duomenų galimos apimtys, veiklos funkcijų vykdymo dažnumas, darbo stiliaus apibrėžimas, kriterijai, kuriais remiantis sprendžiama ar sistema priimtina vartotojams, duomenų audito/kontrolės ir saugojimo/atstatymo reikalavimai, pradinis diegimo planas, peržiūrėtas sistemos kūrimo planas.

Projektavimo etapas. Šiame etape, įvertinus techninės galimybes ir apribojimus, randamas analizės metu suformuluotų reikalavimų realizavimo planas. Etapo rezultatai :

- programinių modulių aprašai, duomenų bazės struktūra, sistemos architektūra, programų specifikacijos, suderintas diegimo planas, sistemos testavimo planas.



1 pav. CASE metodo sistemų gyvavimo ciklo etapai ir pagrindiniai rezultatai

Kūrimo/realizavimo etapas/ Šio etapo tikslas – remiantis projektu, sukurti veikiančią, testuotą programinę įrangą, rankines procedūras :

- sukurta ir suderinta duomenų bazė, dirbančios, testuotos programos, peržiūrėtas diegimo planas, sistemos testavimo rezultatai, pradinės žinios apie sistemos efektyvumą.

Vartotojo dokumentacijos etapas. Vartotojui pateikiama sistemos bei aptarnavimo operacijų dokumentacija, skirta testavimui, apmokymui, eksploatavimui. Etapo rezultatai :

- dokumentacija vartotojui ir sistemos aptarnavimo operacijų dokumentacija.

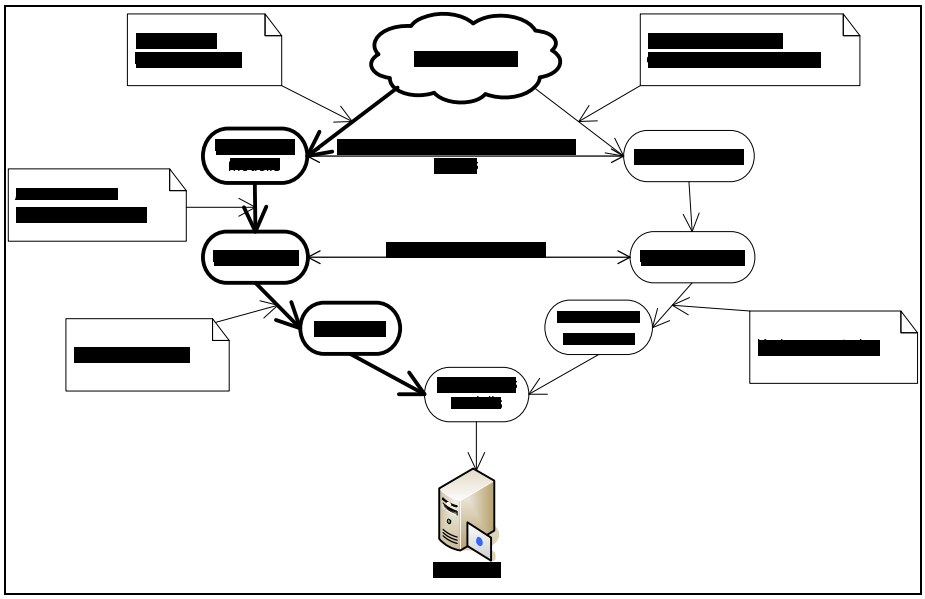
Diegimo etapas. Apmokomi vartotojai, įdiegiama programinė įranga ir rankinės procedūros. Etapo rezultatai :

- apmokyti vartotojai ir aptarnaujantis personalas, instaliuoja ir veikianti sistema, pakrauti duomenys, aptiktų operacinių klasių sąrašas, pabaigta sistemos dokumentacija.

Šie atskiri etapai turi būti aiškiai apibrėžti ir atskirti, nes tai būtina projekto kontrolei ir valdymui palengvinti.

Vartotojo funkciniams reikalavimams specifikuoti *Oracle* CASE metode naudojami keli modeliai : esybių – ryšių modelis, funkcijų hierarchija, duomenų srautų modelis, funkcijų priklausomybių modelis, sistemos būsenų modelis. Darbui pagal Oracle CASE metodą yra sukurta Oracle CASE programinė įranga : CASE dictionary (priemonė skirta CASE metodo etapų informacijai saugoti),

CASE Designer (priemonė darbui su grafine projekto informacija) ir CASE Generator (priemonių rinkinys ekraninių formų, ataskaitų generavimui). Sistemų kūrimo Oracle CASE priemonėmis schema pateikiama paveikslėlyje 2 pav. :



2 pav. Sistemų kūrimo su Oracle CASE priemonėmis schema

Duomenų modelių, bei duomenų bazių schemų sudarymui yra naudojami šie CASE designer įrankiai: entity relationship diagrammer, database design transformer, design editor [10].

Naudojant entity relationship diagrammer įrankį sukuriama esybių ryšių diagrama (konceptualusis duomenų modelis). Ši diagrama vaizduoja projektuojamos sistemos informacijos šaltinius ir šių informacijos šaltinių loginį ryšį. Esybių ryšių diagramą sudaro duomenų analitikas, remdamasis turimomis žiniomis apie projektuojamą informacinę sistemą bei turimais duomenų modelių sudarymo įgūdžiais. Sukurtas modelis saugomas CASE dictionary įrankyje.

Naudojant database design transformer įrankį, remiantis anksčiau sukurta esybių ryšių diagrama yra sukuriamas ir apdorojamas duomenų bazės projektas. Duomenų bazės projektas yra sudarytas iš aibės duomenų bazės objektų : lentelių aprašų (skirtų esybių egzemplioriams saugoti), lentelių stulpelių aprašų (skirtų atributams saugoti) bei išorinių raktų apribojimų (skirtų ryšiams tarp esybių realizuoti). Kaip ir esybių ryšių modelyje, duomenų bazės projektas yra saugomas CASE dictionary įrankyje. Duomenų bazės projektas sukurtas database design transformer įrankiu dažnai vadinamas pirmarūšiu, tačiau gali būti ištobulinti iki realizacijos lygmens.

Naudojant design editor įrankį, galima sukurti fizinį duomenų bazės modelį (*server model diagram*). Kiekvienas fizinis duomenų modelis tai aibė duomenų bazės objektų kurie gali būti sukurti fizinėje duomenų bazėje. Naudojant design editor įrankį, galima papildyti pirmarūšio

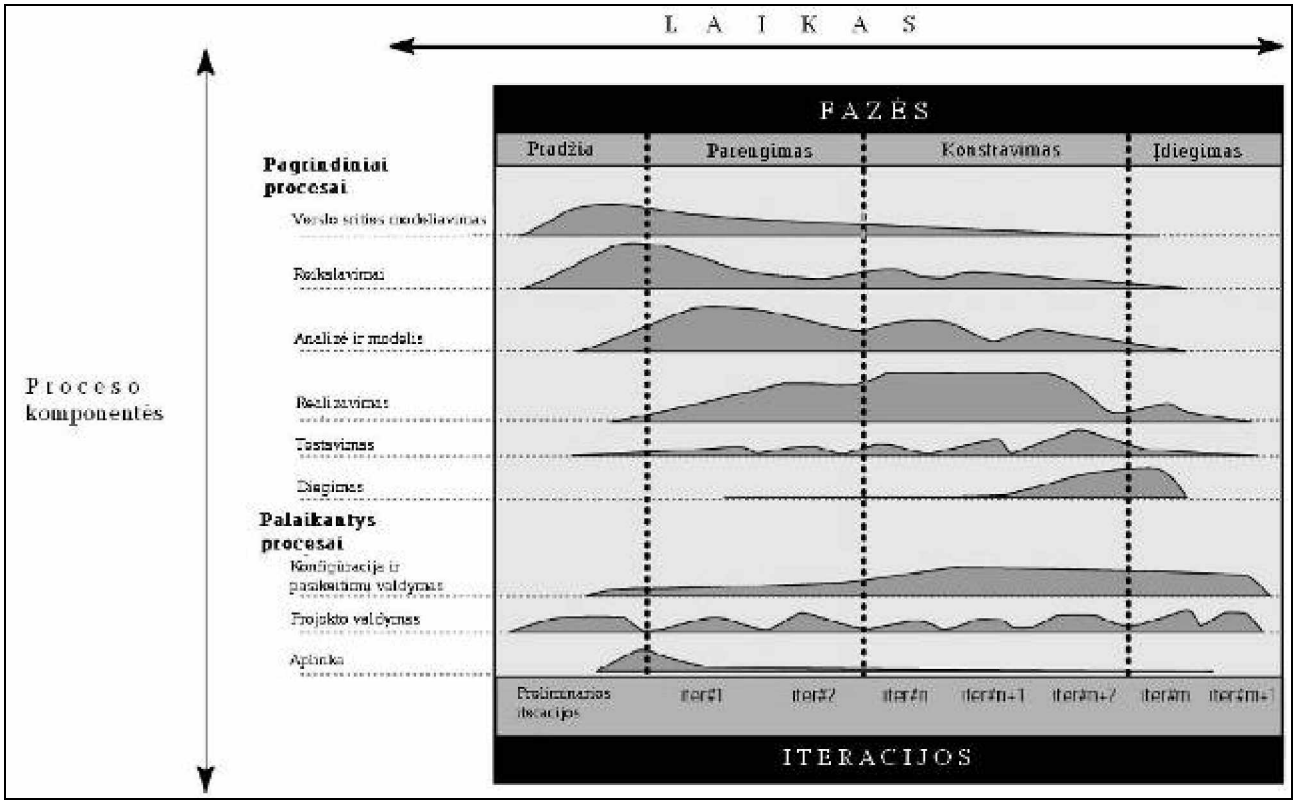
duomenų bazės projekto informaciją, taip sukuriant fizinį duomenų bazės modelį. Sukūrus fizinį duomenų modelį, iš šio galima sugeneruoti SQL DDL aprašą, kuris naudojamas lokaliai ar nuotolinei duomenų bazei sukurti.

2.2 RUP (rational unified process) metodas

Unifikuotas Rational Procesas (RUP) yra vienas iš labiausiai paplitusių ir remiamų programinės įrangos kūrimo metodų (Rational Software Corporation, 2000). Skirtingai nei dauguma kitų metodų, RUP yra komercinis produktas, kuriamas ir remiamas Rational kompanijos. RUP atsirado sujungus Booch , Rumbough OMT ir Jacobson „Use case driven approach“ („panaudojimo atvejais grįstą“) metodus. Iš čia kilo ir pavadinimas „Unifikuotas Rational procesas“. Unifikuotas Rational procesas yra panaudojimo atvejų valdomas, architektūra grindžiamas, iteracinis, vykdomas palaipsniui, riziką mažinantis projektavimo procesas. Jis labai nuosekliai ir pakankamai formaliai aprašo visą programinės įrangos kūrimo procesą, pradedant veiklos procesų modeliavimu, baigiant sukurtos sistemos pateikimu vartotojui. Jame modeliams sudaryti naudojama UML (*unified modeling language)* notacija.

Unifikuotas Rational procesas aprašomas dviejose dimensijose (3 pav.) : laiko ir proceso komponenčių. Programinės įrangos gyvavimo ciklas susideda iš 4 nuoseklių fazių :

pradžios, parengimo, konstravimo, įdiegimo. Kiekvienoje fazėje gali būti atliktos kelios iteracijos.



3 pav. RUP fazės ir etapai

Pradžios fazėje nustatoma kokią organizacijos veiklos dalį turi palaikyti kuriama IS. Tai atliekama identifikavus išorines esybes (aktorius, kurie bendrauja), ir šio bendravimo prigimtį, t.y. identifikuojami visi sistemos panaudojimo atvejai.

Paruošimo fazės tikslas – išanalizuoti dalykinę sritį, parinkti pagrindinę architektūrą, sudaryti projekto planą ir pašalinti rizikingiausius elementus iš projekto. Architektūriniai sprendimai priimami supratus visą sistemą. Tai reiškia, kad turi būti aprašyti visi panaudojimo atvejai ir įvertinti tam tikri apribojimai.

Konstravimo fazėje palaipsniui sukuriamas visas produktas, kuris gali būti įdiegtas. Tai reiškia, kad šioje fazėje aprašomi likę panaudojimo atvejai sukuriamas visas produktas, kuris gali būti įdiegtas. Tai reiškia, kad šioje fazėje aprašomi likę panaudojimo atvejai, užbaigiamas projektavimas, realizavimas ir testavimas.

UML duomenų modeliavimo profilis naudojamas RUP pateikia eilę stereotipų duomenų bazių projektavimui. Šis profilis, kurį palaiko *Data Modeler* įrankis, padidina projektuotojo galimybes, nes leidžia vienu CASE įrankiu projektuoti ir tarpusavyje suderinti taikomąsias programas ir duomenų bazes. *Data Modeler* leidžia projektuoti bazes loginiame ir fiziniame lygyje. Jo privalumai: galima pavaizduoti visą sistemos architektūrą, suderinant DB modelį su objektiniu visos sistemos modeliu; galima atskirti loginį ir fizinį DB projekto lygį bei aprašyti elgseną – operacijas, apribojimus ir verslo taisykles.

Loginį duomenų modelį atitinka klasių diagrama, fizinį – duomenų modelio diagrama (***Data Model diagram***). *Data Modeler* leidžia transformuoti klasių diagramą į duomenų modelio diagramą:

Tiesioginė duomenų modelio inžinerija:

objektinis modelis → duomenų modelis (DB schema)

klasių diagrama → duomenų modelio diagrama

Atvirkštinė duomenų modelio inžinerija:

duomenų modelis (DB schema) → objektinis modelis

duomenų modelio diagrama → klasių diagrama

Iš duomenų modelio diagramos galima sugeneruoti pasirinktos DBVS (Oracle, SQL server, DB2, Sybase) schemą ir iš egzistuojančios duomenų bazės galima gauti schemą:

Tiesioginė duomenų bazės inžinerija

duomenų modelio diagrama (DB schema) → duomenų bazė konkrečioje DBVS

Atvirkštinė duomenų bazės inžinerija

Konkreti DBVS → DB schema (duomenų modelio diagrama)

Rational Rose komponentai vaizduoja faktinę taikomosios programos kalbą. Duomenų bazės projektavimo atveju sukuriamas DB komponentas, kuris tarnauja tik kaip nuoroda. Norint neprisirišti prie konkrečios DBVS, galima sugeneruoti tik DDL (*Data Definition Language*) skriptą. Tam vietoje konkrečios DBVS reikia pasirinkti ANSI SQL'92 – tokiu atveju skriptą galima įvykdyti bet kurios DBVS aplinkoje.

Kiekviena RUP gyvavimo ciklo fazė dar susideda iš kelių etapų : veiklos modeliavimo, reikalavimų surinkimo, analizės ir projektavimo, realizavimo testavimo. Kiekvienas etapas tam tikroje gyvavimo ciklo fazėje užima atitinkamą sistemos kūrimo proceso dalį.

RUP funkciniams reikalavimams specifikuoti naudoja Ivar Jacobson sugalvotus panaudojimo atvejus. Taikant RUP metodą, sistemos reikalavimų specifikacija (SRS) susideda iš dviejų dalių : panaudojimo atvejų modelio ir papildomų specifikacijų.

Rational unifikuotas procesas yra sudėtingas metodas ir jo išmokimo kreivė yra pakankamai gulsčia (investuotas laikas atsiperka pakankamai negreitai). Panaudojimo atvejų modelio sudarymas yra pakankamai sudėtingas, nes pati panaudojimo atvejų idėja, iš pradžių atrodanti paprasta, reikalauja daug mokymosi norint ją sėkmingai pritaikyti. Stiprioji Rational Unified Process metodo pusė yra jo inžinerinė realizacija. Rational kompanijos pateikiamas CASE priemonių rinkinys Rational Suite leidžia automatizuoti nemažai RUP veiklų.

2.3 IDEF metodologija

IDEF (*Integration DEFinition language*) yra grupė modeliavimo metodų, naudojamų organizacijos procesams modeliuoti. IDEF sukūrė JAV oro pajėgos integruotų kompiuterių gamybai. IDEF sudaro 16 metodų, pradedant IDEF0 ir baigiant IDEF14 (įtraukiant IDEF1 ir 1X). Visi metodai aprašyti lentelėje 1. Šie metodai skirti įvairiai informacijai modeliuoti: objektiškai orientuotam projektavimui (IDEF4), simuliacijai (IDEF2), vartotojo sąsajos projektavimui (IDEF8), organizacijos modeliavimui (IDEF12) ir kiti.

Praktiškai yra naudojami tik trys šios grupės metodai: IDEF0 – funkciniam modeliavimui, IDEF3 – procesų modeliavimui ir IDEF1X – duomenų modeliavimui.

1 lentelė IDEF modeliavimo metodai

IDEF0	Funkcinis modeliavimas
IDEF1	Informacijos modeliavimas
IDEF2	Simuliacijos modeliavimas
IDEF1X	Duomenų modeliavimas
IDEF3	Procesų aprašymų nustatymas
IDEF4	Objektiškai orientuotas modelis
IDEF5	Ontologinių aprašų nustatymas
IDEF6	Loginio modelio sudarymas
IDEF7	Informacinių sistemų audito metodas
IDEF8	Vartotojo sąsajos modeliavimas
IDEF9	Scenarijais pagrįstos informacinės sistemos modelio sudarymas
IDEF10	Realizuojamos sistemos architektūros modeliavimas
IDEF11	Informacinių artefaktų modeliavimas
IDEF12	Organizacijos modeliavimas
IDEF13	Trijų schemų konvertavimo modelis
IDEF14	Tinklo modelis

2.3.1 IDEF1X metodas

IDEF1X yra semantinio duomenų modeliavimo technika. Tai metodas skirtas reliacinių duomenų bazių modeliavimui, su apibrėžta sintakse kurios pagalba galima sukurti konceptualią duomenų schemą [18]. IDEF1X technika buvo sukurta, kad atitiktų šiuos reikalavimus:

1. **Privalo palaikyti konceptualių schemų kūrimą**

   IDEF1X sintaksė palaiko semantinius konstruktus būtinus konceptualios schemos sudarymui. Pilnai išvystytas IDEF1X modelis privalo būti nuoseklus, praplečiamas ir konvertuojamas
2. **Privalo būti užrašyta suprantama kalba**

   IDEF1X turi paprastą bei nuoseklią skirtingai semantiškai išreiškiamų koncepcijų struktūrą. IDEF1X sintaksė ir semantika yra santykinai lengvai suprantama vartotojams, taipogi galinga ir išbaigta
3. **Privalo būti lengvai išmokstama**

   IDEF1X kalba yra sumodeliuota taip, kad būtų lengva naudoti tiek verslo srities profesionalams tiek sistemos analitikams, be abejo ir duomenų administratoriams ir

duomenų bazių analitikams. Taigi, ši kalba gali būti naudojama kaip efektyvus komunikavimo įrankis skirtingų sričių specialistų grupėms.

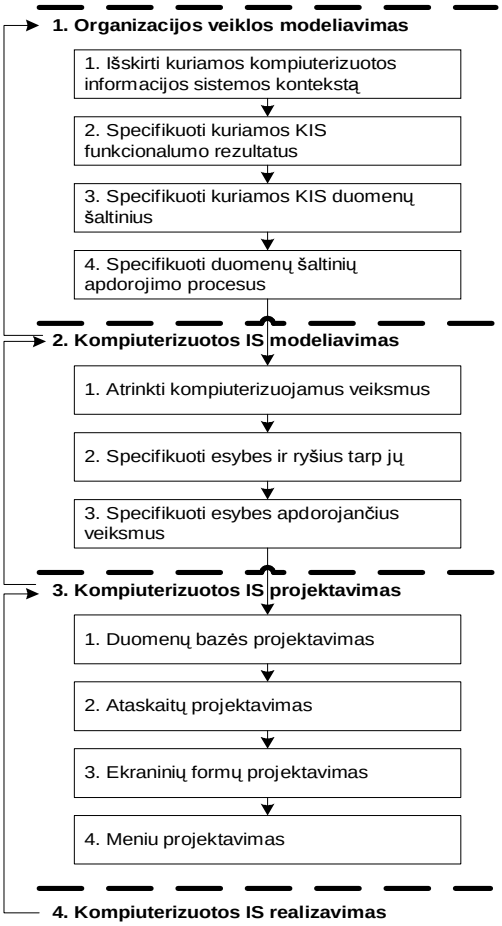
4. **Privalo būti gerai ištestuota ir įrodyta**
   IDEF1X yra paremta metais praktikos ankstesnėmis technikomis ir buvo ištestuota oro pajėgų projektuose kaip ir privačiame sektoriuje.
5. **Privalo būti automatizuojama**
   IDEF1X diagramos gali būti sugeneruojamos daugelyje grafikos paketų. Be to aktyvi medžio schema buvo sukurta oro pajėgose kuri naudoja modeliavimo metu sukurtą koncepcinę schemą programų kūrimui ir transakcijų apdorojimui ir paskirstytoms heterogeninėms aplinkoms. Komercinė programinė įranga taip pat prieinama kuri palaiko tobulinimą analizę ir konfigūracijų valdymą IDEF1X modeliuose.

IDEF1X tai vienas galingiausių duomenų modeliavimo įrankių, netgi turint omenyje, kad egzistuoja daug alternatyvių modeliavimo metodų įskaitant ER ir ENALIM. Šio metodo silpnoji pusė yra ta, kad IDEF1X naudojantis duomenų analitikas privalo turėti patirties sudarant modelius, tam kad sukurtų gerą duomenų modelį su IDEF1X. Modeliavimas tai ne intuityvus procesas ir daug kartų modeliai yra atmetami dėl blogos pradžios. Kuo paprastesnis naudojamasis metodas, tuo geriau, tačiau metodas privalo turėti reikalingą išraiškingumą.

Nors ir egzistuoja aibė IDEF metodologijos modelių, bei atskirų įrankių realizuojančių kiekvieną metodologijos modelį, bendras metodas ar CASE įrankis kompiuterizuotai informacinei sistemai sukurti neegzistuoja.

2.4 Informacijos sistemai keliamų funkcinių reikalavimų specifikavimo metodas

Reikalavimų specifikavimas, naudojantis tokiomis priemonėmis kaip *Oracle CASE[9], Rational Unified Process[17]* informacijos sistemai keliamų reikalavimų specifikacijos sudarymo eiga yra nenatūrali. Pavyzdžiui, duomenų modelio elementai į specifikaciją įtraukiami anksčiau, nei išvedamos informacijos vieneto elementai, kurie motyvuoja duomenų modelio elementų atsiradimą specifikacijoje. Informacijos sistemų katedros pasiūlytas informacijos sistemai keliamų funkcinių reikalavimų specifikavimo metodas yra artimas natūraliai reikalavimų analizės eigai [7]. Taikant šį metodą, sistemos kūrimo procesas yra suskirstomas į 4 fazes (4 pav.):



4 pav. IS kūrimo, taikant informacijos sistemai keliamų funkcinių reikalavimų specifikavimo metodą, procesas

I. Organizacijos kompiuterizuojamos veiklos modeliavimas.

II. Kompiuterizuotos IS modeliavimas.

III. Kompiuterizuotos IS projektavimas.

IV. Realizavimas.

Modeliavimo procesas, taikant nagrinėjamą metodą yra iteracinis. Į kiekvieną vystymo etapą pereinama nuosekliai, tačiau iš bet kurio etapo galima grįžti į ankstesnį etapą.

Funkcinių reikalavimų specifikacijos metodu projektuojamos informacinės sistemos duomenų bazės modelis (kuris yra saugomas ši funkcinių reikalavimų specifikacijos metodą realizuojančio CASE įrankio saugykloje), kuriuo remiantis generuojamas *SQL DDL* aprašas bei vaizduojama esybių ryšių diagrama, sukuriamas tam tikrais etapais [8]. Etapų eigoje palaipsniui surenkama duomenų modeliui (esybių ryšių diagramai) sudaryti reikalinga informacija:

1. **Išskiriamas kuriamos IS kontekstas**

   Išskiriama kompiuterizuojamos organizacijos veiklos funkcijų aibė. Ši pakopa atliekama vartotojui nurodant veiklos sritį bei pateikiant apibendrintą atliekamų funkcijų aibę toje veiklos srityje apjungiant jas į hierarchiją (panašu į *Oracle* CASE metodą).

2. **Specifikuojami duomenų šaltiniai bei jų struktūra** (čia sudaromas ir lokalus autonomiškas duomenų modelis)

   Remiantis ankstesnės pakopos rezultatais – veiklos funkcijomis, gaunami vartotojo pageidaujami rezultatai (įvairios ataskaitos, analizės rezultatai ir panašiai), kuriuos turėtų pateikti IS. Apdorojamos tik tokios funkcijos (specifikuotos ankstesnėje pakopoje), kurias atliekant suformuojami duomenys, naudojami IS funkcionalumo rezultatų atributų reikšmėms išvesti.

3. **Specifikuojami ryšiai tarp duomenų šaltinių bei jų struktūra**

   Specifikuojant duomenų šaltinius bei jų struktūrą nurodoma, iš kur bus paimti duomenys vienokio ar kitokio funkcinio rezultato formavimui, t.y. specifikuojami vartotojo reikalavimai IS įvedamai informacijai. Duomenų šaltiniai (DŠ) – tai organizacijos objektai, kuriuose saugoma informacija, susijusi su organizacijos veikla. Specifikuoti duomenų šaltiniai bus naudojami duomenų modeliui sudaryti. DŠ egzempliorius į specifikaciją įtraukiamas tik tuo atveju, jei yra bent vienas duomenų šaltinio atributas, naudojamas funkcionalumo rezultatų atributų reikšmėms formuoti. Srautas, perduodantis duomenis iš vieno duomenų šaltinio kitam duomenų šaltiniui arba rezultatui, vadinamas duomenų srautu (DS). Jo struktūrą sudarantys elementai nurodo, kokio duomenų šaltinio atributo reikšmės bus perduodamos ir koks išvedamo duomenų srauto arba kito duomenų šaltinio atributas jas gaus.

4. **Sudaromas integruotas IS duomenų modelis**

   Kiekvienam įvedamus duomenis atitinkančiam objektui – duomenų šaltiniui, sudaromi autonomiški duomenų modeliai. Duomenų modeliuose esybės – duomenų šaltiniai, o ryšiai tarp esybių – duomenų srautai. Vėliau visi atskiriems duomenų šaltiniams sudaryti duomenų modeliai integruojami, sujungiant juos į bendrą duomenų modelį – IS loginę schemą.

Informacijos sistemai keliamų funkcinių reikalavimų specifikavimo metodą galima įvertinti kaip metodą, palengvinantį analitiko darbą tiek išgaunant reikalavimus iš vartotojo, tiek atliekant šių reikalavimų specifikavimą. Nuosekli ir natūralizuota metodo etapų atlikimo technologija padidina sudaromos specifikacijos kokybę, padeda geriau patikrinti, ar specifikacija pilna. Projektuojamos IS duomenų bazės modelis sudaromas nuosekliai, specifikavus vartotojo reikalavimus, nustačius projektuojamos IS kontekstą, norimus gauti rezultatus ir kt. IS projektuotojui neprivalu remtis vien tik savo žiniomis bei patirtimi duomenų bazės projektavimo fazėje. Kol kas nėra visas funkcinių reikalavimų specifikavimo metodo (FRSM) fazes

realizuojančio įrankio, tačiau atskiros CASE įrankio modulių prototipai, kuriuos naudojant sudaromas IS duomenų bazės modelis, jau yra sukurti.

3 Duomenų modelių bei jų sudarymo principų analizė

3.1 Reliacinis duomenų bazės modelis

Septintojo dešimtmečio pradžioje E. F. Codd savo darbe aprašė duomenų saugyklos *reliacinį modelį*. Šio duomenų modelio pagrindiniai privalumai yra jais perteikiamų duomenų struktūrų paprastumas, taipogi nesudėtingi manipuliavimo duomenimis metodai ir kalbos [1]. Visi duomenys, kur jie bebūtų užrašyti, ir kompiuterio ekrane ar atspausdinti popieriuje, struktūrizuoti kaip tam tikrų reikšmių lentelė. Programuotojas loginiame lygyje taipogi įsivaizduoja duomenų pateikimo formą – duomenų lentelę. Tos pačios lentelės visos eilutės yra to paties formato, t.y. apibrėžia tokios pačios struktūros duomenų rinkinį. Lentelės duomenys charakterizuoja vieną apibendrintą realaus pasaulio objektą : konkretų tarnautoją, padalinį, detalę ir kt. Lentelės stulpeliai apibūdinantys realaus pasaulio objektą yra vadinami atributais, atributai turi priskirtus vardus, tipus ir apimtį. Kiekviena lentelės eilutė dar vadinama kortežu, sudaryta iš vieną realaus pasaulio esybę apibūdinančių atributų reikšmių aibės - *domeno*. Duomenų lentelė kitaip dar vadinama – santykiu, o lentelės atributų rinkinys sudaro santykio schemą. Santykių schemų visuma sudaro duomenų bazės schemą. Iliustruojant reliacinio modelio pavyzdžius naudojama ANSI SQL-92 standartas.

Santykis R susideda iš atributų sąrašo $<a_1,a_2, ..., a_n>$. Kiekvieną atributą identifikuoja vardas ir domenas – *dom,* atributo suskaičiuojama ir netuščia reikšmių aibė $D_i = dom(A_i)$. Sakykime, kad $D = D_1 \cup D_2 \cup D_3 \cup ... \cup D_n$. Schemos R santykis r(R) - tai aibės R atvaizdų aibėje D rinkinys $\{t_1, t_2,..., t_p\}$, kur p – santykio egzempliorių (kortežų) skaičius. Kiekvienas atvaizdas $tj \in r$, turi tenkinti griežtą apribojimą $tj(Ai) \in Di$; $1 \leq i \leq n$; $1 \leq j \leq p$. Kortežų aibė yra nerūšiuojama paprastoji aibė, neturinti pasikartojančių elementų.

Reliacinės duomenų bazės schema gali būti aprašoma DDL kalba (*data definition language*, toliau DDL) kuri yra daugelio reliacinių DBVS skaičiavimų kalbos – SQL (*Structured Query Language*) poaibis. Reliacinės duomenų bazės schemos pavyzdys, su apribojimais duomenims DDL kalba, pateikiamas 5 pav. :

```
CREATE TABLE Department (
        DepNr NUMBER(5) NOT NULL,
        DepName VARCHAR(30),
        PRIMARY KEY (DepNr));

CREATE TABLE Employee (
        Name VARCHAR(20) NOT NULL,
        Salary NUMBER(6),
        Address VARCHAR(30),
        DepNr NUMBER(5),
        PRIMARY KEY (Name),
FOREIGN KEY (DepNr) REFERENCES Department (DepNr));
```

5 pav. DDL kalba išreikšta reliacinė schema

3.1.1 Pagrindinės sąvokos reliacinėse duomenų bazėse

Reliacinės algebros ir SQL užklausų kalba realizuojamas duomenų apdorojimas bei analizė.

- Reliacinė algebra – formalus aprašymas operacijų duomenims išgauti. Matematine prasme, reliaciniai operatoriai (sąjunga, sankirta) kilę iš aibių teorijos, o reliacinės algebros reikšmių sritis susideda iš kortežų aibės [5]. Reliacinėje algebroje, visų pirma, yra apibrėžti penkių tipų operatoriai: *sąjungos, skirtumo, parinkimo, projekcijos ir produkto*. Kiti operatoriai, pvz., sankirta ar sudūrimas gali būti išreiškiami anksčiau paminėtais operatoriais. Pagrindinių operatorių apibrėžimai yra :
    - Sąjunga : $R \cup S = \{t \mid t \in \mathrm{R} \vee t \in S\}$
    - Skirtumas : $R - S = \{t \mid t \in \mathrm{R} \wedge t \notin \mathrm{S}\}$
    - Parinkimas : $\sigma_p R = \{t \mid t \in \mathrm{R} \wedge p(t)\}$
    - Projekcija: $\pi_{Ap1 \ldots Apk}, , R = \{(a_{p1},\ldots,a_{pk}) \mid (a_1,\ldots,a_{p1},\ldots,a_{pk},\ldots,a_n) \in \mathrm{R}\}$
    - Produktas $R \times S = \{s \oplus r \mid r \in \mathrm{R}, s \in \mathrm{S}\}$
    - Natūralus sudūrimas (join): $R \succ\!\!<_{Ai = Bj} S = \pi_{A1,\ldots,An,B1,\ldots,Bj-1,Bj+1,\ldots,Bm} \sigma_p$ *(Rx S)*

- SQL : SQL (*Structured query language*) kalba – sąsaja, kurią naudojant galima manipuliuoti reliacinių duomenų bazių duomenimis. SQL klaba sudaryta iš dviejų skirtingų kalbų – *data definition language* (DDL) bei *data manipulation language* (DML), kurioje specifikuojamos saugomų duomenų apdorojimo operacijos. Užklausų kalba SQL DML turi sąsają su reliacine algebra. Pagrindinis skirtumas – deklaratyvus būdas, kuriuo yra sudaromos SQL užklausos. Kadangi naudojantis reliacine algebra reikia nurodyti kaip ir kokia tvarka turi būti pritaikomi operatoriai, SQL užklausos turi panašią struktūrą :

SELECT <atributų sąrašas>
FROM <lentelių sąrašas>
WHERE <sąlyga>

Naudojant SQL kalboje sukurtą metodologiją, SQL teiginius galima įterpti į tokias kalbas, kaip *C, C++, COBOL ar DELPHI.*

- Normalizacija – tai sąvoka, naudojama trūkumams, egzistuojantiems reliacinėje duomenų bazėje panaikinti. Tipiniai reliacinės schemos trūkumai yra – pertekliniai įrašai, integralumo nebuvimas ir netrivialių atnaujinimo (*update*) ar trynimo operacijų egzistavimas. Reliacinių schemų normalizacija paprastai sumažina greitaveiką, padidindama užklausų vykdymo laiką, dėl to reliacinės schemos kartais paliekamos nenormalizuotoje būsenoje.

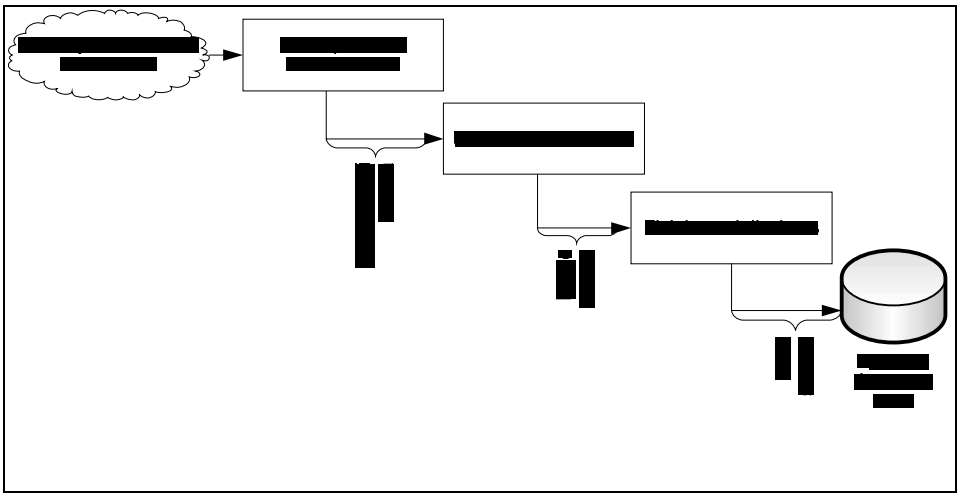
3.1.2 Reliacinės duomenų bazės modelis

Reliacinės duomenų bazės modelis yra sudaromas nuosekliai pereinant duomenų modeliavimo fazes: *koncepcinę, loginę ir fizinę*. Skirtingos duomenų modelių realizacijos apdoroja statines duomenų modelio savybes, aiškiai atskirdamos duomenų bazės modelį nuo projektuojamos programos veiklos modelio. Konceptualiame lygmenyje sukūrus ER (esybių ryšių) schemą yra sukuriamas pirminis duomenų modelis. Konceptualiosios schemos konvertavimas į loginę schemą vykdomas pusiau automatiškai. Loginis duomenų modelis atitinka reliacinį duomenų bazės modelį. Tai reiškia, kad konceptualiosios ER schemos elementai gali būti automatiškai išversti ir į reliacinės schemos elementus. Reliacinio duomenų modelio sudarymo procesas įprastai susideda iš dviejų fazių[6] : ER schemos susiejimo su reliacine schema ir normalizacijos bei denormalizacijos veiksmų.

Didžiojoje dalyje CASE įrankių skirtų duomenų bazėms modeliuoti viena dalių – grafinės ER modelio schemos sudarymas, vėliau ją konvertuojant į DDL aprašą, skirtą specifinei reliacinei DBVS. Konvertavimo proceso iš ER schemos į reliacinę schemą procesas yra apibrėžtas algoritmu. Šis algoritmas susideda iš aibės taisyklių, kurios aprašo kaip ER schemos elementą susieti su reliacinės schemos elementu. Detaliau apie duomenų modeliavimą bus paaiškinta tolimesniuose skyreliuose.

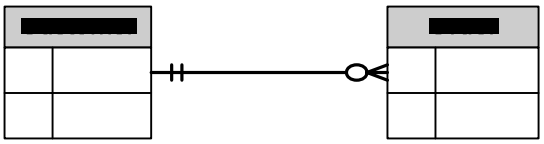
3.2 Bendra duomenų modelio koncepcija

Apibendrintai duomenų modeliavimo procesą būtų galima laikyti sąsaja tarp specifinių vartotojo reikalavimų bei duomenų bazių valdymo sistemų (DBVS). Duomenų bazių schemų sudarymo procesas susideda iš keleto viena po kitos einančių fazių. Kiekvienoje jų yra sudaromi skirtingi duomenų modeliai. Yra skiriamos trys modeliavimo fazės: *konceptualioji, loginė* bei *fizinė* (6 pav.).



6 pav. Duomenų bazės projektavimo fazės

Konceptualiosios modeliavimo fazės paskirtis - formaliai perteikti reikalavimus, gautus iš užsakovo reikalavimų inžinerijos metu. Atlikus konceptualųjį duomenų modeliavimą sukuriama aukšto abstrakcijos lygio schema, kurioje nenurodomos duomenų bazės detalės (atributų ilgis, raktai ir kt.). Konceptualaus modeliavimo pagrindas – sukurta esybių-ryšių diagrama (ERD). Šią sąvoką 1976 m. Peter Chen pristatė savo darbe [2]. Esybių ryšių diagrama, preliminariai atvaizduojama duomenų bazės struktūra, bei nurodoma, kokia informacija joje bus saugoma. ERD privalo būti technologiškai nepriklausoma, t. y. privalo būti sudaryta taip, kad ją būtų galima realizuoti bet kurioje reliacinėje duomenų bazių valdymo sistemoje.



Loginio modeliavimo fazės metu, konceptualioji schema yra konvertuojama į loginę schemą. Loginėje schemoje nurodomi loginiai esybių tipai, esybių atributai bei ryšiai tarp esybių. Reikėtų pastebėti, kad kai kuriuose duomenų bazių modeliavimo metoduose ar įrankiuose,

konceptualaus modeliavimo fazė yra praleidžiama ir iš karto yra kuriama loginė duomenų schema [3]. Loginę schemą gali apdoroti daugelis DBVS, kadangi pastaroji gali būti išreiškiama SQL *data definition language* priemonėmis. Sukūrus loginę schemą ši gali būti konvertuojama į fizinę schemą.

Fizinio modeliavimo fazės metu, loginis duomenų modelis yra pritaikomas specifinei DBVS. Fiziniame duomenų modelyje yra įvedami apribojimai duomenims, pvz., lentelių pavadinimų ar stulpelių pavadinimų ilgių apribojimai.

Loginis duomenų modelis yra pakankamai išsamus, kad jį naudojant būtų galima sudaryti duomenų bazės SQL DDL schemos aprašą.

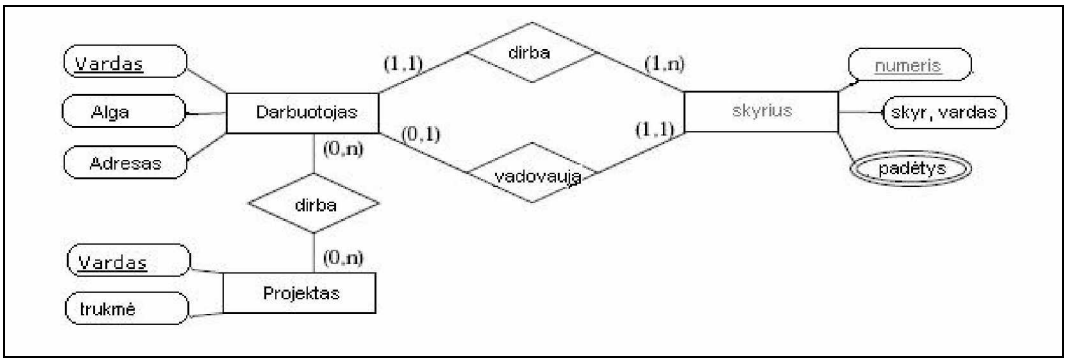
3.2.1 Esybių ryšių modelis

Esybių ryšių modelį 1976 metais pristatė *Peter Chen*[2]. Naudojant šį modelį realaus pasaulio duomenys aprašomi diagramose kaip esybės, ryšiai tarp esybių ir esybių atributai. Esybės tipas arba tiesiog esybė – tai konkrečių arba abstrakčių realaus pasaulio daiktų ar objektų aibė, apie kurią turi būti surinkta ir saugoma informacija. Esybė gali būti naudojama ir duomenų konstrukcijoms (įrašų tipams, lentelėms, segmentams ir kt.) aprašyti. Konkreti interpretacija priklauso nuo schemos abstrakcijos lygio. Esybės tipą atstovauja jos vardas, taigi realaus pasaulio objektų grupė turi būti vaizduojama tik viena esybe.

Esybės turi atributus – tam tikrą esybę apibūdinančias savybes ir vidinius apribojimus. Atributas – tai esybės detalė, kuri išreiškia kokybines arba kiekybines esybės savybes. Atributai turi savo reikšmes, kuriomis esybės egzemplioriai identifikuojami ir klasifikuojami. Pavyzdžiui darbuotojų esybę apibūdina jų atributai: darbuotojo vardas, alga bei adresas. Esybėms identifikuoti gali būti panaudotas vienas *pirminis raktas* ir vienas ar keli *antriniai raktai*. Esybių

identifikavimui panaudotas raktas gali susidėti iš vieno ar kelių atributų – sudėtinių ir/arba paprastųjų. Pavyzdinis pirminis raktas galėtų būti, pavyzdžiui, darbuotojo vardas. Esybės yra klasifikuojamos į priklausomas ir nepriklausomas. *Nepriklausomos esybės* yra tos, kurių unikalumas apibrėžiamas naudojant kitą esybę. *Priklausomos esybės* yra tos, kurių unikalumas yra apibrėžiamas kitų esybių pagalba. Taipogi yra skiriami specialūs esybių tipai : *asociatyvios esybės* – esybės naudojamos surišti dvi ar daugiau esybių (pavyzdžiui norint eliminuoti N:M ryšį). *Esybės – potipiai* – naudojamos generalizacijos hierarchijose.

Ryšio tipas arba tiesiog *ryšys* – tai prasminis santykis tarp esybių. Ryšio tipą sudaro esybių tipai su rolėmis, o rolės sieja ryšio tipą su esybe. Jeigu ryšyje dalyvauja daugiau nei du esybių tipai, tai toks ryšys vadinamas daugianariu. Ryšio tipas, kuris sieja dvi esybes, vadinamas dvinariu. Ryšio tipas, siejantis tą pačią esybę, vadinamas cikliniu arba rekursyviu, ir jis paprastai naudojamas esybių egzempliorių hierarchijoms vaizduoti. Esybių ryšių diagrama, naudojanti originalią *Chen* pasiūlytą notaciją, pavaizduota (7 pav.).



7 pav. ER schema naudojant originalią ER notaciją

ER modelio išplėtimas yra išplėstas ER (EER) modelis, kuris yra papildytas semantinėmis modeliavimo idėjomis, įvedant generalizacijos/specializacijos ryšius.

3.2.2 Esybių ryšių modelių notacijos

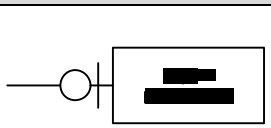
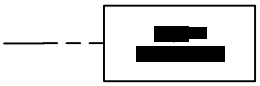
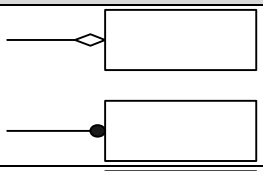
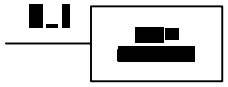
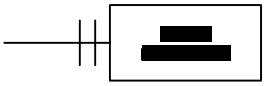
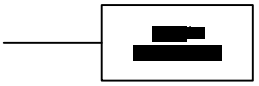
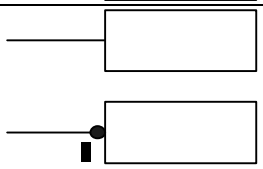
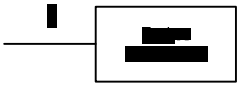
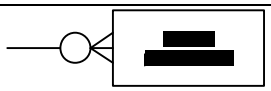
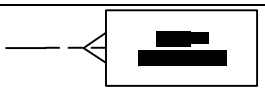
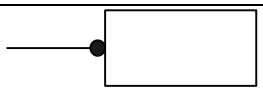
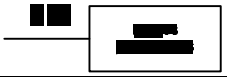
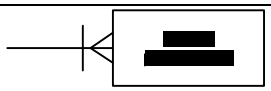
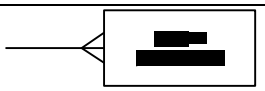
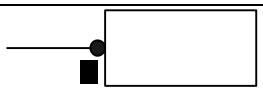
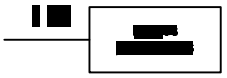
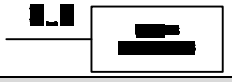
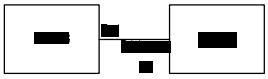
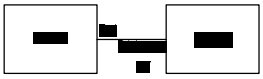
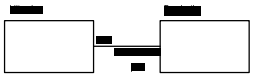
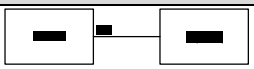
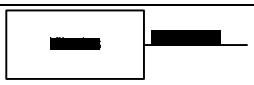
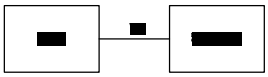
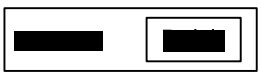
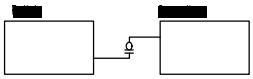
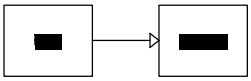
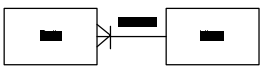
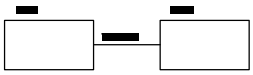
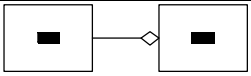
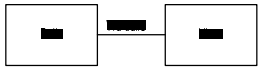
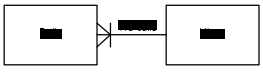
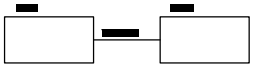
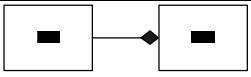
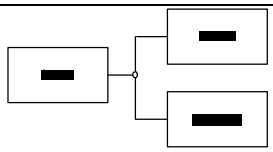
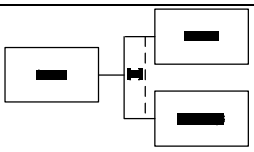
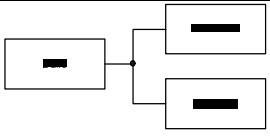
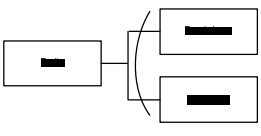
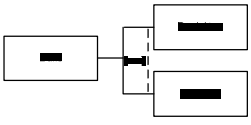
Nepaisant to, kokie grafiniai simboliai naudojami duomenų ar objektų modeliavimui, jų paskirtis vienoda [11]: aprašyti realaus pasaulio objektus bei ryšius tarp tų objektų, apie kuriuos organizacijos pageidauja rinkti duomenis. Dėl šios priežasties, didžioji dalis duomenų modelius vaizduojančių ER diagramų notacijų, gali būti konvertuojamos viena į kitą. Pagrindiniai skirtumai tarp grafinių duomenų modelį atvaizduojančių notacijų yra vaizduojamų duomenų raiškumas,

išsamumą. Kai kurios grafinės duomenų modelių notacijos neturi simbolių nurodyti visoms realioms situacijoms.

Kaip minėta P.Chen pristatyta esybių ryšių modeliavimo koncepcija buvo revoliucinė duomenų vaizdavime. Tačiau tai buvo tik pirmasis bandymas diagrama perteikti realaus pasaulio duomenis, taigi daugelis analitikų tobulino šį modelį. Dabar originalioji ER notacija didžiąja dalimi naudojama mokymosi procese, universitetuose ir kolegijose.

2 lentelėje pateiktas populiariausių ER duomenų modeliavimo grafinių notacijų apžvalga. Lentelės stulpeliuose nurodoma, kaip kiekviena notacija atvaizduoja eilutėse nurodytus kriterijus keliamus notacijai.

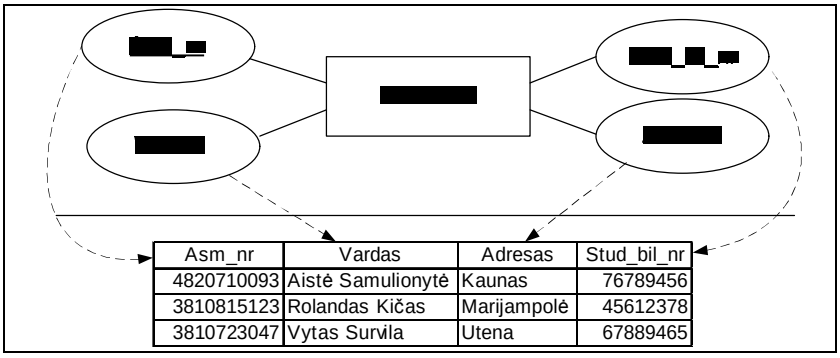
2 lentelė Notacijų palyginimas

Notacija / **Požymis**	Informacijos inžinerija	Barker	IDEF1X	UML
Kardinalumai				
Nulis arba vienas				
Tik vienas				
Nulis arba daug				
Vienas arba daug				
Tam tikras rėžis	Nėra atitikmens	Nėra atitikmens	Nėra atitikmens	
Atributai				
Atributo vardas	Atributo vardas	Atributo vardas : **tipas**	Atributo vardas : **tipas**	AtributoVardas : **tipas**
Pirminis raktas	Paryškintas atributas* / (nėra)	#Atributo vardas	Atributas-vardas	AtributoVardas<<PK>>
Išorinis raktas	Nėra	Nėra	Atributas-vardas(FK)	AtributoVardas<<FK>>
Ryšiai (asociacijos)				
Užrašai				
Esybių rolės	Nėra	Nėra	Nėra	
Potipiai				
Agregavimas				
Kompozicija				
„arba“(OR) apribojimas		Nėra	Nėra	
„išskirtinio arba“(XOR) apribojimas			Nėra	

3.3 Esybių ryšių modelio konvertavimas į reliacinę schemą

Esybių ryšių modelis – tai tik norimos realizuoti duomenų bazės loginis vaizdas. Norint iš esybių ryšių modelio sukurti fizinį SQL DDL aprašą, būtina konvertuoti esybių ryšių modelį reliacinę schemą, kaip pavaizduota 8 pav.. Konvertuojant esybių ryšių modelį, kiekvienas esybės tipas yra konvertuojamas į santykį, esybės atributai tampa santykio stulpeliais, o esybių egzemplioriai – santykio eilutėmis. Santykiai gali būti išreiškiami tekstu :

lentelės pavadinimas (<u>pirminis raktas</u>, atributas 1, atributas 2,... , išorinis raktas)



8 pav. Esybių ryšių modelio konvertavimas į reliacinį modelį (konceptualus atvaizdavimas)

Išoriniai raktai – atributai (ar grupė atributų), kurie yra saugomi kaip pirminis raktas kitame santykyje (lentelėje). Kiekvienas išorinis raktas nurodo sąryšį tarp dviejų esybių tipų. Šie raktai yra įrašomi į duomenų bazės schemą, konvertuojant ER diagramą į reliacinę schemą. Kiekvienas santykis, gali turėti daugiau nei vieną išorinį raktą.

Kaip minėta anksčiau, ER diagramos transformavimas į reliacinę schemą, vyksta pagal apibrėžtą algoritmą [6] susidedantį iš aibės taisyklių. Transformavimo algoritmą sudaro septyni žingsniai. Algoritmo apraše bus apibrėžtos taisyklės kurios yra pakartotinai naudojamos skirtinguose algoritmo žingsniuose (taisyklės numeruojamos su prefiksu **T**) :

1. **Nepriklausomų ER diagramos esybių konvertavimas**

   T1 – reikia sukurti naują lentelę (dėl paprastumo santykį toliau vadinsime lentele) kiekvienai nepriklausomai esybei, pirminiu lentelės raktu pasirenkant apibrėžtą esybės raktą ER diagramoje. Jeigu ER diagramoje nurodytas ne vienas, o keli kandidatiniai raktai, vienas jų išrenkamas kaip pirminis. Jeigu diagramoje pateiktas esybės atributas yra daugianaris (sudėtinis), pirminis lentelės raktas sudaromas iš visų atominių daugianario atributo elementų

(pvz. jeigu asmenį identifikuoja *vardas, lytis, asmens kodas* lentelės pirminis raktas bus sudarytas iš šių trijų atributų).

Kiekvienai nepriklausomai esybei sukūrus lenteles ir nustačius lentelių pirminius raktus, konvertuojami nepriklausomos esybės atributai. Atominių, sudėtinių bei daugiareikšmių atributų konvertavimui atlikti nustatytos skirtingos taisyklės. Reliacinėje duomenų bazėje, visi lentelės atributai privalo būti atominiai(vienetiniai). Taigi, jeigu esybių ryšių diagramoje aprašyti ne atominiai atributai, juos reikia konvertuoti į atominius, kad būtų įmanoma sukurti tų atributų atitikmenis reliacinėje duomenų bazėje.

T1a konvertuojant esybę į lentelę, lentelėje reikia sukurti stulpelius atitinkančius atominius esybės atributus.

T1b esybėms turinčioms sudėtinių atributų, reikia suformuoti lentelės stulpelius kiekvienam atominiam sudėtinio atributo elementui.

T1c esybėms turinčioms daugiareikšmių atributų, reikia suformuoti atskirą lentelę. Naujosios lentelės pirminiu raktu nustatomas daugiareikšmis atributas bei tėvinės lentelės pirminis raktas. Daugiareikšmis atributas iš tėvinės lentelės pašalinamas.

2. **Priklausomų ER diagramos esybių konvertavimas**

T2 – reikia sukurti naują lentelę kiekvienai priklausomai esybei. Kaip ir konvertuojant nepriklausomas esybes, į lentelę reikia įtraukti konvertuojamosios esybės atributus remiantis **T1a**, **T1b** ir **T1c** taisyklėmis. Tam, kad susieti priklausomą esybę ir jos tėvinę (nepriklausomą) esybę, į lentelę reikia įtraukti tėvinės esybės pirminio rakto atributą(-us) kaip išorinį raktą priklausomajai esybei. Pirminis priklausomosios esybės raktas bus nurodytasis pirminis apdorojamos esybės raktas kartu su tėvinės esybės pirminiu raktu. Jeigu priklausomosios esybės, yra nurodytos kaip tėvinės esybės kitoms priklausomoms esybėms, tada pirmiausia konvertuojama ta priklausomoji esybė, kuri tiesiogiai siejasi su nepriklausoma tėvine esybe, o vėliau tik kitos esybės priklausomos nuo apdorojamos. Taigi pirminis kiekvienos tėvinės esybės raktas privalo būti nustatytas prieš konvertuojant priklausomąją esybę.

3. **Dvinarių M:N ryšių konvertavimas**

T3a – kiekvienam M:N sąryšiui tarp esybių ER diagramoje, reikia sukurti naują lentelę, kurios atributais būtų kiekvienos iš susietų ryšiu esybių pirminiai raktai. Naujosios lentelės pirminis raktas bus abiejų santykyje dalyvaujančių esybių pirminių raktų sąjunga. Be to, reikia sukurti atributus kuriuos, be raktų, dar gali turėti naujasis santykis.

4. **Dvinarių 1:1 ryšių konvertavimas – pirminio/išorinio rakto metodas**

Egzistuoja du būdai eliminuoti 1:1 ryšį tarp esybių. Galima sukurti naują lentelę kaip nurodyta **T3a** taisyklėje; arba sąryšis gali būti konvertuojamas pagal pirminio/išorinio rakto taisykles (PK/FK) :

T3b – (A:B::1:1) įtraukti pirminį esybės A raktą į esybę B kaip išorinį raktą

T3b_1 – dvinariams 1:1 ryšiams tarp esybių, jeigu viena ryšio pusių pilnai dalyvauja ryšyje t.y. privaloma, o kita neprivaloma, tuo atveju reikia išsaugoti pirminį raktą tos esybės, kuri ryšyje yra neprivaloma ir išsaugoti tą raktą kaip išorinį pilnai ryšyje dalyvaujančioje esybėje. Įtraukti bet kuriuos atributus toje lentelėje kuri gauna išorinį raktą. Ši taisyklė leis išorinį raktą saugančiuose atributuose išvengti *null* reikšmių

T3b_2 – dvinariams 1:1 ryšiams tarp esybių, jeigu abiejuose ryšio galuose esančios esybės ryšyje yra neprivalomos, egzistuoja trys skirtingi ryšių konvertavimo į reliacinę duomenų bazę būdai :

T3b_2a – galima pasirinkti bet kurią lentelę, kitos lentelės rakto saugojimui (kai kuriuose lentelės eilutėse, atributų reikšmės bus *null*)

T3b_2b – galima sukurti naują lentelę kurioje būtų saugomi abiejų esybių raktai (kaip **T3a**)

T3b_2c – galima sukurti naują lentelę saugančią tik dviejų ryšyje dalyvaujančių lentelių raktus. Tuo atveju ryšius galima konvertuoti taip kaip ir M:N atveju; ir jeigu susidarytų *null* reikšmės, jos nebūtų įtraukiamos į rišančiąją lentelę

T3b_3 – dvinariams 1:1 ryšiams tarp esybių, jeigu abi esybės yra privalomos ryšyje, galima naudoti tokią ryšio semantiką, kad būtų įmanoma nustatyti kurioje lentelėje, kitos lentelės raktas bus nurodomas kaip išorinis. Nekorektiška į abi lenteles įtraukti išorinius raktus, nes tuo atveju duomenų bazėje gali susidaryti pertekliniai duomenys. Į lentelę saugančią išorinį raktą, reikia įtraukti visus ryšio atributus. Visgi šiai situacijai spręsti labiau tinka T3a taisyklė.

5. Dvinarių 1:N ryšių konvertavimas

T3c – nors didžioji 1:N ryšių dalis yra konvertuojama *PK/FK* metodo pagalba, galima sukurti atskirą lentelę kaip ir taisyklėje T3a, kad būtų galima naudoti *PK/FK* metodą reikia patikrinti su kokiu apribojimu ryšyje dalyvauja N(daug) ryšio pusė :

T3c_1 – jeigu N – ryšio pusėje esanti esybė yra privaloma, reikia įtraukti esybės kuri yra ryšio pusėje „1“ raktą, kaip išorinį raktą esybėje „N“ ryšio pusėje. Jeigu „N“ ryšio pusė yra priklausoma, t.y. be pirminio rakto, raktas iš ryšio pusės „1“ bus sukuriamas

„N“ ryšio pusės lentelėje kaip pirminis raktas. Įtraukti bet kokius atributus kurie buvo ryšyje į lentelę į kurią pervedamas išorinis raktas („N“ pusę)

T3c_2 – jeigu „N“ ryšio pusė yra neprivaloma, 1:N ryšys apdorojamas kaip ir M:N ryšys sukuriant atskirą lentelę ryšiui, kad būtų išvengta *null* reikšmių. Naujosios lentelės raktas susideda iš apjungtų susijusių esybių raktų. Taipogi į lentelę reikia įtraukti ir visus ryšio atributus.

Neprivalomas dalyvavimas ryšyje yra problema, kurios išdava - *null* reikšmės duomenų bazės lentelių eilutėse. Jeigu raktas pervedamas iš „1“ ryšio pusės į „N“ ryšio pusę ir jeigu esybės dalyvavimas ryšyje yra neprivalomas (ne kiekvienas kortežas „N“ ryšio pusėje turi atitikmenį ryšio pusėje „1“), tada duomenų bazėje susidaro *null* reikšmės kai į ją vedami duomenys. Todėl yra geriau sukurti atskirą lentelę 1:N ryšiui saugoti ir išvengti *null* reikšmių.

6. **Rekursinių ryšių konvertavimas**

T4 – buvo sukurti du konvertavimo tipai :

T4a – 1:N rekursiniams ryšiams, jeigu yra rekursinis ryšys į tą pačią lentelę, lentelėje reikia sukurti pirminį lentelės atributą atitinkantį atributą kitu pavadinimu .

T4b – M:N rekursiniams ryšiams sukurti atskirą lentelę ryšiui (kaip taisyklėje **T3a**)

7. **Daugianarių ryšių konvertavimas**

T5 – kiekvienam daugianariui (*n-ary*) ryšiui, sukurti naują lentelė. Į lentelę įtraukti visus ryšio atributus. Tada įtraukti visus apjungtus esybių raktus kaip išorinius raktus ir pirminiu raktu nustatyti visų išorinių raktų sąjungą.

Aukščiau pateiktas apibendrintas duomenų konvertavimo algoritmas yra taikomas tuo atveju, kai duomenų analizės bei projektavimo fazės yra atskirai išskirtas procesas informacinių sistemų kūrime.

3.4 Metodų ir įrankių savybių įvertinimas

Ankstesniuose skyriuose buvo apžvelgti naudojami metodai duomenų modelių sudarymui bei konvertavimui, taipogi apžvelgtos duomenų modelių vaizdavimo notacijos. Išanalizuoti Oracle CASE, RUP ir IDEF metodai, automatizuotam sistemų projektavimui ir kūrimui. 3 lentelėje pateiktas Oracle CASE, RUP, funkcinių reikalavimų specifikavimo metodo ir IDEF palyginimas, pagal pasirinktus kriterijus.

3 lentelė **Oracle CASE, RUP, FRSM ir IDEF palyginimas**

Metodai / **Kriterijai**	Oracle CASE	RUP	FRSM	IDEF
Siūloma programinė įranga	+	+	-/+	-/+
Siūlomų grafinių modelių (notacijų) suprantamumas, raiškumas	+	+	+	+
Duomenų modeliavimo fazei pateikiama pradinė informacija	-	-/+	+	-
Galimybė generuoti duomenų bazės SQL DDL aprašus	+	+	-/+	+

Oracle CASE bei RUP metodams, yra siūlomi galingi įrankiai (CASE designer, Rational Rose). IDEF metodologijos atskiriems metodams taipogi siūlomi įrankiai realizuojantys vieną ar kitą metodą, tačiau apjungiančio visus IDEF metodus įrankio nėra sukurta. FRSM metodą realizuojanti programinė įranga (CASE įrankis) atskirais moduliais yra kuriama Kauno technologijos universiteto informacinių sistemų katedroje.

Oracle CASE ir IDEF metoduose, esybių ryšių modelis, kurio pagrindu yra kuriama projektuojamos sistemos duomenų bazė, yra formuojamas remiantis duomenų analitiko žiniomis apie projektuojamos sistemos dalykinę sritį bei patirtį projektuojant duomenų bazes. RUP duomenų modelis yra sudaromas klasių diagramos (sistemos loginio vaizdo) pagrindu, tačiau klasių diagrama yra suprojektuojama analitiko. Tokiu būdu gali būti sukurti, pilnai projektuojamos IS funkcionalumo nepadengiantys duomenų modeliai, o vėliau ir reliacinių duomenų bazių schemos. FRSM esybių ryšių modelis yra sudaromas ir saugomas metamodelyje, nuosekliai įvykdžius tam tikras metodo fazes.

Oracle CASE, RUP ir IDEF metodus realizuojančioje programinėje įrangoje yra galimybė generuoti duomenų bazės SQL DDL aprašus. FRSM suprojektuotos duomenų bazės modelis saugomas CASE įrankio saugykloje ir kol kas nėra įrankio galinčio nuskaityti modelį, bei iš jo sugeneruoti SQL DDL aprašo.

3.5 Analizės išvados

Išanalizuoti automatizuoto IS projektavimo metodai bei įrankiai. Įvertinti šių metodų privalumai ir trūkumai projektuojant IS duomenų bazes. Aprašyto informacijos sistemai keliamų funkcinių reikalavimų specifikavimo metodo kontekste bus sukurtas loginio duomenų modelio vaizdavimo bei konvertavimo į reliacinę duomenų schemą modelis.

Ištirta skirtingų duomenų modeliavimo fazių (konceptualiosios, loginės ir fizinės) reikšmė sudarant duomenų modelius bei nustatyta, kad loginis duomenų modelis yra pakankamai išsamus fizinės duomenų bazės schemos (SQL DDL aprašo) generavimui. Nustatyta kokių loginės schemos sudarymo klaidų bus siekiama išvengti realizuojant loginio duomenų modelio vaizdavimo ir redagavimo modulį CASE įrankyje.

Apžvelgtas esybių ryšių modelis bei palygintos populiarios šio modelio notacijos (*IDEF1X, Barker, informacijos inžinerijos)* iš kurių išrinkta išraiškingiausia informacijos inžinerijos notacija. Esybių ryšių modelis realizuojamame CASE įrankio modulyje bus pateikiamas šia notacija.

Apžvelgtas loginio duomenų modelio konvertavimo į reliacinę schemą algoritmas, kuriuo iš dalies bus remiamasi kuriant CASE įrankio modulį realizuojantį reliacinės duomenų schemos SQL DDL aprašo generavimą iš loginio duomenų modelio suformuoto funkcinių reikalavimų specifikacijos pagrindu.

4 Duomenų modelio sudarymo etapas funkcinių reikalavimų specifikacijos metodo pagrindu

4.1 Modelio sudarymo koncepcija ir pradinės sąlygos

Išanalizavus duomenų modelių projektavimo, atvaizdavimo bei konvertavimo metodus ir įrankius buvo pasiūlyta metodika, informacijos srautų specifikacijos pagrindu, duomenų modelio ER diagramos informacijos inžinerijos notacijoje sudarymui, bei reliacinės duomenų bazės SQL DDL aprašo generavimui [17]. Šis funkcinių reikalavimų specifikavimo metodo (FRSM) etapas tai pirmas žingsnis į modeliuojamos informacinės sistemos fizinę realizaciją. FRSM pagrindinė idėja: priklausomai nuo IS aplinkos, pirmiausia reikia nustatyti, ką kuriamoji sistema turi pateikti kaip išeigą, o po to nustatyti, ką paduoti kaip įeigą ir kaip tą įeigą transformuoti į išeigą. Taikant šį metodą, sistemos kūrimo procesas yra suskirstomas į keturias fazes, kurias detalizuoja tam tikri etapai:

I. Organizacijos kompiuterizuojamos veiklos modeliavimas.
    1. Išskirti kuriamos IS kontekstą;
    2. Specifikuoti kuriamos IS funkcionalumo rezultatus;
    3. Specifikuoti kuriamos IS duomenų šaltinius;
    4. Specifikuoti duomenų šaltinių apdorojimo procesus;
II. Kompiuterizuotos IS modeliavimas.
    1. Atrinkti kompiuterizuojamus veiksmus;
    2. Specifikuoti esybes ir ryšius tarp jų;
    3. Specifikuoti esybes apdorojančius veiksmus;
III. Kompiuterizuotos IS projektavimas.
    1. Duomenų bazės projektavimas;
    2. Ataskaitų projektavimas;
    3. Formų projektavimas;
    4. Meniu projektavimas
IV. Realizavimas.

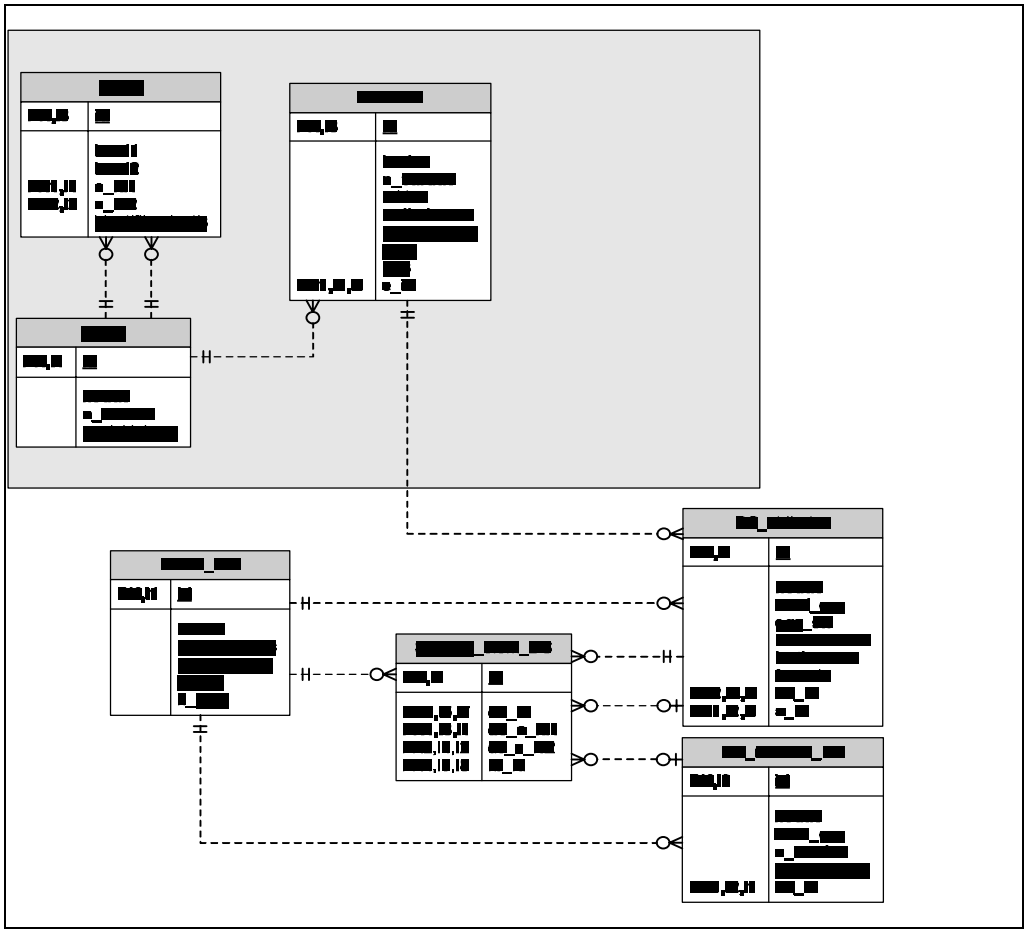
Projektuojamos IS loginio duomenų modelio diagramos sudarymo, bei šios diagramos konvertavimo į SQL DDL aprašą funkcionalumo specifikavimo modelis apima du šio metodo etapus: duomenų bazės projektavimo etapą bei informacinės sistemos realizavimo (duomenų bazės

fizinio aprašo generavimo) etapą. 9 pav. pateikiama principinė realizuojamo modelio ir ankstesnių funkcinių reikalavimų specifikavimo metodo fazių sąsajos schema.

Pirminis FRSM etapas yra konteksto specifikavimas, kurio metu yra specifikuojamos veiklos funkcijos. Vėliau, remiantis apibrėžtomis funkcijomis, gaunami vartotojo pageidaujami rezultatai, kurių analizės metu nustatomi duomenų šaltiniai (organizacijos objektai, kuriuose saugoma informacija susijusi su organizacijos veikla), duomenų šaltinių atributai bei srautai tarp duomenų šaltinių. Kiekvienam duomenų šaltiniui sudaromi autonomiški duomenų modeliai, kuriuos suintegravus gaunamas projektuojamos sistemos duomenų bazės loginės schemos modelis. Loginis duomenų modelis saugomas FRSM realizuojančio CASE įrankio duomenų saugykloje (metabazėje). Sekantys žingsniai patenka į realizacijos fazę. Suformuotas duomenų modelis yra analizuojamas, atvaizduojamas ER diagrama bei pateikiamas duomenų analitikui apdoroti. Vėliau duomenų modelis konvertuojamas į SQL DDL aprašą, kurio pagrindu gali būti sukurta duomenų bazė reliacinėje duomenų bazių valdymo sistemoje, palaikančioje SQL – 92 standartą.

4.2 Modeliavimui naudojamų duomenų sudėtis

Žemiau pateikiamas metamodelis, kuriame saugoma informacija apie projektuojamos IS duomenų saugyklos loginį duomenų modelį, fragmentas. Tai FRSM realizuojančio CASE įrankio duomenų saugyklos dalis. Svarbiausia informacija apie projektuojamos IS duomenų bazės loginį modelį yra saugoma trijose CASE įrankio duomenų saugyklos lentelėse : *Ryšys, Esybė* ir *Atributas (diagramoje išryškintos pilku fonu).* Norint pateikti pilną duomenų modelį, t. y. be nepageidaujamų informacijos duomenų modelyje trūkumų, informacija nuskaityta iš *Esybė, Ryšys ir Atributas* lentelių yra papildoma duomenimis iš lentelių aprašančių duomenų šaltinius bei duomenų šaltinių atributus.



9 pav. CASE duomenų saugykloje saugomo projektuojamos IS duomenų modelio, metamodelio duomenų struktūra

Žemiau pateikiamas detalus metamodelio fragmento, kurio pagrindu yra atvaizduojamas ER modelis informacijos inžinerijos notacijoje aprašymas.

Esybė. Šioje lentelėje yra saugoma informacija apie FRSM etapuose išskirtas esybės – kurios bus vaizduojamos projektuojamos IS duomenų modelio ER diagramoje. Lentelėje saugoma informacija:

- *Kodas* - unikalus esybę apibudinantis kodas, kurį sudaro sistemos vartotojas (analitikas/projektuotojas);
- *Vardas* - esybės vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo;
- *Paaiškinimas* – detalizuota informacija apie esybę.

Atributas. Šioje lentelėje yra saugoma informacija apie FRSM etapuose išskirtų esybių atributus. Atributas priklauso tik vienai esybei. Lentelėje saugoma informacija:

- *Kodas* – unikalus atributą apibudinantis kodas, kurį sudaro sistemos vartotojas (analitikas/projektuotojas);

- *Vardas* – atributo vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo;
- *Rakto požymis* – požymis kuris nurodo, ar atributas priklausantis esybei yra esybės pirminis raktas;
- *Atributo unikalumas* – požymis rodantis ar atributo reikšmė turi būti unikali. TRUE – reikšmė turi būti unikali, o FALSE - reikšmės unikalumas nėra būtinas;
- *Tipas* – atributo duomenų tipas. Atributo reikšmė saugoma kaip tekstinė informacija;
- *Ilgis* – atributo ilgis nurodytam duomenų tipui.

Informacijos dviprasmiškumui panaikinti, kai atributo tipas išreikštas tekstu gali būti interpretuotas neteisingai, duomenų metamodelis praplėstas atributo duomenų tipo klasifikatoriumi.
Duomenų tipo klasifikatoriuje apibrėžti duomenų tipai įvairioms, fizinio duomenų modeliavimo metu pasirenkamoms duomenų bazių valdymo sistemoms, priklausomai nuo to kaip duomenų tipas yra įvardinamas skirtingose DBVS.

Ryšys. Šioje duomenų saugyklos lentelėje yra saugoma informacija apie ryšius tarp FRSM etapuose išskirtų esybių. Lentelėje be informacijos aprašančios, kurias esybes sieja ryšys, saugoma informacija :

- *Kardinalumas* – informacija apie ryšio galus. Kardinalumo aprašymo struktūra: <būtinumas>,<kardinalumas>. Būtinumas gali įgyti šias reikšmes: 0-nebūtinas, 1- būtinas. Kardinalumas gali įgyti šias reikšmes: 1 – vienas, x – daug.
- *Identifikuojantis* – ar ryšys siejantis dvi esybes yra identifikuojantis, t. y. ar pirmosios (tėvinės) ryšyje nurodytos esybės raktiniai atributai bus apibrėžiami kaip išorinis antrosios(vaiko) esybės raktas, o kartu ir kaip pirminis antrosios esybės raktas.

Kadangi nurodytame duomenų metamodelyje, nėra informacijos apie veiksmus (*referential action*) kurie turi būti atliekami vaiko esybėje, atnaujinus ar ištrynus tėvo esybės egzempliorių. Lentelėje įvesti papildomi laukai: ***atnauj_id***, ir ***istryn_id***, o projektuojamos IS metamodelis papildytas ryšio veiksmų klasifikatoriumi (lentele ***Ryšio veiksmas***), kuriame saugoma aibė veiksmų kuriuos reikia atlikti vaiko esybėje modifikavus tėvo esybę (*No Action, Cascade, Set NULL, Set default, Do not enforce)*

DS_atributas. Šioje duomenų saugyklos lentelėje saugoma informacija apie organizacijos objektų (duomenų šaltinių) saugančių duomenis, reikalingus funkcijoms įvykdyti, atributus. Remiantis šioje lentelėje saugoma informacija detalizuojami esybių atributai, aprašyti metamodelio lentelėje

*Atributa*i. Šios lentelės pagalba, nustatoma informacija apie atributo būtinumą projektuojamos IS duomenų bazės lentelėje.

Salygos_elem_DS. Lentelėje saugoma informacija apie duomenų šaltinio sudarymo sąlygos elementus. Šioje lentelėje saugoma informacija apie duomenų šaltinio atributus, kurie įtakoja ar yra įtakojami kitų duomenų šaltinio atributų. Be to, šioje lentelėje, saugoma sąsaja su duomenų šaltinio atributo ribinėmis reikšmėmis (*reikšmių domenu*) kurias gali įgyti atributas tenkinant apibrėžtas sąlygas.

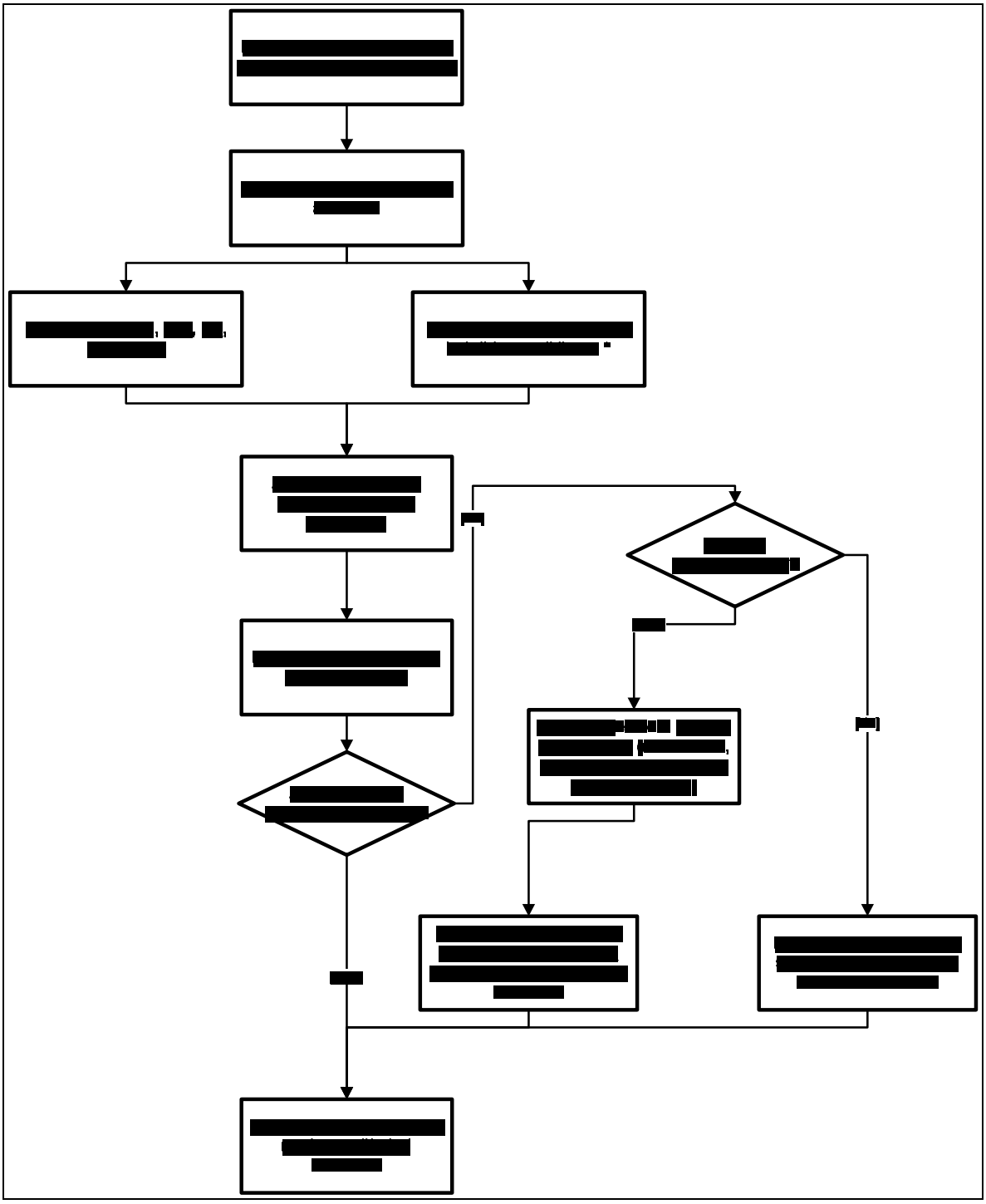
Rib_reiksme_DS. Lentelėje saugoma duomenų šaltinyje esančių atributų egzempliorių ribinių reikšmių informacija. Šioje lentelėje saugomos informacijos pagrindu ateityje gali būti suformuojamas patikrinimo apribojimas (*CHECK constraint*) projektuojamos esybės atributui. Tačiau, tam reikia, kad sąlygos elementas būtų bent iš dalies formalizuotas.

4.3 CASE įrankio saugykloje aprašyto duomenų modelio analizės algoritmas

Informacija apie projektuojamos IS duomenų sudėtį iki šios funkcinių reikalavimų specifikavimo metodo fazės buvo apdorojama tik šį metodą realizuojančiame CASE įrankyje. Kadangi FRSM neužtikrina visapusiškai teisingo projektuojamos IS duomenų modelio, pravartu, kad ankstesnėse FRSM fazėse automatiškai sudarytą duomenų modelį peržiūrėtų duomenų analitikas. Duomenų analitikas gali įvesti reikiamus pakeitimus į duomenų modelį (pavyzdžiui modifikuoti atributus, jų pavadinimus ar ryšių veiksmus) prieš generuojant duomenų bazės SQL DDL aprašą. Žemiau pateikiama apibendrinta veiksmų seka, kuri apibrėžia metamodelio interpretavimą ir saugojimą *fragmentų saugykloje* (metamodelį praplečiančiose duomenų lentelėse, kurios bus aprašytos vėliau). Eksportuojant duomenis iš CASE įrankio saugyklos, duomenų loginis modelis perkeliamas į fragmentų saugyklą, kur bus saugoma informacija susijusi tik su ta duomenų dalimi, kuri reikalinga duomenų modeliui eksportuoti į generuojamą SQL DDL aprašą bei pateikti duomenų analitikui, kad pastarasis galėtų peržiūrėti ar paredaguoti iš saugyklos įkeltą duomenų modelį. Tai lentelių bei jų atributų, tiek lokalių, tiek ir atkeliamų iš kitų esybių identifikuojančiais ryšiais, aprašai, greta kurių saugoma informacija apie apribojimus generuojamiems duomenims. Tolimesnis projektuojamos IS duomenų bazės modelio apdorojimas vyks ne tiesiogiai su CASE įrankio duomenų saugykloje aprašytu duomenų modeliu, o su fragmentų saugykloje saugomais ir apdorotais duomenimis. Naudojant fragmentų saugyklą sukurta galimybė išsaugoti ar pakartotinai įsikelti tam tikrą analizuojamą loginio duomenų modelio fragmentą.

Duomenų saugomų CASE įrankio duomenų saugykloje saugomo metamodelio, interpretavimas vyksta sekančiais žingsniais (11 pav.):

1. Iš metamodelio nuskaitoma informacija apie visas ankstesniuose funkcinių reikalavimų specifikavimo metodo etapuose išskirtas esybes ir pradedama analizuoti po vieną konkrečią esybę.
2. Išrenkami tiesiogiai esybei priklausantys atributai. t. y. atributai kurių tėvinė esybė yra apdorojamoji esybė.
3. Lygiagrečiai, išrenkama informacija susijusi su esybės atributais. Iš duomenų šaltinių atributų lentelės yra išrenkamas atributo būtinumo esybėje požymis, iš duomenų šaltinių apibrėžtos sąlygos sudaromas atributo reikšmių domenas.
4. Visa surinkta informacija apie atributo savybes yra saugoma fragmentų saugyklos lentelėse.
5. Išrenkami ryšiai kuriuose dalyvauja apdorojama esybė (tiek ryšiai kuriuose esybė – „tėvinė“, tiek ir tie kuriuose esybė apibrėžta kaip „vaikas“).
6. Iš išrinktos ryšių aibės ryšiai analizuojami nuosekliai :
7. Visų pirma tikrinama ar apdorojama esybė ryšyje dalyvauja kaip vaikas ar kaip tėvas :
   7.1. Jeigu esybė – „tėvas“, papildyti fragmentų saugyklą nauja esybe. Šis papildymas įvykdomas tik tada jei yra nurodyta importuoti susijusius atributus duomenų modelyje.
   7.2. Jeigu esybė – „vaikas“, tikrinama ar ryšys tarp esybių yra identifikuojantis :
      7.2.1. Jeigu ryšys identifikuojantis, rekursiškai kartoti identifikuojančių ryšių išrinkimo tėvinei esybei (kur tėvinė esybė ryšyje dalyvauja kaip vaikas) funkciją. Pasiekus giliausią iteraciją, grįžtama iki pradėtos analizuoti esybės, grįžtant formuojant pirminį esybės raktą. Esybės atributų sąrašas fragmentų saugykloje papildomas naujais atributais. Taipogi papildomas apdorojamos esybės raktas
      7.2.2. Jeigu ryšys neidentifikuojantis, papildyti esybės atributų sąrašą tėvinės esybės raktiniais atributais
8. Fragmentų saugykla papildoma naujais sąryšiais bei atributais.



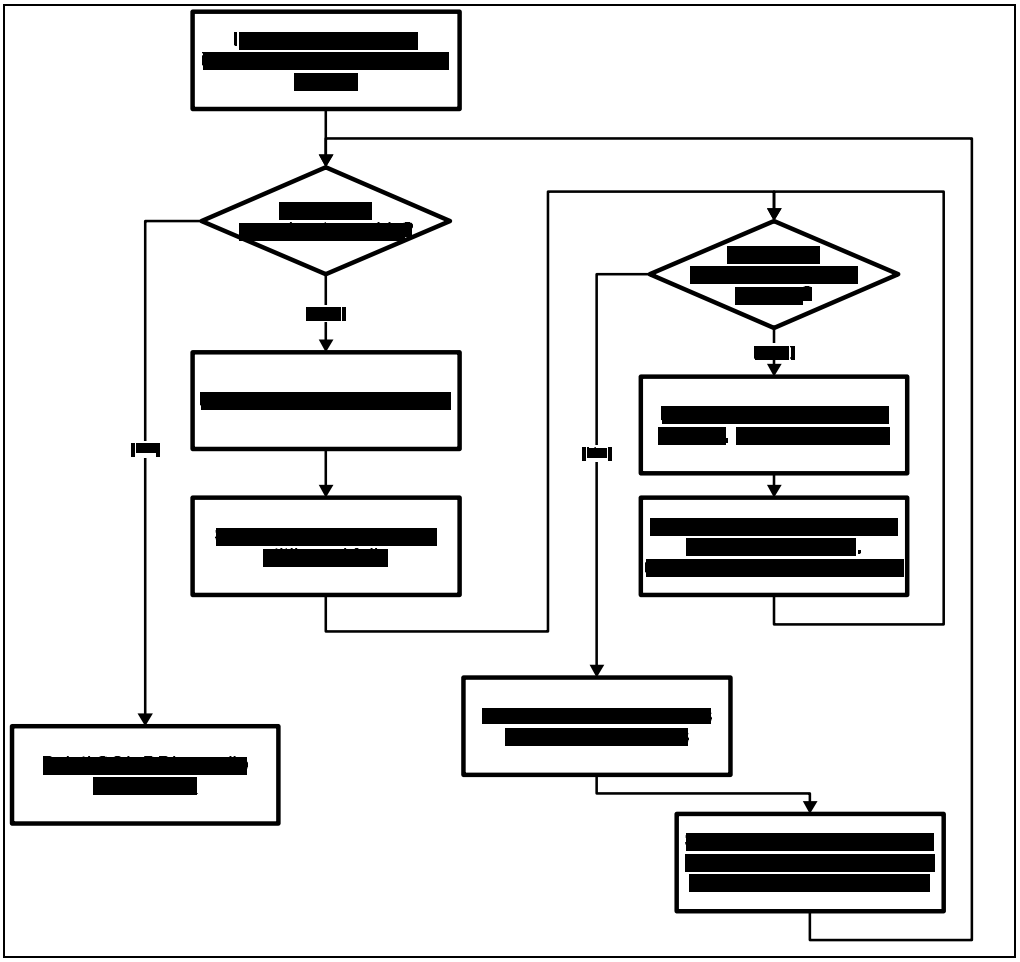
10 pav. Projektuojamos IS duomenų modelio analizės algoritmas

4.4 Loginio duomenų modelio transformavimo į reliacinę schemą algoritmas

Išanalizavus projektuojamos IS duomenų modelį ir išsaugojus jį fragmentų saugykloje, duomenų modelį galima pavaizduoti esybių ryšių diagrama. ER modelio vaizdavimui galima išrinkti tik reikalingas apdoroti esybes, kaip ir visas duomenų modelio esybes.

Patikrinus duomenų modelio korektiškumą, modelį paredagavus galima jį konvertuoti į SQL DDL aprašą. Konvertavimas vyksta sekančiais žingsniais (13 pav.):

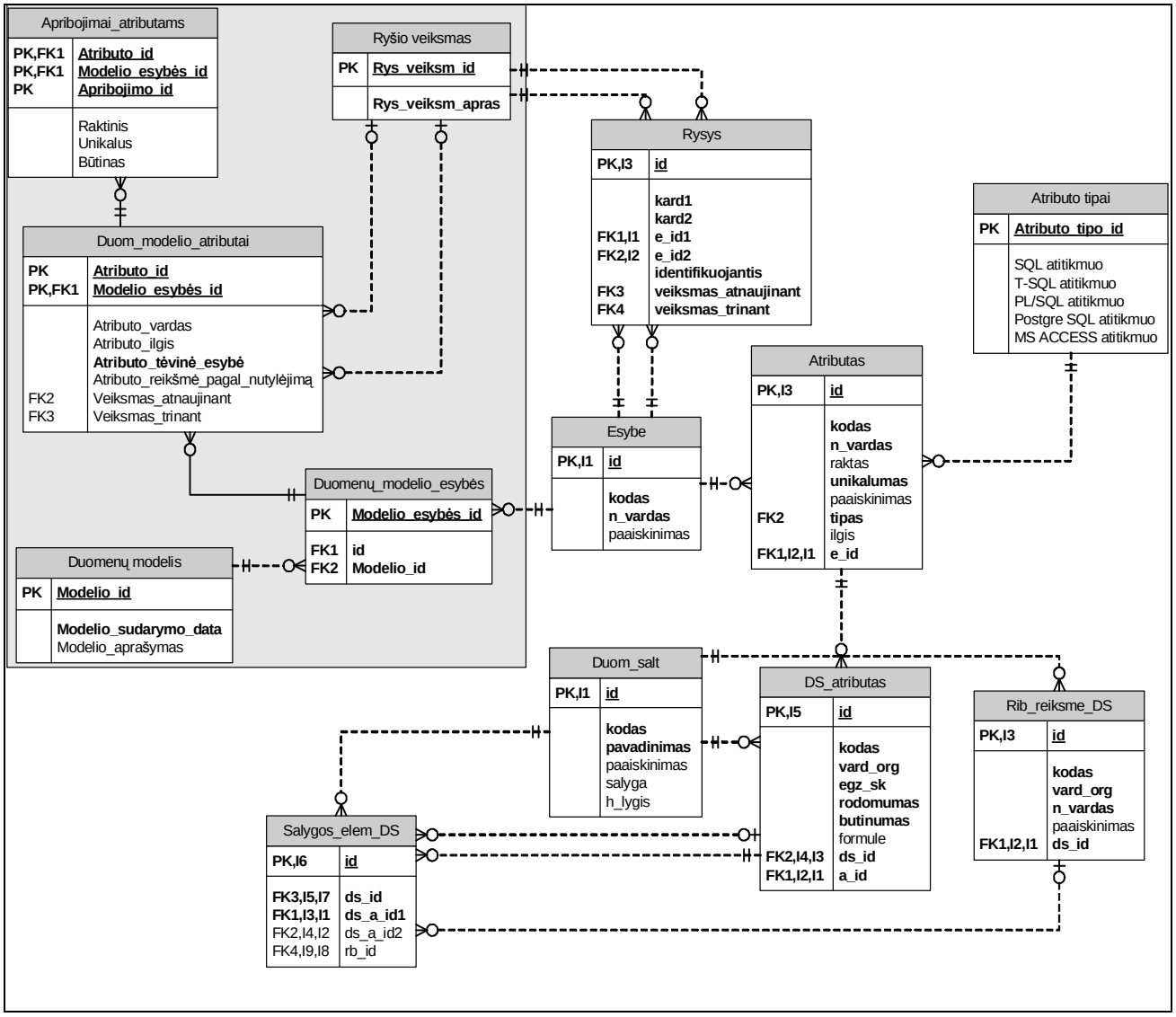
1. Iš tarpinės duomenų struktūros išrenkamos tos esybės kurios yra pateikiamos ir ER diagramoje (t.y. tik vizualizuotos esybės)
2. Patikrinama ar yra dar neapdorotų esybių
   2.1. Jei egzistuoja tokia esybė :
      2.1.1. nuskaityti tos esybės informaciją bei sukurti tuščią SQL DDL konstruktą **CREATE TABLE** <esybės pavadinimas> ( )
   2.2. Jei neegzistuoja daugiau neapdorotų esybių nutraukti aprašo generavimą
3. Patikrinama ar yra dar neapdorotų esybės atributų
   3.1. Jei toks atributas egzistuoja :
      3.1.1. į apdorojamos esybės lentelės kūrimo konstruktą įterpti atributo pavadinimą, duomenų tipą, atributo dydį ir **NULL**/**NOT NULL,** apribojimą
      3.1.2. jei atributui apibrėžtas pirminio rakto apribojimas įterpti **CONSTRAINT** <unikalus apribojimo pavadinimas> **PRIMARY KEY** <atributo pavadinimas>
      3.1.3. jei atributui apibrėžtas unikalumo apribojimas, įterpti **CONSTRAINT** <unikalus apribojimo pavadinimas> **UNIQUE**
   3.2. Jei visi esybės atributai apdoroti , išrinkti (jei tokių yra) tuos esybės atributus kurie tarpinėje struktūroje saugomi kaip priklausantys kitai esybei
      3.2.1. sugrupuoti atributus kurių tėvinė esybė ta pati
      3.2.2. pagal kiekvieną išskirtą atributų grupę, po lentelės kūrimą žyminčiu konstruktu sukurti išorinio rakto apribojimą esybei, tėvinės esybės egzemplioriaus pašalinimo ar atnaujinimo veiksmus pasirinkus iš *Ryšio veiksmas* laikinosios struktūros lentelės **ALTER TABLE** <apdorojama esybe> **ADD CONSTRAINT <**unikalus apribojimo pavadinimas**> FORIEGN KEY** <apdorojamo atributo pavadinimas> **REFERENTES <**tėvinė atributo esybe**>(**<apdorojamo atributo pavadinimas>) **ON DELETE** <veiksmas iš *ryšio veiksmas* lentelės> **ON UPDATE** <veiksmas iš *ryšio veiksmas* lentelės>
   3.3. Jei visi esybės atributai apdoroti ir išsaugoti išorinio rakto apribojimai, vykdyti 2 žingsnį.



11 pav. Projektuojamos IS duomenų saugyklos SQL DDL aprašo generavimo algoritmas

4.5 Metamodelio papildymas

Modifikavus duomenų metamodelį , t. y. papildžius jį naujais klasifikatoriais, bei fragmentų saugyklos lentelėmis, būtų galima realizuoti tam tikrų loginio duomenų modelio fragmentų išsaugojimą, ar duomenų analitiko pakeitimų duomenų modeliui išsaugojimą bei išsaugotų fragmentų pakartotinį įkrovimą iš naujo neanalizuojant duomenų modelio saugomo CASE įrankio saugykloje. Fragmentų saugykla pilnai nusako loginį duomenų modelį, taigi nebereikia atlikti pakartotinės metamodelyje saugomo loginio duomenų modelio analizės, kas reikalauja didelių laiko bei kompiuterio resursų sąnaudų esant didelėms loginio duomenų modelio specifikacijoms. Metamodelis su jį praplečiančiomis fragmentų saugyklos lentelėmis (pilkas fonas) esybių ryšių diagrama pateikiama 13 pav.



12 pav. Fragmentų saugykla papildytas metamodelis

Trumpas fragmentų saugyklos lentelių aprašymas :

- *Duomenų modelis* – lentelė skirta nurodyti saugomo duomenų modelio egzempliorių, nurodant šio egzemplioriaus sudarymo datą. Einamu momentu apdorojamas duomenų modelio fragmentas turi ***modelio_id*** = 0.
- *Duomenų modelio esybės* – šioje lentelėje nurodomos esybės kurios sudaro tam tikrą duomenų modelį. Pilnos CASE įrankio saugyklos analizės atveju duomenų modeliui priskiriamos visos rastos esybės.
- *Duomenų modelio atributai* – lentelė skirta loginio duomenų modelio esybių atributams bei informacijos susijusios su atributais (tėvinė esybė, veiksmai atnaujinant ar trinant tėvinės esybės egzempliorių, reikšmė pagal nutylėjimą) saugoti.
- *Apribojimai atributams* – lentelė skirta saugoti atributų apribojimus.

5 Duomenų bazės schemos SQL aprašo generavimo modulio realizacija

5.1 Reikalavimų specifikacija

5.1.1 Projekto kūrimo pagrindimas

Nesuprasti vartotojų reikalavimai yra viena iš pagrindinių programinės įrangos (PĮ) projektų žlugimo priežasčių. Vartotojo reikalavimų supratimas įgyjamas reikalavimų inžinerijos procese. Šiuo metu egzistuoja keletas metodų reikalavimams specifikuoti bei perkelti į projekto specifikaciją. Tai *Oracle CASE* metodas [9], *Rational Unified* procesas [16] ir kiti. Tačiau CASE (*Computer Aided System Engineering*) priemonėse siūloma specifikavimo technologija neatitinka natūraliu būdu naudojamos vartotojo reikalavimų bei projektinių sprendimų registravimo technologijos. Todėl buvo pasiūlyta realizuoti CASE įrankį, naudojantis sukurtu organizacijos veiklos modeliavimu paremtu metodu, kuris leidžia specifikuoti kompiuterizuotai IS keliamus funkcinius reikalavimus natūraliai ir sklandžiai[7].

Šio projekto tikslas – funkcinių reikalavimų specifikacijos metodo pagrindu kuriamo CASE įrankio modulio realizacija. Projekto metu kuriamo CASE modulio pagalba, turi būti galima išanalizuoti, CASE įrankio saugykloje esantį kuriamos programų sistemos duomenų bazės modelį. Pasirinktą modelio epizodą, ar pilną modelį atvaizduoti ER diagrama, bei sugeneruoti SQL DDL aprašą, iš kurio bus sukuriama fizinė duomenų bazė Microsoft SQL, Oracle ar kitoje duomenų bazių valdymo sistemoje.

5.1.2 Apribojimai reikalavimams

5.1.2.1 Apribojimai sprendimui

- Sistema turi veikti Microsoft Windows 9x/ME/2000/XP operacinėse sistemose;
- Sistemai būtina, jog operacinėje sistemoje būtų įdiegtas Microsoft .NET framework (v1.0) darbo platforma (*framework*);
- Sistema naudojasi Microsoft Access duomenų bazių valdymo sistema (per ODBC sąsają);
- Sistema, sąveikai su kitomis programomis naudoja ODBC sąsają;
- Sistema turi dirbti su dideliu kiekiu įrašų, neperkraunant kompiuterio resursu;

5.1.2.2 Diegimo aplinka

Diegimo aplinkos charakteristikos :

- Windows 2000/XP operacinės sistemos platforma;
- MS Visio 2000/2002 tekstinis redaktorius;
- MS Access duomenų bazių valdymo sistema.

5.1.2.3 Bendradarbiaujančios sistemos

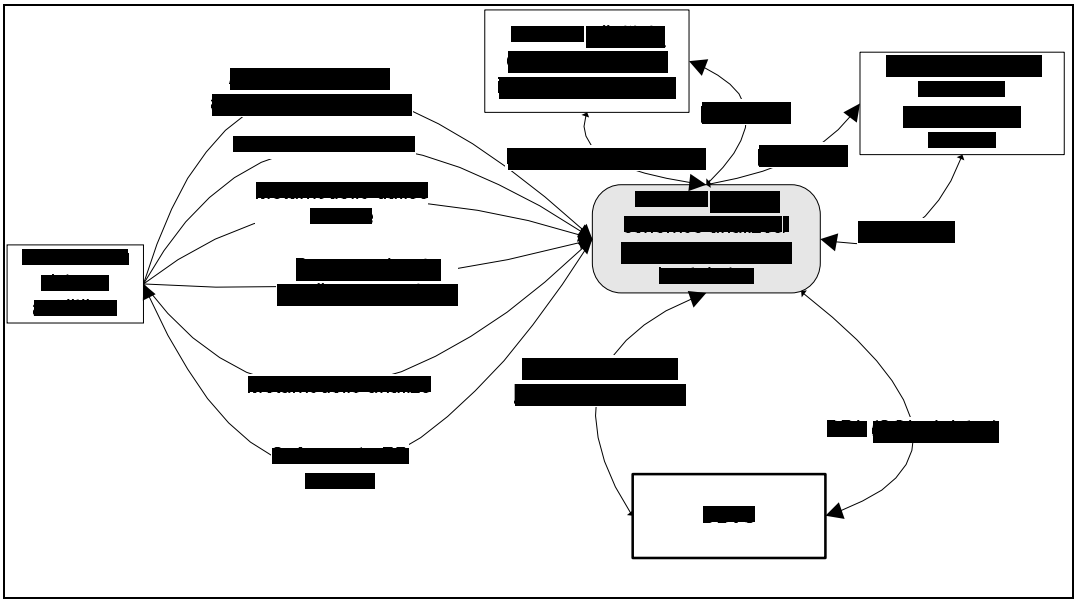
Kuriamas programinis modulis yra dalis CASE įrankio, realizuojančio IS kūrimą funkcinių reikalavimų specifikacijos pagrindu. Šis modulis naudosis šių posistemių rezultatais:

- Funkcijų hierarchijos specifikavimo modulis;
- Rezultatų ir DŠ(duomenų šaltinių) struktūros specifikavimo modulis;
- Ryšių tarp DŠ ir jų sudėties specifikavimo modulis;
- Ryšių tarp duomenų šaltinių ir DŠ būsenų specifikavimo modulis;
- Integruoto IS duomenų modelio sudarymo modulis;

Šio CASE įrankio modulio rezultatus – sugeneruotą duomenų bazės schemos aprašą, bus galima suvykdyti į MS SQL Server DBVS ar Oracle DBVS

5.1.3 Veiklos kontekstas

Pateikiama kuriamo CASE įrankio modulio veiklos konteksto diagrama (14 pav.), kuri apibrėžia šios sistemos ribas, išorines esybes, kurios bendrauja su sistema, bei pagrindinius informacijos srautus tarp sistemos ir išorinių esybių.



13 pav. Sistemos veiklos kontekstas

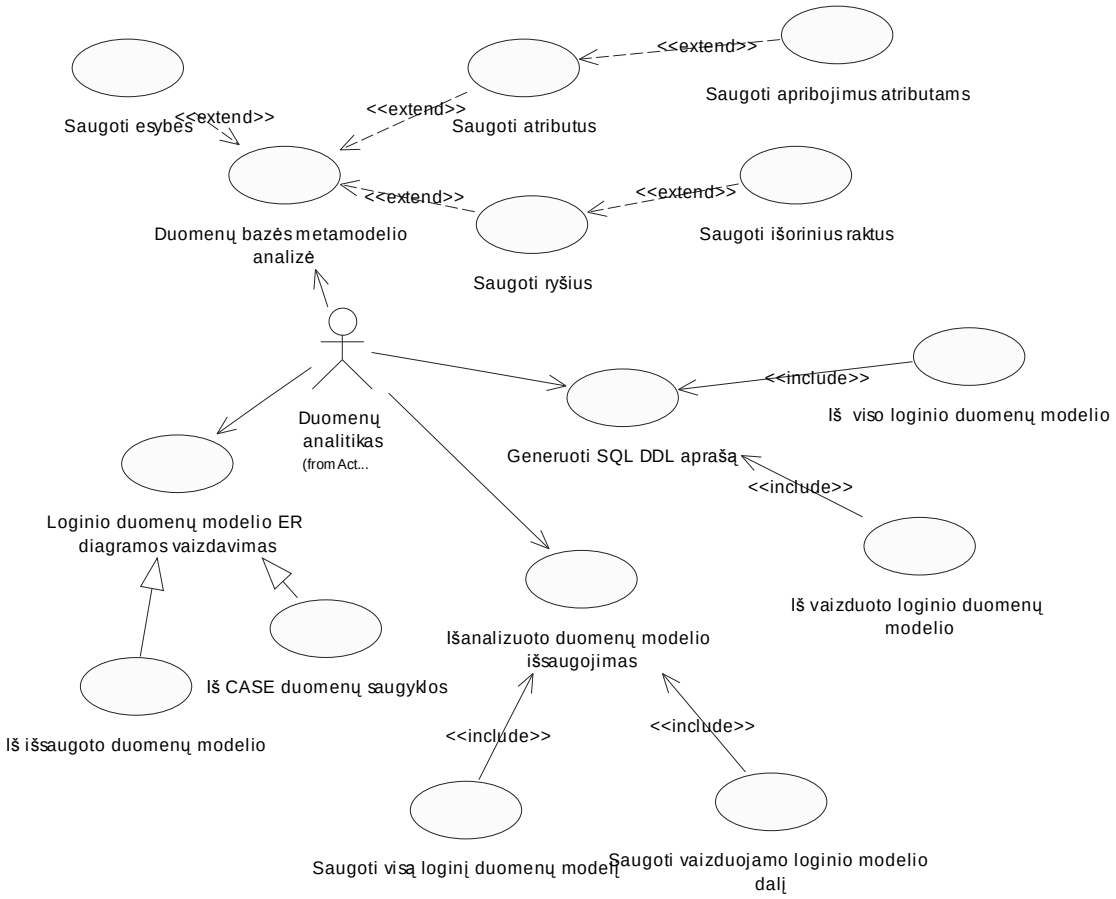
Pateiktas veiklos įvykių sąrašas, kuris apima visus veiklos įvykius, už kuriuos atsakinga nagrinėjama veikla.

4 lentelė Veiklos įvykių lentelė

#:	Įvykio pavadinimas	Įeinantys/išeinantys informacijos srautai
1.	Metamodelio analizė	Metamodelis (in)
2.	Dalies metamodelio analizė	Metamodelio dalis (in)
3.	Duomenų bazės aprašo generavimas	SQL DDL skriptas (out)
4.	Suformuota ER schema	ER diagramos vaizdas (out)
5.	Apribojimų esybių atributams nustatymas	Papildomų apribojimų duomenų modeliui įvedimas (in)
6.	Duomenų specifikacija	Duomenų modelių apjungtas modelis (in)
7.	Pakeitimai duomenų modeliui	Modelio savybių modifikacijos (in)
8.	Apribojimai	Apribojimai duomenų modelio esybių atributams (in)
9.	Pakeitimai duomenų modelio apribojimams	Modelio atributų ar ryšių apribojimų nustatymas(in)
10.	DDL (SQL skriptas)	Duomenų bazės schemos aprašas SQL DDL skripto pavidalu (out)
11.	Patvirtinimas apie įvykdytus veiksmus	Patvirtinimas apie importuotą ar atmestą duomenų bazės schemos aprašą (in)

5.1.4 Panaudojimo atvejų sąrašas

Panaudojimo atvejų diagrama (15 pav.) nusako ribas tarp sistemos ir vartotojo. Toliau yra pateikiamas reikalavimų metu suformuotų panaudojimų atvejų aprašas.



14 pav. Panaudojimo atvejų diagrama

1.Panaudojimo atvejis	**Duomenų bazės metamodelio analizė**
Tikslas	Išanalizuoti CASE įrankio saugykloje saugomą projektuojamos IS duomenų modelį
Aktoriai	Sistemos analitikas

2.Panaudojimo atvejis	**Saugoti esybes**
Tikslas	Saugoti esybes aprašytas CASE įrankio saugykloje į *fragmentų saugyklos* esybių lentelę
Aktoriai	Analizavimo posistemė

3.Panaudojimo atvejis	**Saugoti atributus**
Tikslas	Saugoti atributus priklausančius nurodytai esybei *fragmentų saugyklos* apribojimų lentelėje
Aktoriai	Analizavimo posistemė

4. Panaudojimo atvejis	**Saugoti apribojimus atributams**
Tikslas	Saugoti apribojimus nustatytus nurodytiems esybės atributams, *fragmentų saugyklos* atributų apribojimų lentelėje
Aktoriai	Analizavimo posistemė

5. Panaudojimo atvejis	**Saugoti ryšius**
Tikslas	Saugoti raktinius atributus priklausančius esybei, kurios vaidmuo sąryšyje yra „tėvas“, toje esybėje kurios vaidmuo sąryšyje yra „vaikas“. Atributų kilmės esybe nurodoma „tėvo“ esybė.
Aktoriai	Analizavimo posistemė

6. Panaudojimo atvejis	**Saugoti išorinius raktus**
Tikslas	Saugoti atributus priklausančius „protėvinei“ esybei, kuri gali būti pasiekiama identifikuojančiais ryšiais
Aktoriai	Analizavimo posistemė

7. Panaudojimo atvejis	**Generuoti SQL DDL aprašą**
Tikslas	Modifikuoti duomenų bazės modelį sistemos atmintyje ir atvaizdavime, kad būtų galima generuoti SQL DDL aprašą jau su įvykdytomis modifikacijomis.
Aktoriai	Duomenų analitikas

8. Panaudojimo atvejis	**Iš viso loginio duomenų modelio**
Tikslas	Generuoti SQL DDL aprašą iš viso išanalizuoto *fragmentų saugykloje* saugomo išanalizuoto loginio duomenų modelio.
Aktoriai	Duomenų analitikas

9. Panaudojimo atvejis	**Iš vaizduoto loginio duomenų modelio**
Tikslas	Generuoti SQL DDL aprašą, generavimo metu iš *fragmentų saugyklos* išrenkant tik tas esybes kurios vaizduojamos ER diagramoje
Aktoriai	Duomenų analitikas

10. Panaudojimo atvejis	**Išanalizuoto duomenų modelio išsaugojimas**
Tikslas	*Fragmentų saugykloje* laikinai saugomą išanalizuotą IS metamodelį (loginį duomenų modelį) saugoti duomenų modelių saugykloje
Aktoriai	Duomenų analitikas

11. Panaudojimo atvejis	**Saugoti vaizduojamo loginio modelio dalį**
Tikslas	Saugoti loginį duomenų modelį *fragmentų saugykloje* išrenkant tik tas esybes kurios vaizduojamos ER diagramoje
Aktoriai	Duomenų analitikas

12. Panaudojimo atvejis	**Saugoti visą loginį duomenų modelį**
Tikslas	Saugoti visą laikinai saugomą išanalizuotą IS metamodelį(loginį duomenų modelį) duomenų modelių saugykloje.
Aktoriai	Duomenų analitikas

13. Panaudojimo atvejis	**Loginio duomenų modelio ER diagramos vaizdavimas**
Tikslas	Vaizduoti duomenų analitiko nurodytą išanalizuoto IS metamodelio dalį (loginio modelio dalį) ar visą išanalizuotą metamodelį
Aktoriai	Duomenų analitikas

14. Panaudojimo atvejis	**Iš išsaugoto duomenų modelio**
Tikslas	Vaizduoti duomenų loginį modelį iš pasirinkto, saugomo duomenų modelių saugykloje, duomenų modelio.
Aktoriai	Duomenų analitikas

15. Panaudojimo atvejis	**Iš CASE duomenų saugyklos**
Tikslas	Vaizduoti duomenų loginį modelį iš CASE įrankio duomenų saugykloje saugomo IS metamodelio, prieš tai jį išanalizavus
Aktoriai	Duomenų analitikas

5.1.5 Funkciniai reikalavimai

Reikalavimai CASE įrankio duomenų saugyklos analizei:

Reikalavimas 1 : Sistema turi tinkama forma saugoti išanalizuotą duomenų modelį CASE įrankio duomenų saugyklos dalyje – fragmentų saugykloje.

Reikalavimas 2 : Sistema privalo išsaugoti išanalizuotą, fragmentų saugykloje saugomą loginį duomenų modelį : *nutrūkus ryšiui su CASE įrankio duomenų saugykla, programai baigus darbą dėl klaidos.*

Reikalavimas 3 : Sistema turi leisti išsaugoti loginio duomenų modelio dalį ar visą interpretuotą loginį modelį fragmentų saugykloje

Reikalavimai ER diagramos vaizdavimui/redagavimui:

Reikalavimas 4 : Sistema turi pateikti loginį duomenų modelį *informacijos inžinerijos notacijoje.*

Reikalavimas 5 : Sistema turi leisti vaizduoti tik tam tikras loginio modelio esybes, bei su pažymėtomis esybėmis susijusias esybes

Reikalavimas 6 : Sistema turi leisti vaizduoti iki tam tikro lygio susijusias esybes.

Reikalavimas 7: Sistema turi leisti redaguoti tam tikrus loginio duomenų modelio ER diagramos elementus, pakeitimus laikinai išsaugant.

Reikalavimai SQL DDL aprašo generavimui:

Reikalavimas 8: Generuojant SQL DDL aprašą būtina sugeneruotą modelį išsaugoti tekstiniame faile kurį vėliau galėtų nuskaityti reliacinė DBVS

Reikalavimas 9: Generuojant tik dalį loginio modelio, į SQL DDL aprašą nereikia įtraukti atributų bei apribojimų iš nevaizduojamų esybių

Reikalavimas 10: Sistema privalo užtikrinti SQL DDL aprašų korektiškumą, bei atitikimą SQL-92 standartui.

5.1.6 Nefunkciniai reikalavimai

Reikalavimai sistemos išvaizdai

- Turi būti naudojama grafinė vartotojo sąsaja.
- Sąsaja turi būti lengvai skaitoma, suprantama.
- Sąsaja turi būti sąveikaujanti.
- Turi būti išlaikomi realizuojamo CASE įrankio grafinės vartotojo sąsajos standartai.

Reikalavimai panaudojamumui

- Paprasta naudotis IS projektuotojams- analitikams.
- Turi būti lengvai išmokstama dirbti su sistema.

Reikalavimai vykdymo charakteristikoms

- Interaktyvaus bendravimo su vartotoju metu programa turi veikti pakankamai greitai, užtikrinti efektyvų darbą.
- Sistema neturi reikalauti daugiau resursu nei to maksimaliai gali prireikti Microsoft aplinkai.

5.2 Architektūros specifikacija

Šiame skyriuje yra specifikuojama “Duomenų bazės schemos SQL aprašo generavimas” projekto architektūra, kurios pagrindu realizuotas CASE įrankio modulis.

Kuriamos programų sistemos architektūros parinkimas – svarbus projektinis sprendimas – atliekamas vadovaujantis reikalavimais gautais analizės etape. Architektūros grafinėms pateiktims bus naudojama UML (*Unified Modeling Language)* notacija. Kuriama sistema nėra paskirstyta ir yra projektuojama veikti vienoje darbo stotyje.

Esminiai reikalavimai, kurie įtakoja architektūros specifikavimą:

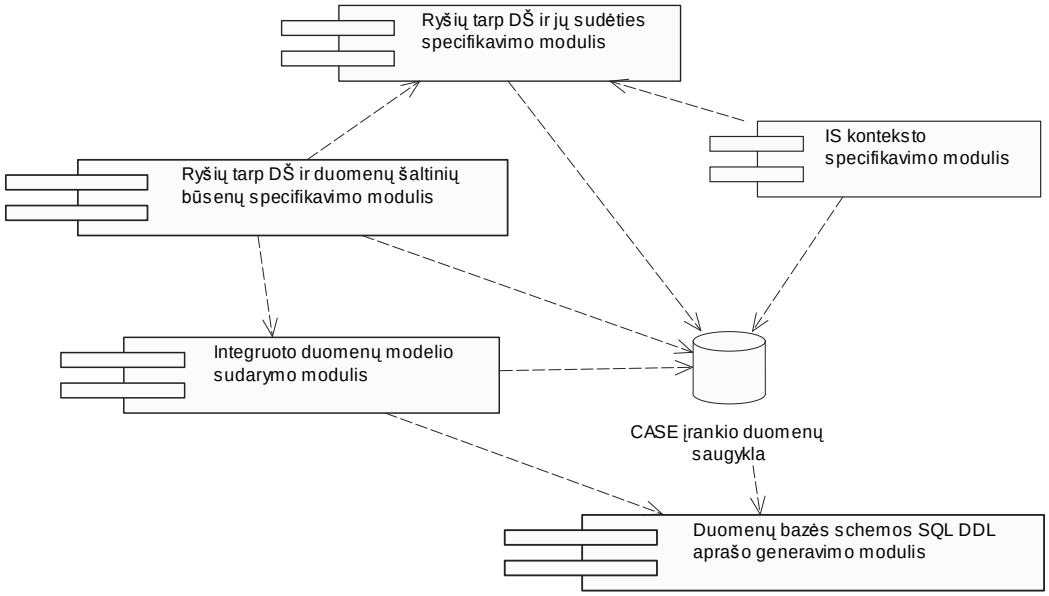
- Duomenų bazės schemos SQL aprašo generavimo posistemė turi būti integruota į kuriamą sistemą, užtikrinant korektišką CASE įrankio funkcionavimą.
- Kadangi kai kurie ankstesnes funkcinių reikalavimų specifikavimo metodo fazes realizuojantys CASE įrankio moduliai realizuoti MS Visio 2000/2002 aplinkoje, realizuojamas modulis turi būti sukurtas taip, kad jį būtų galima iškviesti iš MS Visio aplinkos.

Kadangi jau suprojektuoti ir realizuoti CASE įrankio moduliai buvo sukurti kaip Microsoft Visio paketo plėtiniai, tikėtasi duomenų bazės schemos generavimo įrankį realizuoti naudojant Microsoft Visio šabloną *Database modeling*. Tačiau ištyrus biblioteką (*Modelenglib.dll*) kurią naudojant yra operuojama duomenų modelio elementais, buvo nustatyta, kad Microsoft korporacija neteikia įrankių ir galimybių sudaryti duomenų modeliams Visio paketo gamintojų ar trečiosios šalies realizuotomis priemonėmis [19].

Kadangi, kaip minėta, CASE įrankio moduliai buvo sukurti naudojant Microsoft technologijas, naujo CASE įrankio modulio realizacijai buvo pasirinkta naujausia platforma skirta programų kūrimui siūloma Microsoft korporacijos – *Microsoft .NET* .

5.2.1 Duomenų bazės schemos SQL aprašo generavimo modulio sąveika su kitomis sistemomis

Komponentų diagrama (16 pav.) atvaizduoja projektuojamos sistemos vietą kitų sistemų kontekste, parodo, kad duomenų bazės schemos SQL aprašo generavimo modulis yra vienas iš CASE įrankio modulių. Šis modulis kaip ir visi CASE įrankio moduliai naudojasi anksčiau paruošta ir nuolat papildoma bei atnaujinama informacijos srautų specifikacijos metaduomenų baze.



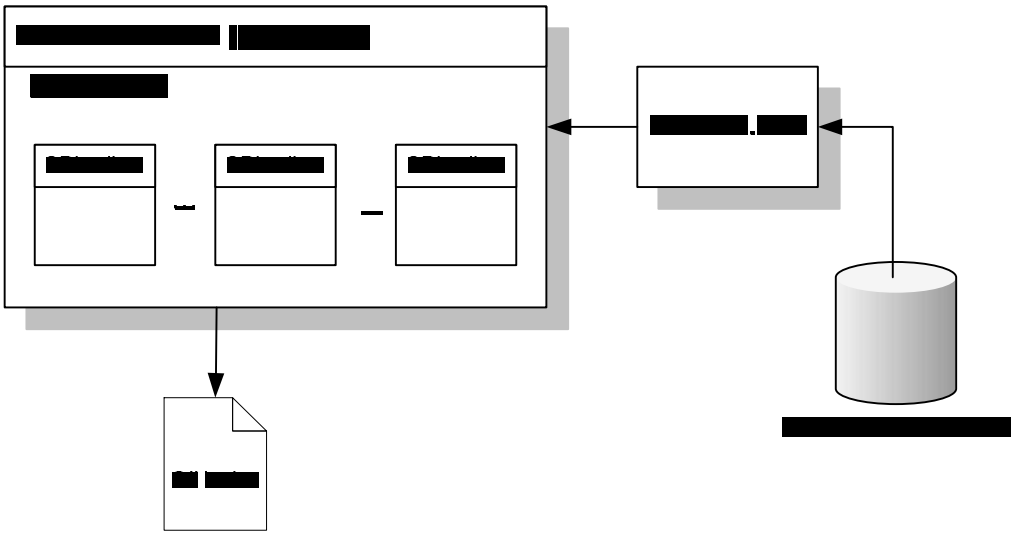
15 pav. Duomenų bazės schemos SQL aprašo generavimas modulio vieta IS kompiuterizavimo CASE modulių kontekste

5.2.2 CASE įrankio modulio principinė schema

Duomenų bazės schemos SQL aprašo generavimo įrankiui realizuoti panaudota Microsoft .NET technologija. Sąsaja su CASE įrankio duomenų saugykla realizuota ADO.NET technologijomis, ODBC sąsajos pagalba. Duomenys transformuojami pačioje CASE įrankio duomenų saugykloje, einamąjį duomenų modelį išsaugant su atitinkamu požymiu duomenų modelių saugykloje (CASE įrankio saugyklos viduje). Duomenų modelio ER diagramos vaizdavimas atliekamas MS Windows formų tipo programos lange, esybes vaizduojant kaip SDI (*single document interface)* langus, o ryšius braižant ant MDI (*multiple document interface)* fono. Principinė įrankio struktūros schema pateikta 17 paveiksle. Žemiau pateiktas šios schemos komentaras:

- Microsoft Access duomenų bazė. Šioje bazėje saugomi specifikuoti elementai.
- Microsoft ADO.NET biblioteka, tai MS.NET biblioteka skirta programinės kalbos priemonėmis pasiekti ir valdyti lokalios arba nutolusios duomenų bazę duomenis, objektus ir struktūrą per įvairias sąsajas (ODBC, OLEDB ir kt.).
- Microsoft Windows forma
- Microsoft Windows formos pagrindinis langas (*MDI parent)* ir vidiniai langai (*SDI children)* ir jos grafinė sąsaja su vartotoju. Kiekvienas vidinis langas priklauso pagrindiniam langui ir žymi esybes, o ryšiai tarp esybių braižomi pagrindinio lango fone.

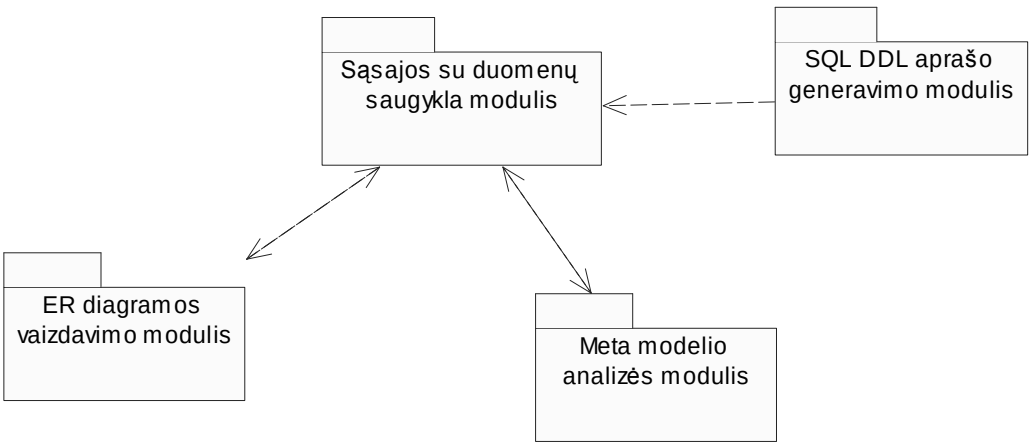
- Duomenų modelio diagramos vaizdavimo programinė dalis. Programinę dalį sudaro C# programavimo kalba užrašytas kodas. Vienam MS Windows formos langui pavaizduoti.



16 pav. Įrankio principinė struktūros schema.

5.2.3 Sistemos paketų diagrama

Projektuojamame įrankyje yra išskirti keturi pagrindiniai sistemos paketai (18 pav.). Toliau pateikiamas trumpas kiekvieno modulio aprašas:



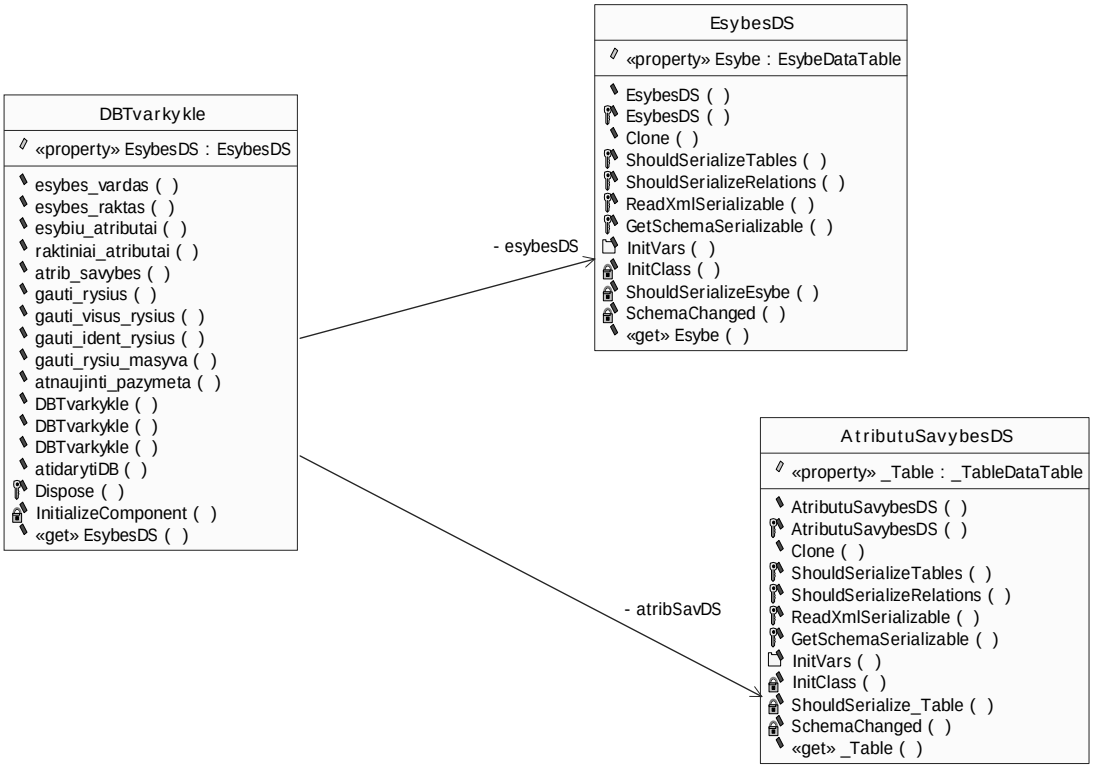
17 pav. Pagrindiniai realizuojamo įrankio moduliai

Žemiau kiekvienas paketas yra detalizuojamas trumpai aprašomos kiekvieną paketą realizuojančios svarbiausios klasės.

5.2.3.1 Sąsajos su duomenų saugykla paketas

Sąsajos su duomenų saugykla pakete realizuota sąsaja su CASE įrankio duomenų saugykla esančia Microsoft Access duomenų bazėje. Modulio funkcionalumas :

- § Užklausų siuntimas į CASE įrankio duomenų bazę per ODBC sąsają;
- § Įvairių nagrinėjamo metamodelio elementų išgavimas, bei perdavimas;
- § Informacijos fragmentų saugykloje atnaujinimas;
- § Informacijos iš fragmentų saugyklos nuskaitymas.



18 pav. Sąsajos su duomenų saugykla modulis

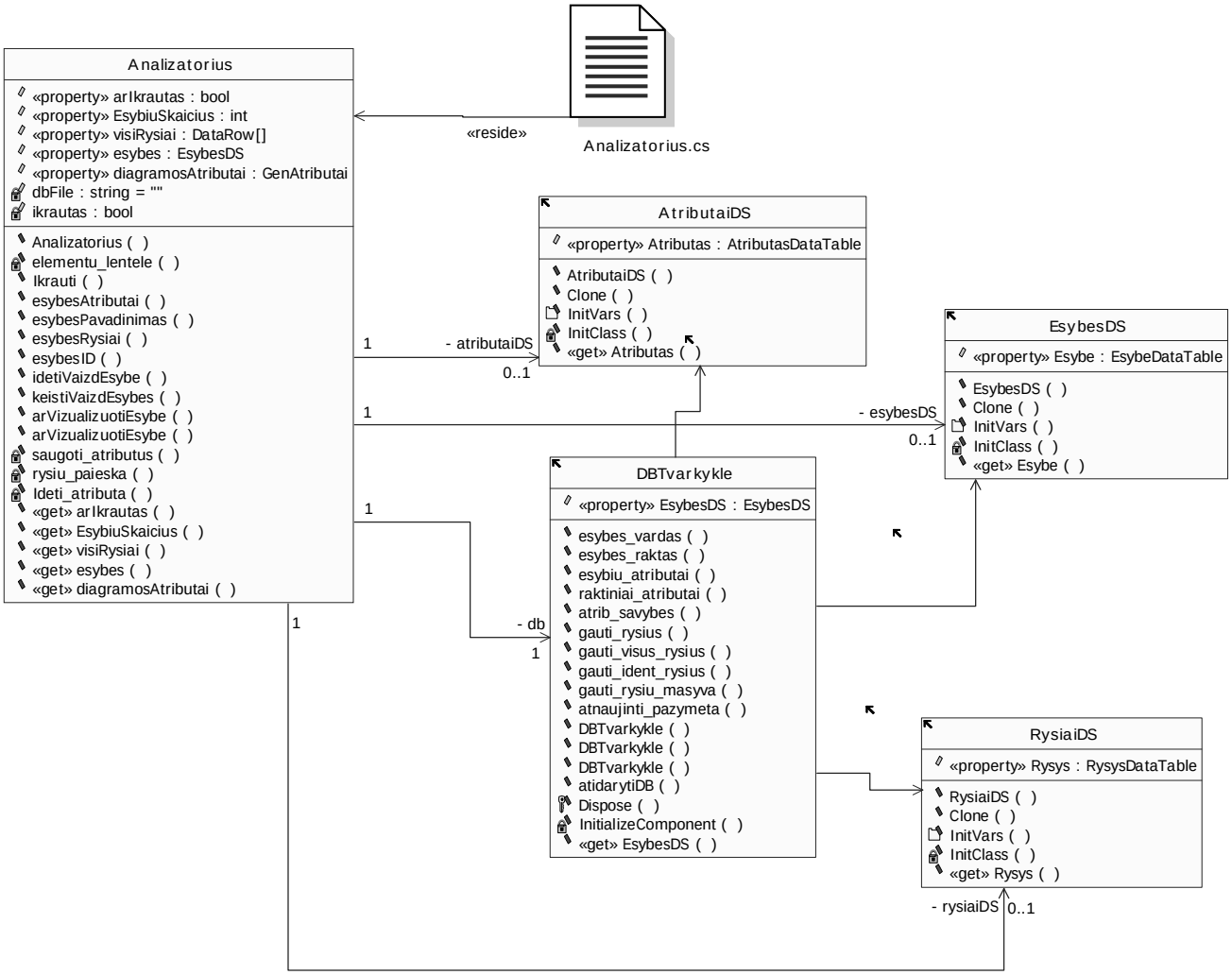
DBTvarkyklė (klasė-komponentas) skirta tiesiogiai sąveikauti su duomenų CASE įrankio duomenų saugykla, t. y. prisijungimą per ODBC sąsają, įvairių užklausų reikalingų analizavimo ir vaizdavimo procese vykdymą

5.2.3.2 Duomenų modelio analizės paketas

Duomenų modelio metabazėje analizės modulis (20 pav.) naudojamas, CASE įrankio saugykloje saugomo duomenų bazės modelio nuskaitymui ir interpretavimui. Šis modulis yra esminis *ER atvaizdavimo bei SQL DDL aprašo generavimo posistemyje* nes jo suformuotus rezultatus naudoja *ER diagramos vaizdavimo modulis* bei *SQL DDL aprašo generavimo modulis*.

Šio paketo pagalba taipogi vyksta anksčiau išsaugotų duomenų modelių įkrovimas į sistemą. Šio modulio funkcionalumas :

- § Duomenų saugykloje esančio duomenų modelio (*esybių, atributų, ryšių tarp esybių)* analizavimas bei išanalizuotų duomenų perkėlimas į fragmentų saugyklą;
- § Fregmentų saugykloje saugomų duomenų modelių perkėlimas



19 pav. Duomenų modelio analizės paketas

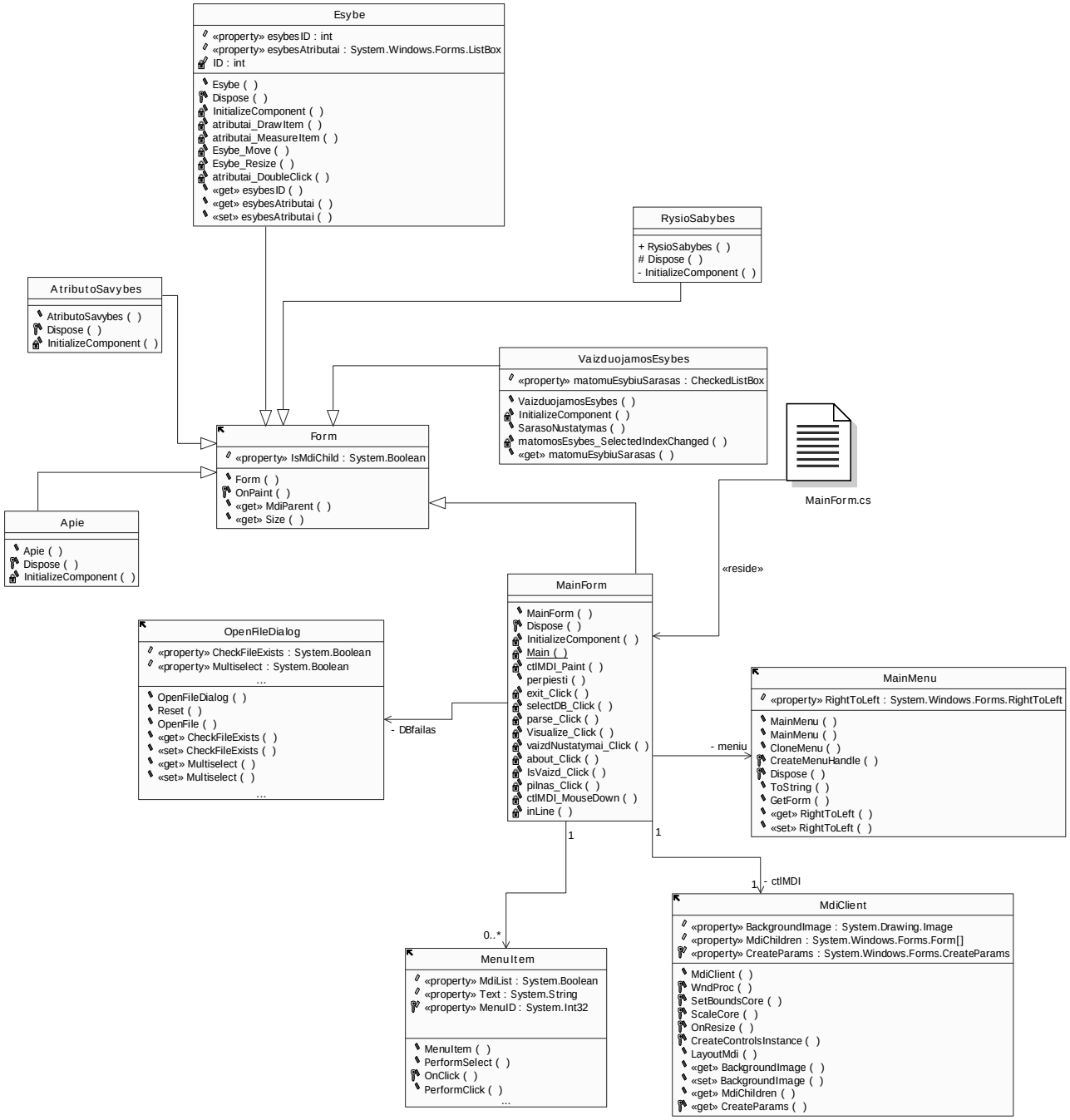
Analizatorius (klasė). Šios klasės paskirtis – CASE įrankio saugykloje saugomo duomenų modelio interpretavimą bei modelio informacijos saugojimą kuriamajame modulyje. Taipogi ši klasė saugo informaciją apie vizualizuojamą modelio dalį.

AtributaiDS (klasė). Šios klasės paskirtis – esybių atributų saugojimas analizatoriaus klasėje Ši klasė yra rišantis atributų duomenų saugykloje bei atributų saugomų fragmentų saugykloje elementas.

RysiaiDS (klasė). Ši klasė reikalinga ryšių saugomų CASE įrankio duomenų saugykloje ir susijusių su viena ar daugiau esybių perdavimui į analizatoriaus modulį.

5.2.3.3 *ER diagramos vaizdavimo paketas*

Šis įrankio paketas (21 pav.) naudojamas išanalizuoto duomenų modelio esančio fragmentų saugykloje atvaizdavimui ER diagrama. Be to, šiame pakete yra realizuota galimybė duomenų analitikui modifikuoti loginį duomenų modelį. ER diagramos vaizdavimo paketas yra didelis funkcionalumo prasme. Naudojant šį paketą atliekamas visos programos valdymas (šio paketo pagalba yra realizuotas meniu valdymas) bei vartotojui pateikiama duomenų bazės aprašo diagrama. Taipogi galima peržiūrėti duomenų bazės diagramos detales, jas modifikuoti. Šio paketo funkcionalumas :



20 pav. ER diagramos vaizdavimo paketas

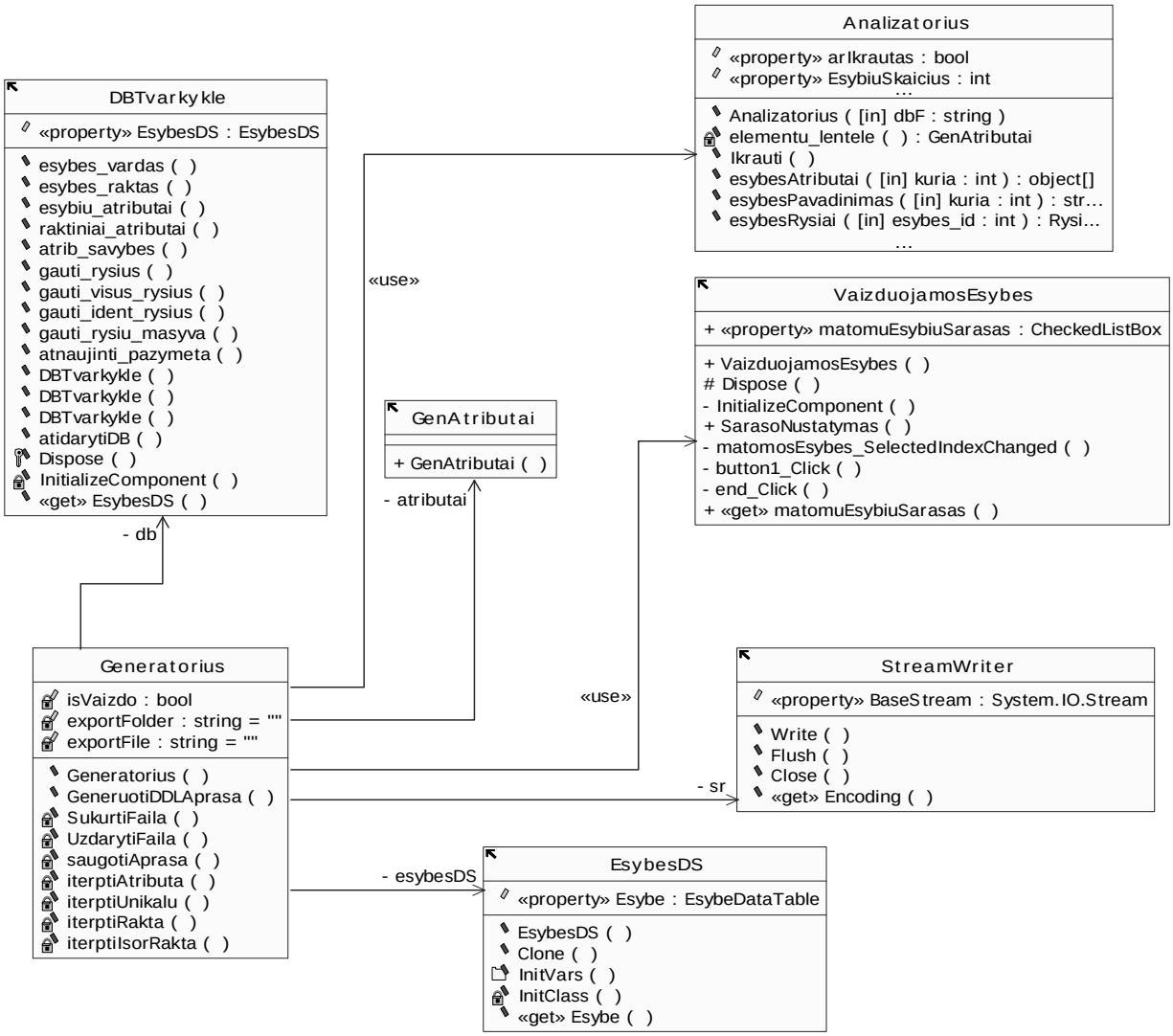
- § Programos meniu valdymas;
- § ER diagramos vaizdavimas;
- § Dalies ER diagramos vaizdavimas;
- § Duomenų bazės diagramos atributų modifikavimas;

Esybė(klasė-forma). Šios klasės paskirtis – pavaizduoti esybę ER diagramoje. Kiekvienas šios klasės atributas vienaip ar kitaip apibūdina vaizduojamą esybę.
RysioSavybes (klasė-forma). Ši klasė pateikia duomenis apie vartotojo pažymėtą ryšį bei nurodo ryšio galų kardinalumus, požymį ar ryšys identifikuojantis bei ryšio veiksmą.
AtributoSavybes (klasė-forma). Ši klasė pateikia duomenis apie vartotojo pažymėtą atributą parodydama atributo savybes : *tipą, ilgį, būtinumo apribojimą, unikalumą bei požymį ar atributas yra raktinis.*
VaizduojamosEsybės (klasė-forma). Naudojant šią klasę yra nustatomas sąrašas esybių kurios bus vaizduojamos ER diagramoje. Taipogi galima nustatyti, ar papildomai reikia vaizduoti su esybėmis susijusias esybes ar tik nurodytas.
MainForm (klasė-forma). Ši naudojama programos veiksmams valdyti, kadangi joje yra patalpintas pagrindinis programos meniu. Taipogi naudojant šią formą yra braižoma ER diagrama, taigi diagramos savybių nuskaitymas/nustatymas vyksta šioje formoje.
MDIClient (klasė). Ši klasė naudojama kaip ER diagramos braižymo pagrindas. Joje vaizduojami visi ryšiai tarp esybių, taipogi „gaudomi“ pelės spustelėjimo įvykiai. Šioje formoje yra nurodytos visos esybės, formuojamas išdėstymas.

5.2.3.4 *SQL DDL aprašo generavimo paketas*

SQL DDL (*Data Definition Language*) aprašo generavimo paketas (22 pav.) skirtas duomenų bazės SQL DDL aprašą saugančio failo generavimui. Šis modulis iš programos pagrindiniame lange vaizduojamos duomenų diagramos generuoja pilnai į DBVS importuojamą SQL aprašą, kurį įvykdžius DBVS gaunama CASE įrankiu projektuojamos informacinės sistemos duomenų bazė. Šio modulio funkcionalumas :

- § Sugeneruoti visos duomenų bazės SQL aprašą;
- § Sugeneruoti tik vaizduojamos duomenų bazės dalies SQL aprašą;



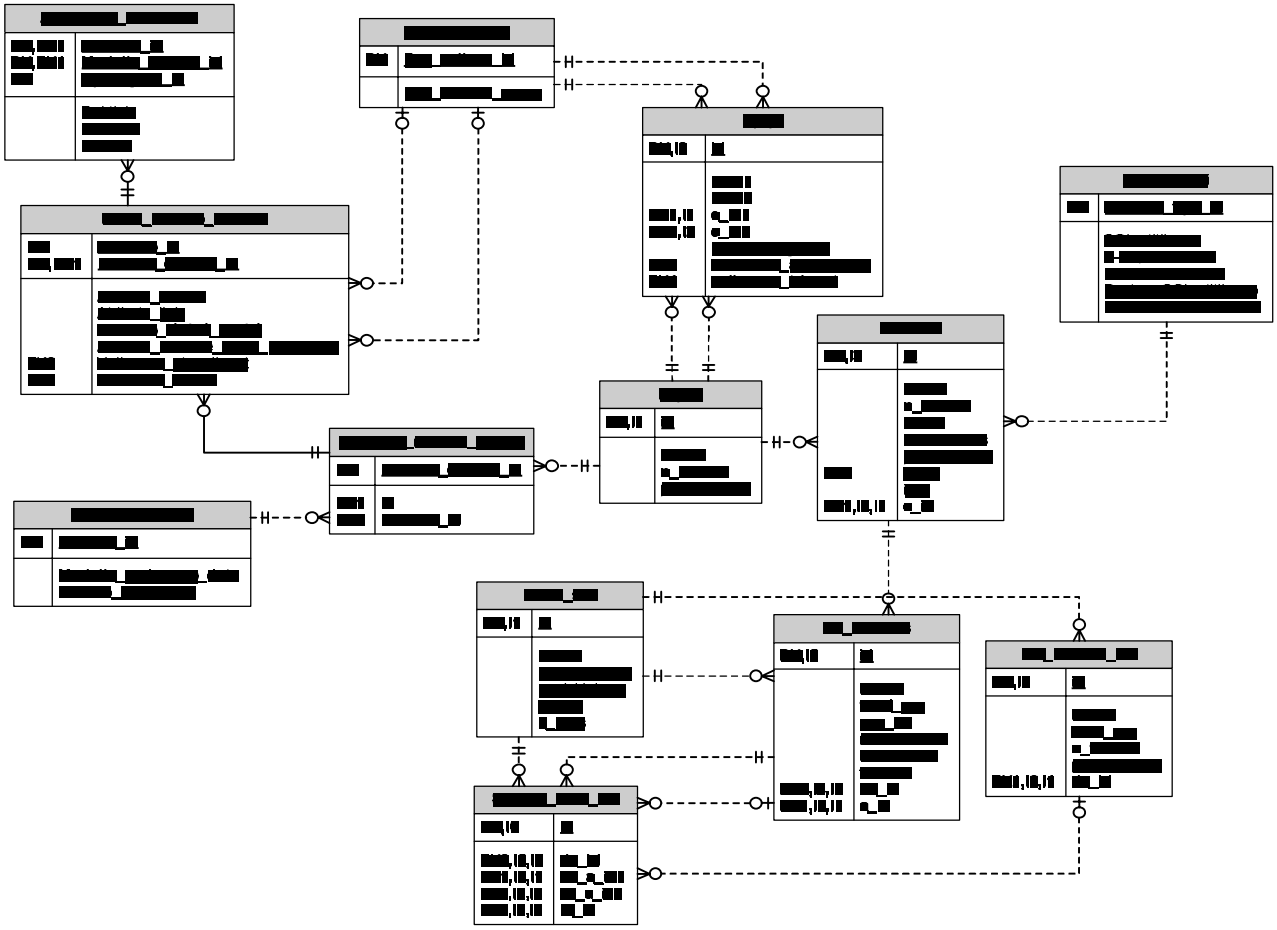
21 pav**. SQL DDL aprašo generavimo modulis**

GenAtributai (klasė). Ši naudojama eksportuojamų į SQL DDL aprašą atributams saugoti. Naudojant šią klasę, galima greitai išgauti reikiamą informaciją apie atributą

Generatorius (klasė). Naudojant šią klasę iš pasirinktos aibės esybių vaizduojamų ER diagramoje (pagrindinė forma) yra sugeneruojamas SQL DDL aprašas kurį galima eksportuoti beveik į kiekvieną populiarią DBVS.

5.2.4 Duomenų bazės schema

CASE įrankio duomenų saugyklos schemos naudojamos loginių duomenų modelių sudarymui bei SQL DDL aprašų generavimui diagrama yra pateikiama 23 pav.



22 pav. CASE įrankio duomenų saugyklos fragmentas

5.2.5 Kokybės kriterijai

Atliekant architektūros projektavimą bus remiamasi šiais kriterijais :

- efektyvumas – kokių kompiuterio resursų reikia sistemai veikti;
- panaudojamumas – ar sistema pakankamai tiksliai interpretuoja vartotojo įvedamus duomenis, ar yra priimtina vartotojui;
- integralumas – ar sistema gali būti sujungta su kitais moduliais;
- teisingumas – ar sistema pilnai atitinka specifikaciją ir veikia logiškai;
- patikimumas – ar sistema yra tolerantiška klaidoms;
- pakartotinis panaudojimas – ar sistema yra nepriklausoma nuo operacinės sistemos, ar gali būti pakartotinai panaudoti jos komponentai;
- pernešamumas – ar sistema gali būti lengvai perkeliama į kitą platformą.

6 Duomenų bazės schemos SQL aprašo generavimo modulio eksperimentinis įvertinimas

Realizuotas CASE įrankio modulis ištirtas pagal architektūrinio projektavimo metu nustatytus kokybės kriterijus ir palygintas su panašaus pobūdžio sistema: *Rational Rose Enterprise Edition*.

6.1 Sukurtos sistemos kokybės tyrimas ir įvertinimas

Sukurtos sistemos funkcionalumo analizavimas, atitikimas funkcinei specifikacijai ir vartotojo įvertinimas ar jam tinka galutinis rezultatas – visa tai atliekama kokybės įvertinimo procese. Pagrindiniai kokybės vertinimo tikslai:

- Patikrinti, ar realizuota programų sistema atitinka reikalavimų specifikaciją. Neatitikimas reikalavimų specifikacijai gali nulemti, kad sukurta sistema nebus naudojama.
- Funkcionavimo, logikos klaidų aptikimas, kurios nebuvo rastos testavimo metu. Paieškos tikslas surasti dar nesurastas klaidas ir surinkti galimų papildymų siūlymus sistemos tolimesniam vystymui.

6.1.1 Kokybės vertinimo procesas

Peržiūros

Realizuotos sistemos kokybė buvo tikrinama įvairiais būdais. Vienas iš jų – peržiūros. Ši sistema buvo analizuota :

- Statiniu būdu. Projekto konsultantas peržiūrėjo sistemos programos išeities tekstą ieškodamas galimų funkcionalumo problemų.
- Interviu su užsakovu. Sistema buvo pristatoma užsakovui evoliuciniu modeliu, todėl jis dalyvavo sistemos projektavimo ir realizavimo etapuose. Sistema buvo keičiama pagal užsakovo pageidavimus.

Interviu su užsakovu

Sistema buvo kuriama evoliuciniu modeliu, todėl nuolat vyko interviu su užsakovu. Pirmiausia buvo surinkti kuriamos sistemos reikalavimai ir parašyta reikalavimų specifikacija. Po reikalavimų surinkimo pradėtas projektavimo ir realizavimo etapas. Pristačius užsakovui prototipą, kuriame buvo realizuotos pagrindinės reikalavimų dokumente specifikuotos funkcijos, buvo pareikštas pageidavimas papildyti naujomis funkcijomis:

- Diagramos sudarymo pratęsimas naujame lape;
- Rezultatų specifikavimą ir apdorojimą pateikti funkcijų hierarchijos diagramoje.

Taip pat buvo pateiktas pageidavimas pakeisti funkcijų specifikavimo elementų žymėjimą, labiau atitinkantį įprastinį šio elemento žymėjimą. Po pirmosios peržiūros buvo papildytas reikalavimų ir architektūros dokumentas. Sekančių peržiūrų metų nebuvo pateikti papildomi reikalavimai, tačiau pateikti pasiūlymai, kad sukurtas įrankis būtų patogesnis ir priimtinesnis sistemų analitikams.

Sistemos atitikimas specifikacijai

Buvo atliktas tikrinimas, ar sistema atitinka specifikacijoje pateiktus reikalavimus. Nebuvo rasta netenkinamų reikalavimų, todėl vertinama, kad sistema atitinka reikalavimų specifikaciją.

6.1.2 Vertinimo rezultatai

Architektūros dokumente apibrėžti kokybės reikalavimai yra įvertinti 5 lentelėje.

5 lentelė Programinės įrangos kokybės įvertinimas

Įvertinimas ***Kriterijus***	***l. aukštas***	***aukštas***	***vidutiniškas***	***žemas***	***l. žemas***
Efektyvumas	V				
Panaudojamumas			V		
Integralumas			V		
Teisingumas	V				
Patikimumas	V				
Palaikomumas	V				
Suprantamumas	V				
Pakartotinas panaudojimas		V			
Pernešamumas			V		

Paaiškinimai

- Pernešamumas – šis kriterijus įvertintas aukštai, nes sukurtas produktas yra realizuotas Microsoft .net technologijomis ir jį bus galima naudoti tik tokiose sistemose kuriose yra įdiegta Microsoft .net framework platforma..
- Integralumas įvertintas vidutiniškai, nes kiti CASE įrankio moduliai yra dar realizavimo fazėje ir buvo išbandytas darbas tik vienu realizuotu moduliu.
- Panaudojamumas įvertintas vidutiniškai, nes šis modulis dar bus tobulinimas, siekiant jį padaryti patogesniu vartotojui.

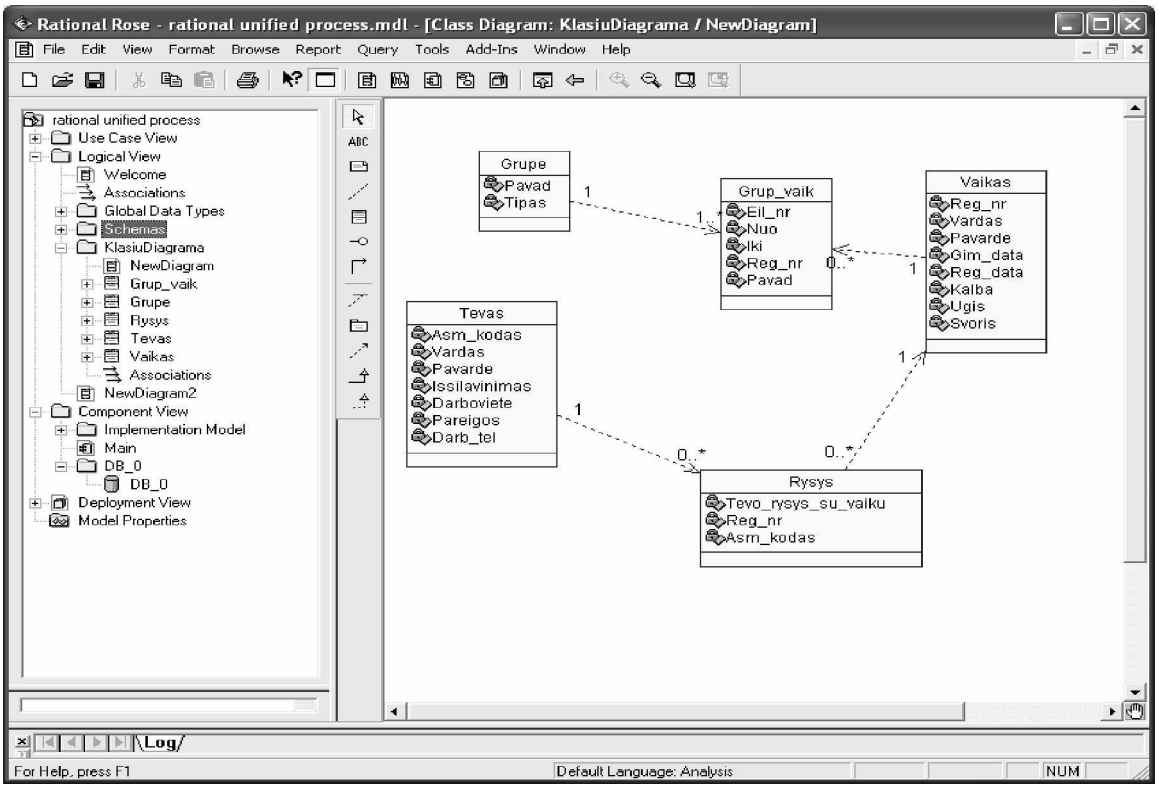
6.1.3 Rational Rose paketo eksperimentinis tyrimas

Unifikuotas Rational procesas (RUP) yra vienas iš labiausiai paplitusių programinės įrangos kūrimo metodų. RUP nuosekliai ir pakankamai formaliai aprašo visą programinės įrangos kūrimo procesą, pradedant veiklos modeliavimu, baigiant sukurtos sistemos pateikimu vartotojui. Modelių sudarymui naudojama UML notacija.

Rational Rose pakete, nėra galimybės sudaryti duomenų modelių iš funkcinių reikalavimų specifikacijos duomenų saugyklos. Taigi, norint sukurti SQL DDL aprašą reikia sudaryti duomenų modelius rankiniu būdu.

Kadangi duomenų modelius *Rational Rose* sistemoje galima kurti iš klasių diagramos arba rankiniu būdu, bus pateikiamas klasių diagramos sudarymo, klasių diagramos konvertavimo į loginį duomenų modelį, bei fizinės schemos generavimo fazės.

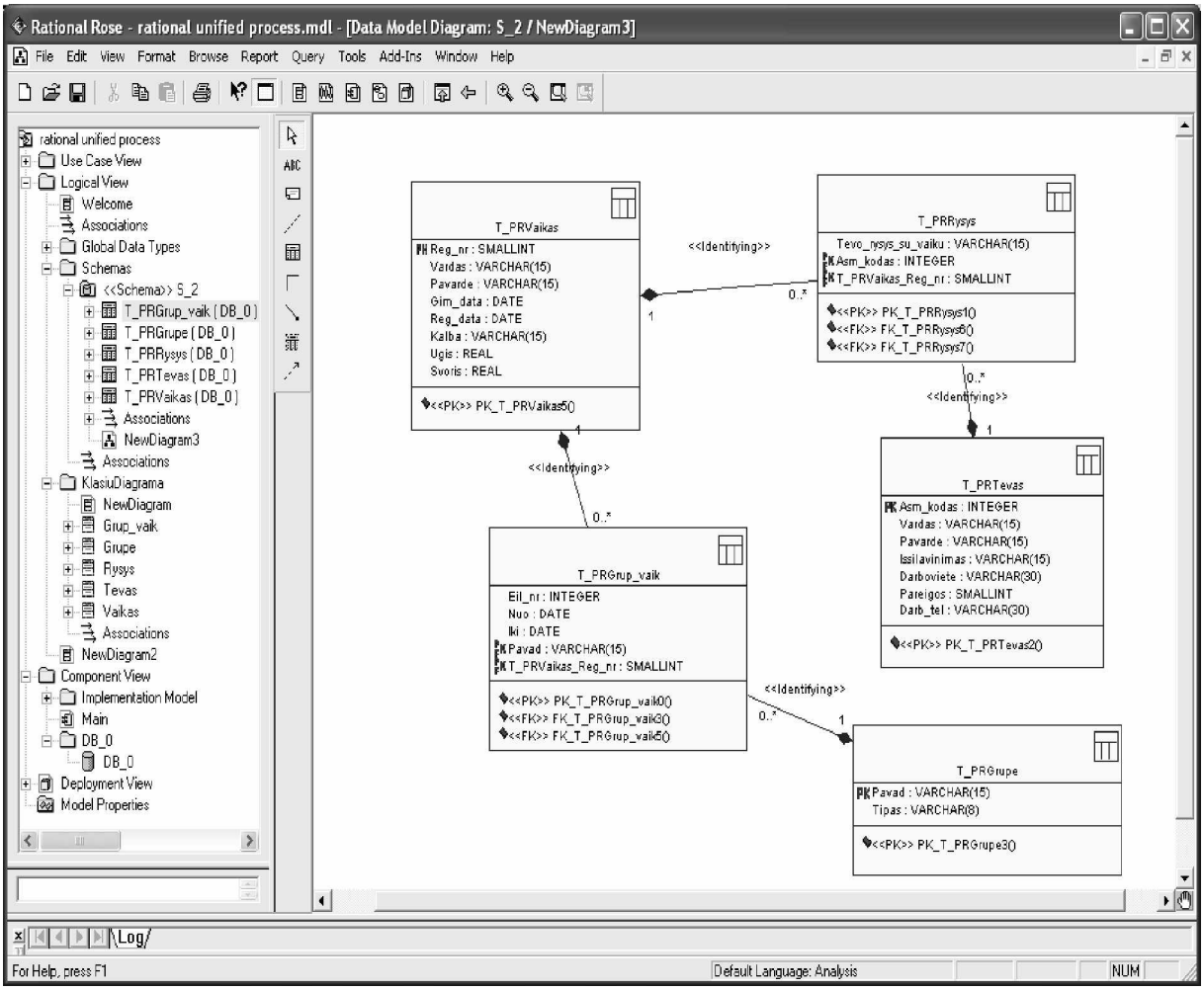
Loginiame vaizde sudaroma klasių diagrama. Klasių diagramos išsamumas gali svyruoti nuo apibendrinto iki detalaus (24 pav.), priklausomai nuo to kokią strategiją pasirenka duomenų modeliuotojas: ar klasių diagramą naudoti kaip konceptualųjį modelį ar kaip loginį modelį iš kurio bus generuojama SQL DDL schema.



23 pav. Duomenų modelio sudarymas naudojant klasių diagramą

Suformuotas klasių modelis transformuojamas į duomenų modelį, nurodžius ANSI SQL - 92 duomenų bazės schemą. Gaunamas duomenų modelis pateikiamas 25 pav. Klasių diagramai

konvertuoti į duomenų modelį iš kurio galima sugeneruoti SQL DDL aprašą reikia sudaryti schemą bei duomenų bazės specifikaciją.



24 pav. Duomenų modelis gautas konvertavus klasių diagramą

25 paveiksle pavaizduoto duomenų modelį konvertavus į fizinį duomenų bazės modelį gautas SQL DDL aprašas pateikiamas žemiau 26 pav.

```
CREATE TABLE T_PRGrup_vaik (
Eil_nr INTEGER NOT NULL,
Nuo DATE NOT NULL,
Iki DATE NOT NULL,
Pavad VARCHAR ( 15 ) NOT NULL,
T_PRVaikas_Reg_nr SMALLINT NOT NULL,
CONSTRAINT PK_T_PRGrup_vaik0 PRIMARY KEY (T_PRVaikas_Reg_nr, Pavad)
);
CREATE TABLE T_PRRysys (
Tevo_rysys_su_vaiku VARCHAR ( 15 ) NOT NULL,
Asm_kodas INTEGER NOT NULL,
T_PRVaikas_Reg_nr SMALLINT NOT NULL,
CONSTRAINT PK_T_PRRysys1 PRIMARY KEY (T_PRVaikas_Reg_nr, Asm_kodas)
);
```

```
REATE TABLE T_PRTevas (
Asm_kodas INTEGER NOT NULL,
Vardas VARCHAR ( 15 ) NOT NULL,
Pavarde VARCHAR ( 15 ) NOT NULL,
Issilavinimas VARCHAR ( 15 ) NOT NULL,
Darboviete VARCHAR ( 30 ) NOT NULL,
Pareigos SMALLINT NOT NULL,
Darb_tel VARCHAR ( 30 ) NOT NULL,
CONSTRAINT PK_T_PRTevas2 PRIMARY KEY (Asm_kodas)
);
CREATE TABLE T_PRGrupe (
Pavad VARCHAR ( 15 ) NOT NULL,
Tipas VARCHAR ( 8 ) NOT NULL,
CONSTRAINT PK_T_PRGrupe3 PRIMARY KEY (Pavad)
);
REATE TABLE T_PRVaikas (
Reg_nr SMALLINT NOT NULL,
Vardas VARCHAR ( 15 ) NOT NULL,
Pavarde VARCHAR ( 15 ) NOT NULL,
Gim_data DATE NOT NULL,
Reg_data DATE NOT NULL,
Kalba VARCHAR ( 15 ) NOT NULL,
Ugis REAL NOT NULL,
Svoris REAL NOT NULL,
CONSTRAINT PK_T_PRVaikas5 PRIMARY KEY (Reg_nr)
);
ALTER TABLE T_PRGrup_vaik ADD CONSTRAINT FK_T_PRGrup_vaik3 FOREIGN KEY (Pavad) REFERENCES T_PRGrupe (Pavad)
ON DELETE NO ACTION ON UPDATE NO ACTION;
ALTER TABLE T_PRGrup_vaik ADD CONSTRAINT FK_T_PRGrup_vaik5 FOREIGN KEY (T_PRVaikas_Reg_nr) REFERENCES
T_PRVaikas (Reg_nr)  ON DELETE NO ACTION ON UPDATE NO ACTION;
ALTER TABLE T_PRRysys ADD CONSTRAINT FK_T_PRRysys7 FOREIGN KEY (T_PRVaikas_Reg_nr) REFERENCES T_PRVaikas
(Reg_nr)  ON DELETE NO ACTION ON UPDATE NO ACTION;
ALTER TABLE T_PRRysys ADD CONSTRAINT FK_T_PRRysys6 FOREIGN KEY (Asm_kodas) REFERENCES T_PRTevas (Asm_kodas)
ON DELETE NO ACTION ON UPDATE NO ACTION;
```

25 pav. Rational data modeler sugeneruotas SQL DDL aprašas

6.1.4 Sukurto įrankio panaudojimas duomenų modelio sudarymui bei SQL DDL aprašo generavimui

Kadangi RUP metodo bei FRSM metodo duomenų modelio sudarymui reikalingi pradiniai duomenys skiriasi, RUP metodą realizuojančiame įrankyje būtina duomenis paruošti rankiniu būdu, o FRSM metodo loginio duomenų modelio vaizdavimo bei SQL DDL aprašo fazę realizuojančiam CASE įrankio moduliui reikalingi duomenys yra paruošiami per eilę FRSM fazių ir saugomi metabazėje. 6- 8 lentelėse pateikiami loginį duomenų modelį aprašantys duomenys kuriuos naudos SQL DDL aprašą generuojantis CASE įrankio modulis.

6 lentelė Loginio duomenų modelio atributai

Atributas								
id	**kodas**	**n_vardas**	**raktas**	**unikalumas**	**paaiskinimas**	**tipas**	**ilgis**	**e_id**
1	a1	Reg_nr	True	True	vaiko registracijos numeris	integer		1
2	a2	Vardas	False	False	vaiko vardas	text	15	1
3	a3	Pavarde	False	False	vaiko pavarde	text	15	1
4	a4	Gim_data	False	False	vaiko gimimo data	datetime		1
5	a5	Reg_data	False	False	nuo kada vaikas pradejo lankyti darzeli	datetime		1
6	a6	Kalba	False	False	kalba, kuria vaikas kalba namuose	text	15	1
7	a7	Ugis	False	False	vaiko ugis	Real		1
8	a8	Svoris	False	False	vaiko svoris	Real		1
9	a9	Asm_kodas	True	True	tevo/globejo asmens kodas	integer		2
10	a10	Vardas	False	False	tevo/globejo vardas	text	15	2
11	a11	Pavarde	False	False	tevo/globejo pavarde	text	15	2
12	a12	Issilavinimas	False	False	tevo/globejo issilavinimas	text	15	2
13	a13	Darboviete	False	False	tevo/globejo darboviete	text	30	2
14	a14	Pareigos	False	False	tevo/globejo pareigos darbovieteje	text	15	2
15	a15	Darb_tel	False	False	tevo/globejo darbovietes telefonas	text	15	2
16	a16	Tevo_rysys_su_vaiku	False	False	parodo, kokiu rysiu tevas/globejas susijes su vaiku, pvz., tevas, motina, itevis, imote ir pan.	text	15	3
19	a19	Pavad	True	True	lopselio/darzelio grupes pavadinimas	text	15	5
20	a20	Tipas	False	False	parodo, ar grupe priklauso lopseliui, ar darzeliui	text	8	5
21	a21	Eil_nr	False	False	parodo vaiko eiles numeri lankytos/lankomos grupes zurnale	integer		6
22	a22	Nuo	False	False	parodo, nuo kada vaikas lanke/lanko grupe	datetime		6

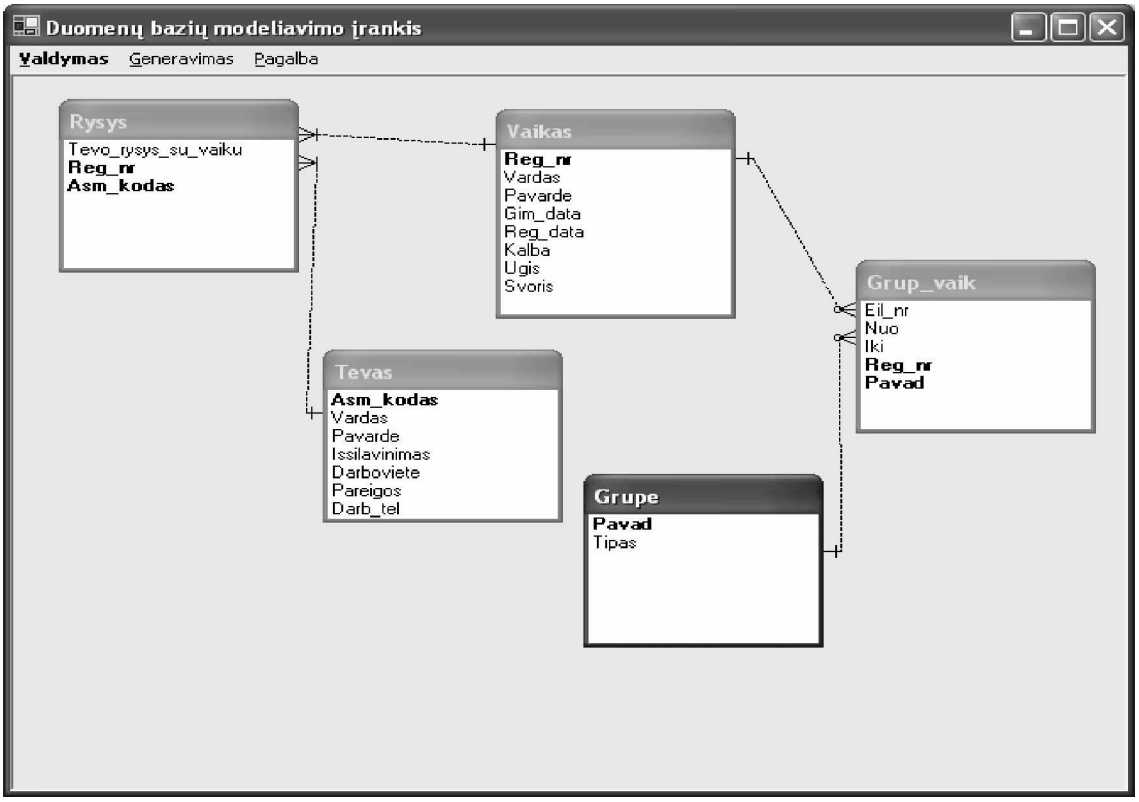
7 lentelė Loginio duomenų modelio esybės

Esybe			
id	**kodas**	**n_vardas**	**Paaiskinimas**
1	e1	Vaikas	asmeniniai duomenys apie vaika
2	e2	Tevas	duomenys apie vaiko teva ar globeja
3	e3	Rysys	parodo vaiko rysi su tevu
5	e5	Grupe	duomenys apie lopselio/darzelio grupe
6	e6	Grup_vaik	parodo, kokias lopselio/darzelio grupes lanke/lanko vaikas; parodo, kurie vaikai lanke/lanko tam tikra grupe

8 lentelė Loginio duomenų modelio ryšiai tarp esybių

Rysys					
id	**kard1**	**kard2**	**e_id1**	**e_id2**	**identifikuojantis**
13	1,x	1,1	1	3	True
16	0,x	1,1	1	6	True
23	1,x	1,1	2	3	True
56	0,x	1,1	5	6	True

Naudojant sukurtą CASE įrankio modulį, loginis duomenų modelis saugomas CASE įrankio duomenų saugykloje, pavaizduojamas loginiu duomenų modeliu (27 pav.), kurį galima redaguoti bei jį konvertuoti į SQL DDL aprašą (28 pav.).



26 pav. Loginio duomenų modelio pavaizdavimas naudojant realizuotą CASE įrankio modulį

```
CREATE TABLE Vaikas (
                Reg_nr INTEGER  NOT NULL  ,
                Vardas VARCHAR ( 15 ) NOT NULL  ,
                Pavarde VARCHAR ( 15 ) NOT NULL  ,
                Gim_data DATETIME  NOT NULL  ,
                Reg_data DATETIME  NOT NULL  ,
                Kalba VARCHAR ( 15 ) ,
                Ugis DECIMAL  ,
                Svoris DECIMAL  ,
                CONSTRAINT TC_Vaikas UNIQUE ( [Reg_nr] ),
                CONSTRAINT PC_Vaikas PRIMARY KEY ([Reg_nr]));
CREATE TABLE Tevas (
                Asm_kodas INTEGER  NOT NULL  ,
                Vardas VARCHAR ( 15 ) NOT NULL  ,
                Pavarde VARCHAR ( 15 ) NOT NULL  ,
                Issilavinimas VARCHAR ( 15 ) NOT NULL  ,
                Darboviete VARCHAR ( 30 ) ,
                Pareigos VARCHAR ( 15 ) ,
                Darb_tel VARCHAR ( 15 ) ,
                CONSTRAINT TC_Tevas UNIQUE ( [Asm_kodas] ),
                 CONSTRAINT PC_Tevas PRIMARY KEY ([Asm_kodas]));
CREATE TABLE Rysys (
                Tevo_rysys_su_vaiku VARCHAR ( 15 ) NOT NULL  ,
                Reg_nr INTEGER  NOT NULL  ,
```

```
                Asm_kodas INTEGER  NOT NULL  ,
                CONSTRAINT TC_Rysys UNIQUE ( [Reg_nr] ),
                CONSTRAINT TC_Rysys1 UNIQUE ( [Asm_kodas] ),
                 CONSTRAINT PC_Rysys PRIMARY KEY ([Reg_nr],[Asm_kodas]));
CREATE TABLE Gyvenamoji_vieta (
                Namu_adr VARCHAR ( 30 ) NOT NULL  ,
                Namu_tel VARCHAR ( 15 ) ,
                Reg_nr INTEGER  NOT NULL  ,
                CONSTRAINT TC_Gyvenamoji_vieta UNIQUE ( [Namu_adr] ),
                CONSTRAINT TC_Gyvenamoji_vieta1 UNIQUE ( [Reg_nr] ),
                 CONSTRAINT PC_Gyvenamoji_vieta PRIMARY KEY ([Namu_adr],[Reg_nr]));
CREATE TABLE Grupe (
                Pavad VARCHAR ( 15 ) NOT NULL  ,
                Tipas VARCHAR ( 8 ) NOT NULL  ,
                CONSTRAINT TC_Grupe UNIQUE ( Pavad ),
                 CONSTRAINT PC_Grupe PRIMARY KEY (Pavad));
CREATE TABLE Grup_vaik (
                Eil_nr INTEGER  NOT NULL  ,
                Nuo DATETIME  NOT NULL  ,
                Iki DATETIME  ,
                Reg_nr INTEGER  NOT NULL  ,
                Pavad VARCHAR ( 15 ) NOT NULL  ,
                CONSTRAINT TC_Grup_vaik UNIQUE ( [Reg_nr] ),
                CONSTRAINT TC_Grup_vaik1 UNIQUE ( Pavad ),
                 CONSTRAINT PC_Grup_vaik PRIMARY KEY ([Reg_nr],Pavad));
ALTER TABLE Rysys ADD CONSTRAINT FK_Rysys FOREIGN KEY ([Reg_nr]) REFERENCES Vaikas([Reg_nr])
ON DELETE NO ACTION ON UPDATE NO ACTION;
ALTER TABLE Rysys ADD CONSTRAINT FK_Rysys1 FOREIGN KEY ([Asm_kodas]) REFERENCES
Tevas([Asm_kodas]) ON DELETE NO ACTION ON UPDATE NO ACTION;
ALTER TABLE Gyvenamoji_vieta ADD CONSTRAINT FK_Gyvenamoji_vieta FOREIGN KEY ([Reg_nr])
REFERENCES Vaikas([Reg_nr]) ON DELETE NO ACTION ON UPDATE NO ACTION;
ALTER TABLE Grup_vaik ADD CONSTRAINT FK_Grup_vaik FOREIGN KEY ([Reg_nr]) REFERENCES
Vaikas([Reg_nr]) ON DELETE NO ACTION ON UPDATE NO ACTION;
ALTER TABLE Grup_vaik ADD CONSTRAINT FK_Grup_vaik1 FOREIGN KEY (Pavad) REFERENCES
Grupe(Pavad) ON DELETE NO ACTION ON UPDATE NO ACTION;
```

27 pav. Suformuotas SQL DDL aprašas

6.1.5 Eksperimentinio tyrimo įvertinimas

Palyginus sugeneruotus SQL DDL aprašus, pastebėta, jog realizuotas CASE įrankio modulis aprašo pagrindinius SQL DDL schemos elementus bei yra korektiškas.

Realizuotas duomenų modelio vaizdavimo bei SQL DDL aprašo generavimo modelis nėra dar tiek išvystytas kaip komerciniai tokio pobūdžio paketai, kaip *Rational Rose* ar panašūs, nes tai tik prototipas. Produktas nėra toks imlus resursams kaip komerciniai tokio pobūdžio paketai, jis yra skirtas tiesioginiam darbui su informacijos srautų specifikacijos saugykla, t.y. metaduomenų baze. Duomenų loginiams modeliams pavaizduoti naudojama pakankamai išsami ir informatyvi notacija.

Atliktas eksperimentas įrodė, kad FRSM metodas gali buti realizuotas taip, kad padengtu visas IS kūrimo fazes nuo reikalavimu surinkimo ir specifikavimo iki IS projektavimo ir realizavimo. Sukurto įrankio

7 Išvados

1. Išanalizuoti *konceptualus, loginis, fizinis ir reliacinis* duomenų modeliai, jų sudarymo koncepcijos, transformavimas iš vieno modelio į kitą. Aprašytas esybių ryšių modelis bei palygintos jo notacijos. Išanalizuoti metodai duomenų modeliams sudaryti įvairiose metodologijose, aptarti jų privalumai ir trūkumai, į kuriuos buvo atsižvelgta sudarant projektuojamos IS loginio duomenų modelio pavaizdavimą esybių ryšių diagrama bei generuojant fizinę duomenų bazės schemą.
2. Pristatytas projektuojamos IS loginio duomenų modelio analizavimo, atvaizdavimo esybių ryšių diagrama (informacijos inžinerijos notacijoje) bei SQL DDL aprašo (fizinės duomenų bazės schemos) generavimo procesas funkcinių reikalavimų specifikacijos pagrindu. Šis procesas realizuotas, kaip FRSM CASE įrankio prototipo modulis.
3. Funkcinių reikalavimų specifikavimo metodo metamodelis papildytas loginių duomenų modelių *fragmentų saugyklos* lentelėmis, taip sukuriant galimybę išsaugoti ir prireikus atstatyti tam tikrus projektuojamos IS loginio duomenų modelio fragmentus bei išplečiant kuriamo CASE įrankio modulio funkcionalumą.
4. Realizuojant CASE įrankio modulį, pagal pasiūlytą procesą, buvo įvykdyti ir dokumentuoti būtini programinės įrangos kūrimo etapai, sudarant reikalavimų, architektūros, detalios architektūros specifikacijas ir vartotojo vadovą.
5. Realizuoto CASE įrankio modulis sukurtas panaudojant Microsoft .NET technologijas – MS *ADO.NET*, MS *Windows Forms .NET* programuojant C# programavimo kalba. Šis modulis leidžia analizuoti projektuojamos IS duomenų modelį, saugomą CASE įrankio duomenų saugykloje, kuriai realizuoti buvo panaudota DBVS MS Access. Duomenų modeliis ar jo fragmentas atvaizduojamas esybių ryšių diagrama bei eksportuojamas į SQL DDL aprašą saugančią bylą.
6. Sugeneruoto duomenų bazės modelio SQL DDL aprašo teisingumas bei pilnumas buvo patikrinti eksperimentiškai. Sugeneruotas SQL DDL aprašas sėkmingai importuotas į MS SQL DBVS, taip sukuriant projektuojamos IS duomenų bazę. Sistemos funkcionalumas palygintas su panašaus pobūdžio sistema *Rational rose data modeler* bei jos suformuotais rezultatais. Suformuotas duomenų bazės reliacinės schemos aprašas gali būti importuotas į įvairaus tipo populiarias reliacines duomenų bazių valdymo sistemas kadangi jis nėra susietas su konkrečia DBVS. Realizuotas programinis CASE įrankio modulis užbaigia nuosekliai formuotą IS kūrimo fazę.

8 LITERATŪRA

[1] Codd E. F. A relational model for large shared data banks. Communications of the ACM, 13(6):377–387, 1970.

[2] Chen P. P. The entity-relationship model - towards a unified view of data. ACM Transactions on Database Systems, 1(1):9–36, March 1976.

[3] Barbara A. Carkenord Why Build A Logical Data Model. [žiurėta 2005.01.15] Prieiga per internetą: http://www.embarcadero.com/resources/tech_papers/datamodel.pdf

[4] Poolet. M. A. Data Modeling. SQL Server Magazine. [žiūrėta 2005.03.15] Prieiga per internetą http://www.windowsitpro.com/Articles/Index.cfm?ArticleID=8241&DisplayTab=Article

[5] Paradauskas B., Nemuraitė L. Duomenų bazės ir semantiniai modeliai : monografija. KTU – Kaunas: Technologija, 2002.

[6] Bagui S., **Earp R.** Database Design Using Entity-Relationship Diagrams. Addison-wesley, 2004

[7] Butkienė R. Informacijos sistemai keliamų funkcinių reikalavimų specifikavimo metodas: daktaro disertacija. KTU – Kaunas: Technologija, 2002.

[8] Butkienė R., Butleris R. The Approach for the User Requirements Specification. 5th East-European conference ADBIS'2001, Research Communications, Ed. by A.Čaplinskas, J.Eder. Vilnius, 2001, p.p. 225-240.

[9] Barker R., Longman C.,. Case Method, function and process modelling. Addison-wesley, 1992.

[10] Butkienė R., Informacijos sistemų projektavimas Oracle Designer/2000 priemonėmis. Kaunas:Technologija, 1998.

[11] Hay D. C. A COMPARISON OF DATA MODELING TECHNIQUES Essential Strategies, Inc. October, 1999.

[12] Aleksandravičienė A., Danikauskas T., Butleris R., Šidlauskas K. Duomenų modelio automatizuoto sudarymo prototipo funkcionalumo tyrimas. Informacijos mokslai. Vilnius 2005, p. 118- 12

[13] Danikauskas T., Butleris R. Conceptual Data Model Design Using Functional Requirements Specification Methods. *CAiSE'04 Workshop Evaluating Modeling for Systems Analysis and Design* (EMMSAD'04): CAiSE'04 Workshops Proceedings. Ryga, 2004 vol. 1, p. 221-232.

[14] Maciaszek L. A., Requirements analysis and system design developing information systems with UML. Addison-wesley, 2001.

[15] Kekys A. Duomenų bazės schemos SQL aprašo generavimas funkcinių reikalavimų specifikacijos pagrindu Iš: Informacinės technologijos 2005: konferencijos pranešimų medžiaga, Kaunas 2005 balandžio 29 d.. Kaunas, 2005, p. 142-145. Informacinės technologijos 2005

[16] Muller P. Instant UML. Wrox press Ltd.,1997.

[17] Kruchten P., Rational Unified Process—An Introduction, Addison-Wesley, 1999.

[18] Integration Definition For Function Modeling (Idef1X). Draft Federal InformationProcessing Standards Publication 184, 1993 December 21.

[19] Miller J. Visio Modeling Engine Wiki. [žiūrėta 2004.08.15] Prieiga per internetą http://www.pdata.com/VME/VisioModelingEngine.htm

9 Abstract

The data modeling phase is one of the main phases in automated system development methods. Usually data modeling is separated as independent process which success and robustness depends largely on the intuition and experience of the data analyst. This can become problematic when IS database is being designed for large scale IS.

To solve this type of problems it's proposed to accomplish data modeling phase not as a separate, poorly with the generic modeling process connected part but as one of the fundamental system development phases. Functional Requirements Specification Method is proposed for this matter. This methods main idea is to build database model according to user requirements for the functionality of the system being developed.

The purpose of this work is to present a process intended for building a entity relationship diagram from the analysed functional requirements specification metamodel and generation of the SQL DDL script (relational schema) from that model. The presented model will be implemented as one of the CASE tool modules specified for the database modeling and SQL DDL script generation. Implemented tool will be verified and it's correctness will be experimentally approved.

10 Priedai

10.1 Realizuotos programos vartotojo vadovas

1. Sistemos funkcinis aprašymas

1.1. Paskirtis

Realizuotas programinis įrankis – vienas CASE įrankio komponentų. Šis įrankis skirtas duomenų bazės analitikui ar analitiko darbą atliekančiam asmeniui, suinteresuotam korektiškos duomenų saugyklos projektuojamai PĮ sukūrimu. Įrankio pagrindinės funkcijos yra duomenų meta modelio nuskaitymas iš funkcinių reikalavimų specifikacijos metodu grįsto CASE įrankio duomenų saugyklos bei nuskaitytų duomenų transliavimas į loginį duomenų modelį vaizduojamą ER diagrama, įvairios duomenų loginio modelio modifikacijos bei reliacinio duomenų modelio SQL DDL (data definition language) (t.y. fizinio duomenų modelio) sukūrimą.

1.2. Galimybės

- Analizuoti funkcinių reikalavimų specifikacijos pagrindu sukurtą, projektuojamos PĮ duomenų bazės meta modelį;
- Vaizduoti iš CASE įrankio duomenų saugyklos importuotą meta modelį pasirinktu detalumo lygiu, loginio duomenų bazės modelio vaizdavimui naudojant ER diagramą bei *informacijos inžinerijos* notaciją :
    - § Nurodyti tik pageidaujamas nagrinėti probleminės srities esybes bei ryšius tarp tų esybių, loginiame duomenų modelyje;
    - § Nurodyti tik pageidaujamas nagrinėti probleminės srities esybes, bei toms esybėms giminingas esybes (pavyzdžiui vaizduojant esybes tris lygius gilyn) ir ryšius tarp jų;
    - § Vaizduoti ir analizuoti visas probleminės srities esybes, bei ryšius tarp tų esybių, loginiame duomenų modelyje.
- Peržiūrint ir analizuojant duomenų bazės loginį modelį, įvesti modifikacijas iš CASE įrankio duomenų saugyklos įkeltam duomenų bazės modeliui:
    - § Specifikuoti atributų būtinumo apribojimus;
    - § Specifikuoti atributų duomenų tipą bei atributų ilgius;
    - § Specifikuoti atributų unikalumo apribojimus;
    - § Keisti ryšių kaardinalumo apribojimus abiejose ryšio pusėse;

- § Keisti ryšių daugialypiškumo apribojimus abiejose ryšio pusėse;
- § Nurodyti ryšio stiprumo apribojimą.

- Iš apdoroto ar ne duomenų modelio vaizdo generuoti SQL DDL (reliacinį duomenų modelį) į fizinį failą kurį galima importuoti į kiekvieną reliacinę DBVS;
- Pilnam duomenų modeliui importuotam iš CASE įrankio duomenų saugyklos sukurti SQL DDL reliacinį modelį į fizinį failą, kurį galima importuoti į kiekvieną reliacinę DBVS.,

2. Vartotojo atmintinė

2.1. Reikalavimai vartotojui

Įrankis skirtas vieno iš PĮ sistemos projektavimo etapų. Konkrečiau – duomenų modeliavimui bei reliacinės duomenų bazės fizinės schemos SQL DDL aprašo generavimui.

Taigi, šio produkto vartotojas privalo turėti tam tikrų apdorojamos srities žinių. Vartotojas turi būti žmogus turintis bent bazines žinias informacinių technologijų srityje, būti susipažinęs su CASE metodais, bei išmanyti reliacines duomenų bazes bei pagrindinius šios probleminės srities terminus. Šios žinios reikalingos fizinės duomenų bazės kūrimo procesui suprasti, ER diagramų (duomenų loginio modelio) analizei bei modifikacijoms.

2.2. Bendras naudojimasis įrankiu

- Atidaromas duomenų saugyklos failas. CASE įrankio duomenų saugykla saugoma Microsoft Access duomenų bazėje, todėl atidarant duomenų saugyklą, iš visų failų sąrašo išfiltruojami failai turintys **.mdb* išplėtimą. Jei pasirenkamas neteisingas duomenų saugyklos failas jis nėra atidaromas.
- Analizuojamas duomenų saugykloje esantis duomenų modelis. Šio proceso metu atliekami duomenų meta modelio nuskaitymo ir interpretavimo veiksmai, tam tikrus duomenų saugyklos konstruktus perkeliant į programos atmintį.
- Vartotojas specifikuoja kokiu detalumo lygiu norės analizuoti importuotą duomenų modelį. Iš sąrašo pasirenkamos importuotos esybės bei nurodomas požymis ar su pasirinktomis esybėmis būtina kartu vaizduoti ir su jomis reliaciniais ryšiais susijusias esybes.
- Duomenų loginis modelis pateikiamas ER diagrama, naudojant *informacijos inžinerijos* notaciją (crows feet).
- Vartotojas gali modifikuoti atributų esybėse parametrus formoje iššaukiamoje du kartus spragtelėjus atributą esybėje. Taipogi , vartotojas gali keisti ryšių tarp esybių savybes atidaromoje formoje, kai du kartus spragtelimas ryšys tarp esybių.

- Vaizduojamą ER diagramą, kaip ir visą analizavimo metu importuotą duomenų modelį galima išsaugoti reliacinės duomenų bazės schemos SQL DDL aprašu fiziniame faile kuris importuojamas į DBVS palaikančią ANSI SQL.

Pastaba : detalų naudojimosi įrankiu aprašą ir galimų klaidų sąrašus rasite 3 skyriuje, kuriame detalizuojamos įrankio atliekamos f-jos.

3. Detali sistemos atmintinė

3.1. Pagrindinės programos struktūra

Duomenų modelio saugomo CASE saugykloje apdorojimui paleidžiamas failas Modeler.exe (failas būtinai turi būti saugomas direktorijoje kartu su dviem *dll* bibliotekomis : *Valdymomodulis.dll* ir *DBModulis.dll*). Atidarytame pagrindiniame programos lange Rodomas pagrindinis meniu, kurio pagalba valdomi programos veiksmai. Meniu sudarytas iš meniu *Valdymas, Generavimas, Pagalba* meniu punktų (1 pav.).

1 pav. Pagrindinis programos langas

Pagrindiniame meniu, kaip pavaizduota paveikslėlyje, tam tikri meniu punktai yra uždrausti, jei nėra įvykdytos tam tikros operacijos. Pavyzdžiui generavimo meniu punktas bus uždraustas tol, kol į programą nebus importuotas duomenų saugykloje saugomas meta modelis. Žemiau pateikiamas sąrašas visų meniu punktų sąrašas :

- Valdymas

  Skirtas veiksmams su meta modeliu koordinuoti bei ER diagramai valdyti

  - Pasirinkti duomenų bazę

    Pasirenkama CASE įrankio duomenų saugykla;

  - Nagrinėti meta modelį

    Nagrinėjamas saugykloje esantis duomenų meta modelis;

- Vaizdavimas
  - Vaizdavimo nustatymai

    Specifikuojamas ER diagramos detalumo lygis
  - Vaizduoti ER modelį

    ER modelio vaizdavimopožymis
- Išeiti

  Indikuojama darbo su produktu pabaiga.

- Generavimas

  Skirtas veiksmams su reliaciniu duomenų modeliu koordinuoti
  - Generuoti DDL aprašą iš diagramos

    Generuoja tik vaizduojamos ER diagramos SQL DDL aprašą
  - Generuoti pilną DDL aprašą

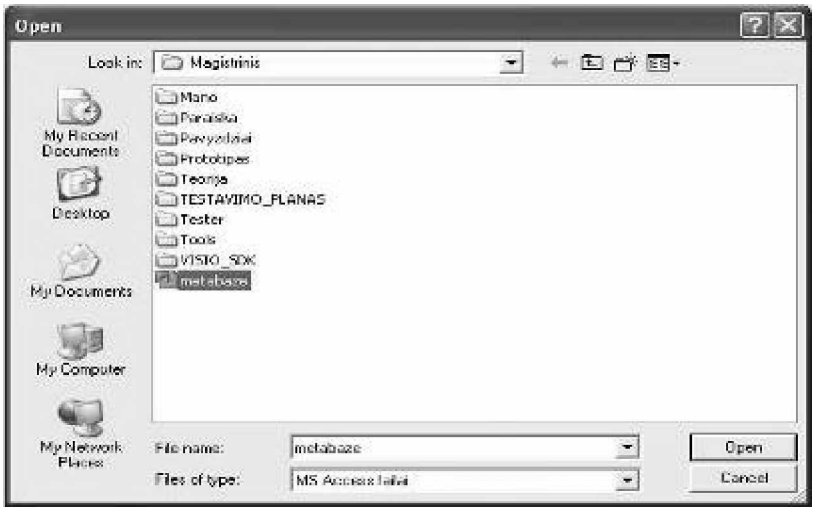
    Generuoja SQL DDL aprašą iš viso išanalizuoto duomenų modelio.
- Pagalba
  - Apie

    Informacija apie programos versiją bei atlikėją.

Šiame projekte realizuotas duomenų loginio modelio sudarymo bei jo konvertavimas į reliacinį duomenų modelio SQL DDL aprašą. Kadangi duomenys į sistemą patenka iš CASE įrankio duomenų saugyklos, būtina pasirinkti tokią saugyklą kurioje yra saugomas ankstesnėse PĮ kūrimo fazėse specifikuotas duomenų meta modelis.

3.2. Apdorojamos duomenų saugyklos išsirinkimas

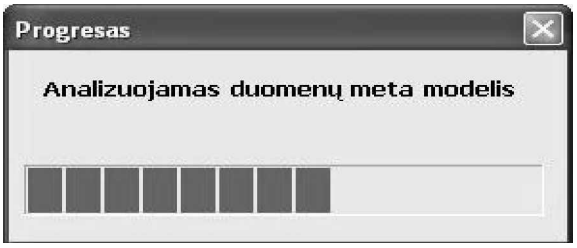
CASE įrankio duomenų saugykla atidaroma pasirinkus meniu punktą (pagrindinėje programos meniu juostoje) *Vaizdavimas -> Pasirinkti duomenų bazę,* arba tiesiog nuspaudus klavišų kombinaciją *Ctrl+A* atidaromas standartinis *Windows* aplinkos meniu kuriame pasirenkamas duomenų saugyklos failas (2 pav.).



2 pav. Standartinis failo pasirinkimo dialogas

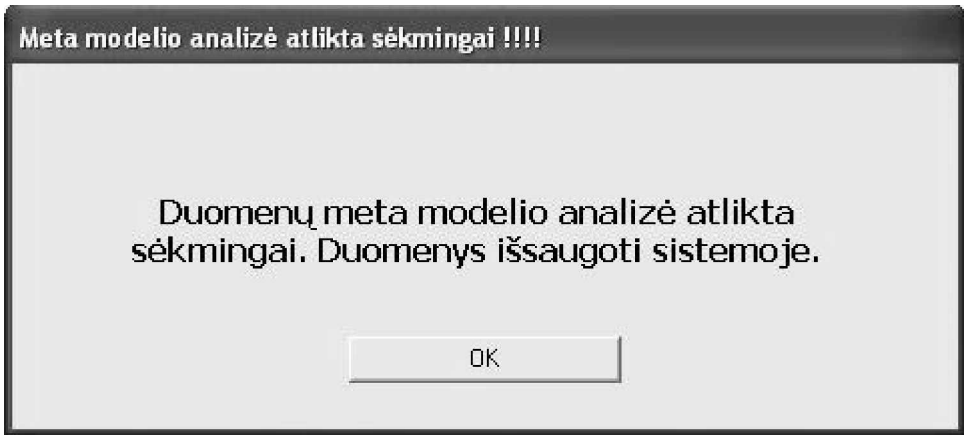
3.3. Duomenų meta modelio analizavimas

Pasirinkus duomenų saugyklos failą, galima duomenis iš duomenų saugyklos perkelti į kuriamą modulį. Šis procesas atliekamas pasirinkus meniu punktą *Valdymas -> Nagrinėti metamodelį* arba nuspaudus klavišų kombinaciją *Ctrl+N*. Analizuojant duomenų meta modelį ekrane matomas proceso eigos valdiklis (3 pav.)



3 pav. Analizuojamo duomenų metamodelio progresas

Duomenų modelio analizavimo proceso pabaigoje vartotojas informuojamas apie sėkmingą duomenų nuskaitymą iš duomenų saugyklos (4 Pav.)



4 Pav. Vartotojo informavimas apie sėkmingą veiksmų atlikimą

3.4. Duomenų modelio vaizdavimo nustatymai

Į sistemą įkėlus išanalizuota duomenų meta modelį, vartotojas gali pasirinkti pageidaujamas esybes kurias norėtų matyti ER diagramoje. Tai galima padaryti pasirinkus meniu punktą *Valdymas -> Vaizdavimas -> Vaizdavimo nustatymai...* Iššaukus šį meniu punktą yra iškviečiama Windows forma, kurios pagalba galima specifikuoti vaizdavimo detales (5 pav.)

5 pav. Vaizdo nustatymo forma

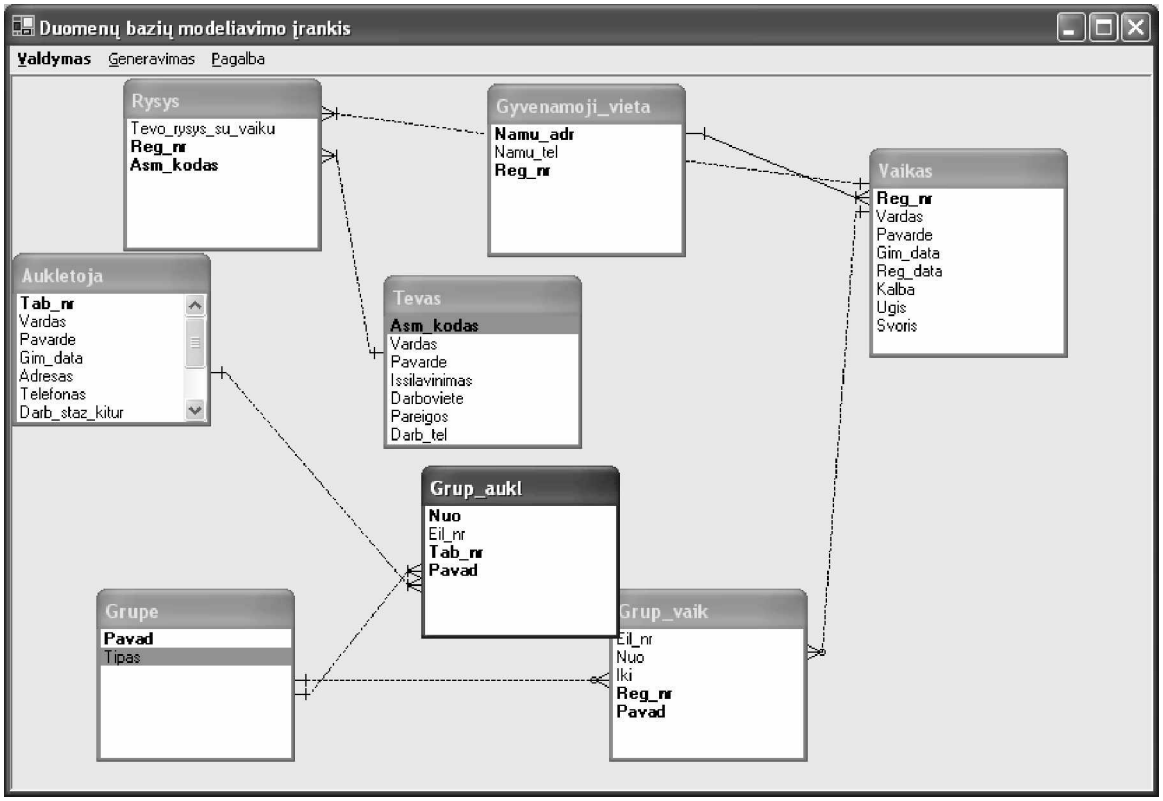
Vaizdavimo detalių formoje, diagramoje vaizduojamas esybes galima pasirinkti „*Vaizduojamos esybės"* detalių lape kuriame yra pateikiamas sąrašas visų iš CASE duomenų saugyklos importuotų esybių sąrašas. Galima rinktis esybes po vieną, arba pasirinkti vaizduoti visas esybes nuspaudus funkcinį klavišą „Žymėti visus". Nuspaudus mygtuką „Žymėti visus" jo pavadinimas keičiasi į „*Invertuoti"* ir paspaudus jį visos pažymėtos esybės pakeičiamos nežymėta požymiu.
Be galimybės pasirinkti norimas vaizduoti esybes, galima nustatyti požymius , ar bus vaizduojamos su pažymėtomis esybėmis susijusios esybės bei kiek „lygių į gylį" bus analizuojami sąryšiai (6 pav.) :

6 Pav. Papildomi nustatymai duomenų modeliui

Jei *Vaizdavimo Nustatymų* formoje pasirenkama bent viena esybė, nuspaudus mygtuką „*Gerai"* į pagrindinį programos ekraną išvedamas duomenų loginis modelis ER diagramos pavidalu.

3.5. Esybių – ryšių modelis

Nurodžius pageidaujamus vaizdavimo nustatymus programos pagrindiniame ekrane pavaizduojama esybių ryšių diagrama, detalizuojanti iš CASE duomenų saugyklos importuotą loginį duomenų modelį (7 pav.)



7 pav. Bendras programos vaizdas į ekraną išvedus ER modelį

3.5.1. Esybės ER modelyje

Vaizduojamame ER modelyje Esybės (8 pav.) yra vaizduojamos stačiakampiais, o atributai tekstinių eilučių sąrašu stačiakampio viduje. Kiekvieno esybę vaizduojančio stačiakampio viršuje (mėlynoje juostoje) yra išvedamas esybės pavadinimas. Taip, be didesnio vargo galima lengvai identifikuoti kiekvieną analizuojamą esybę.

8 Pav. Esybės vaizdas

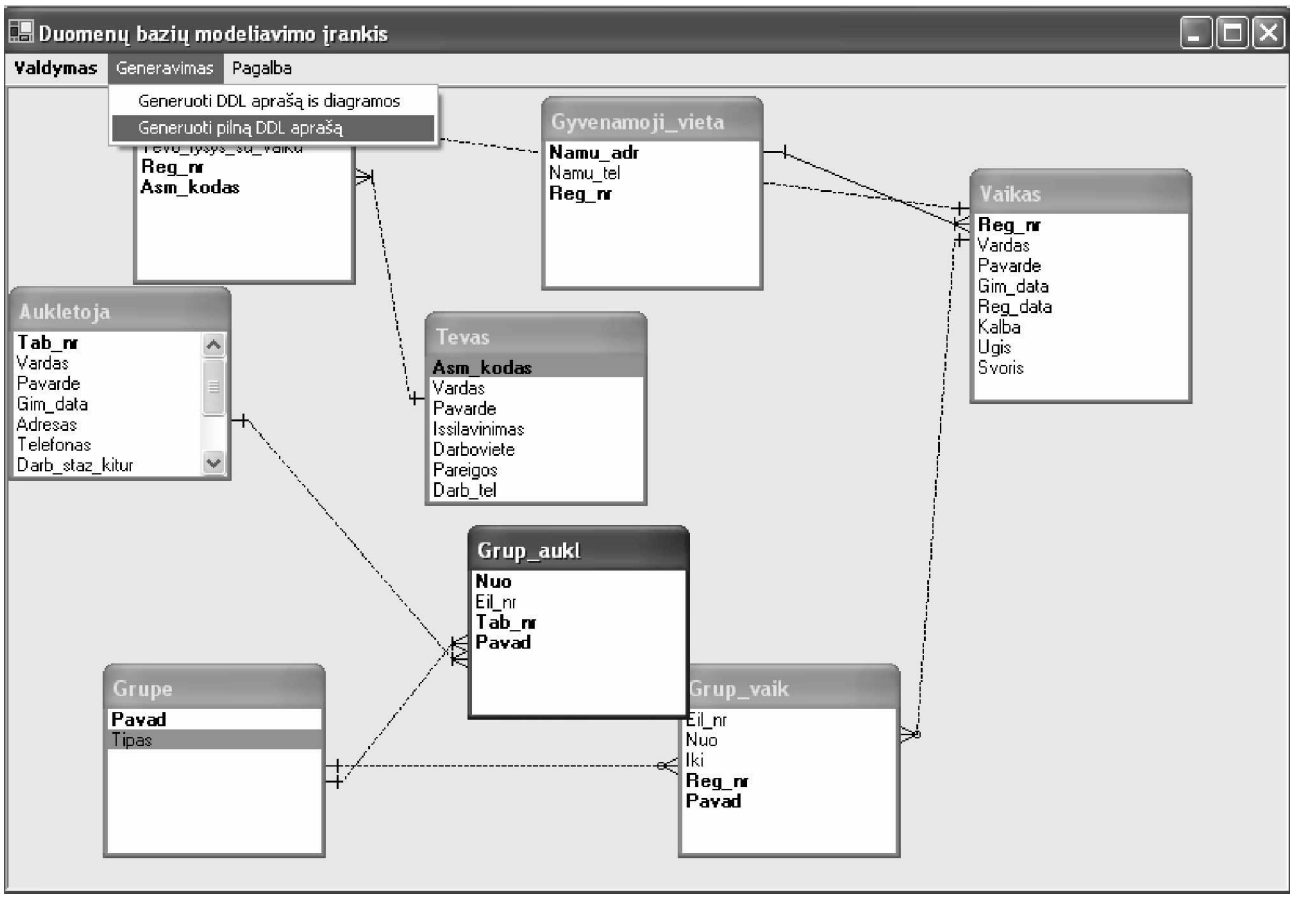
Kiekvienas raktinis esybės atributas (ar tai būtų išorinis identifikuojantis raktas ar pirminis esybės raktas) yra vaizduojamas pastorintu šriftu. Taigi, vizualiai labai lengva atskirti kuris atributas yra raktinis kuris „paprastas“ esybės atributas. Kiekvieną atributą galima detalizuoti du kartus spragtelėjus jį kairiu pelės klavišu. Vieną kartą spustelėjus atributo fono spalva pasikeičia į šviesiai melsvą, o spustelėjus atributą du kartus atidaroma detalizuoto atributo vaizdavimo forma. Po atributų sąrašą, pažymėjus esybę, galima naviguoti ir klaviatūros rodyklių pagalba („aukštyn“ ir „žemyn“).

3.5.2 Ryšiai ir jų savybės ER modelyje

Vaizduojamos esybės tarpusavyje yra siejamos *informacijos inžinerijos* notacijoje apibrėžtais ryšiais. Jei ryšys yra identifikuojantis jis yra punktyrinis, jei ryšio kaardinalumas „daug“ tai linija ties rišama esybe yra išskaidoma į tris ir t.t. Peržiūrint esybių ryšių modelį aprašomame įrankyje, informacija apie ryšį siejantį dvi esybes yra gaunama du kartus spustelėjus kairį pelės mygtuką pelės žymeklį užvedus ant ryšio. Vieną sykį spustelėjus kairį pelės klavišą ryšio linija yra pastorinama. Jei ryšio liniją du kartus spustelime pelės žymeklį užvedus ant ryšio linijos bus atidaroma ryšio savybes detalizuojanti forma, kurioje bus galima peržiūrėti ar modifikuoti ryšio savybes.

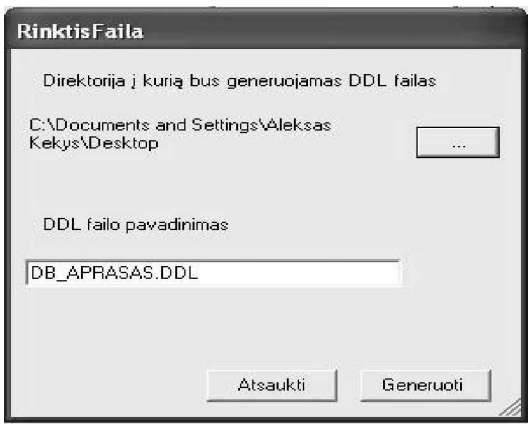
3.6. SQL DDL aprašo generavimas
3.6.1. Generavimas iš dalies ER modelio

Atlikus reikiamas modifikacijas ir įsitikinus loginio duomenų modelio teisingumu galima iš ER diagrama atvaizduoto duomenų modelio, generuoti fizinę reliacinės duomenų bazės schemą. Šiam veiksmui sužadinti reikia pasirinkti meniu punktą *„Generavimas“->”Generuoti DDL aprašą iš diagramos“* (9 pav.)



9 Pav. SQL DDL aprašo generavimo paleidimas iš dalies ER diagramos

Pasirinkus šį meniu punktą į ekraną išvedama forma kurioje nustatomas eksportuojamo failo kelias bei direktorija (10 pav.)



10 pav. Eksportavimui būtinų duomenų paruošimo forma

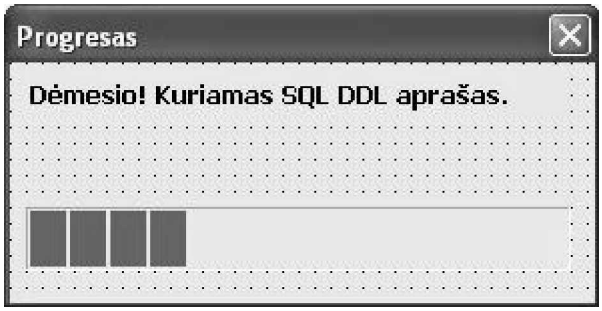
Šioje formoje galimi veiksmai yra : *pasirinkti eksporto direktoriją.* Šį veiksmą galima atlikti pasirinkus mygtuką pažymėtą trimis taškais „...“. Pasirinkus šį mygtuką yra atidaroma standartinė Windows direktorijų pasirinkimų forma (11 pav.)

11 pav. Standartinė direktorijų pasirinkimo forma

Šioje formoje nurodoma, kur bus saugomas reliacinės duomenų bazės schemą aprašantis tekstinis failas. Pasirinkus eksportavimo direktoriją, kelias vizualiai matomas vartotojui kairėje mygtuko pusėje.

Apatinėje formos dalyje yra laukas į kurį galima įvesti eksportuojamo failo pavadinimą. Failas gali būti įvedamas bet kokia pageidaujama vartotojui forma. Jei nepavyksta failo sukurti sistema informuoja vartotoją apie klaidą ir schemos generavimas nutraukiamas.

Pasirinkus mygtuką *„Generuoti“* į ekraną išvedama progreso stebėjimo forma kuri informuoja vartotoją apie formuojamo SQL DDL failo progresą (12 pav.)

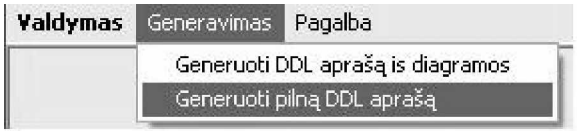


12 pav. SQL DDL aprašo generavimo progreso matuoklis

Suformavus SQL DDL aprašą sistema vartotoją informuoja apie sėkmingai atliktus veiksmus. Informuojama į ekraną išvedant pranešimo formą.

3.6.2. Generavimas iš pilno duomenų modelio sistemos atmintyje

Panaši veiksmų seka reikalinga norint suformuoti SQL DDL aprašą iš pilno duomenų modelio programinio modulio atmintyje. Skirtingas tik funkcijos iškvietimas. Paleisti generavimą iš viso modelio galima pasirinkus meniu punktą *„Generavimas“-> “Generuoti pilną DDL aprašą“*.



13 pav. Meniu punkto pilno SQL DDL duomenų aprašo generavimui paleisti parinkimas (13 pav.).

10.2 Straipsnis, „Duomenų bazės schemos SQL aprašo generavimas funkcinių reikalavimų specifikacijos pagrindu“

Duomenų bazės schemos SQL aprašo generavimas funkcinių reikalavimų specifikacijos pagrindu

Aleksas Kekys

Kauno technologijos universitetas, Informacijos sistemų katedra, Studentų g.50, Kaunas

Straipsnyje aprašomos funkcinių reikalavimų specifikavimo metodo fazės, kurių metu yra sudaromas projektuojamos informacinės sistemos duomenų saugyklos meta modelis, saugomas CASE įrankio, realizuojančio funkcinių reikalavimų specifikacijos metodą, duomenų saugykloje. Aprašomas vienas kuriamų CASE įrankio modulių bei procesas, kurio pagalba iš meta modelio, esančio duomenų saugykloje, sugeneruojamas SQL DDL aprašas – duomenų bazės reliacinė schema.

1. Įžanga

Kiekvienos kuriamos informacinės sistemos neatsiejama dalis informacijos saugyklos – duomenų bazė. Daugelyje projektavimo metodologijų, duomenų bazės modelio projektavimas išskiriamas kaip atskiras ir nepriklausomas etapas, kurio sėkmingumas priklauso nuo duomenų bazės analitiko/projektuotojo žinių lygio bei patirties. Taigi natūralu, kad projektuojant dideles informacines sistemas (IS), pavyzdžiui, atliekančių įmonių kompiuterizuotų funkcijų valdymą, apskaitos vedimą ar realizuojant didelės įmonės turinio valdymo sistemą, dažnai neišvengiama duomenų bazių analitiko klaidų.
Šio tipo klaidoms išvengti, IS duomenų bazės projektavimas ir realizavimas turėtų būti vientisas procesas. Idealiu atveju, duomenų bazės konceptualus modelis galėtų būti sukuriamas, identifikavus pagrindinius objektus bei duomenų srautus funkciniuose reikalavimuose, keliamuose sistemai.
Tada, duomenų bazių projektuotojas galėtų įvesti tinkamus pakeitimus, jau galutinėje projektavimo fazėje egzistuojančiam duomenų modeliui.

Kauno technologijos universiteto Informacijos sistemų katedroje plėtojamas *funkcinių reikalavimų specifikavimo metodas* bei jį realizuojantis CASE įrankis. Šio metodo esmė – kurti sistemą remiantis pirminiais vartotojo funkciniais reikalavimais sistemai. Viena šio metodo fazių – duomenų modelio, saugomo CASE įrankio meta duomenų saugykloje konvertavimas į fizinį duomenų bazės modelį bei SQL *data definition language (DLL)* aprašą – reliacinę duomenų schemą. Būtent ši konvertavimo fazė detalizuojama bei realizuojama šiame darbe.

2. Funkcinių reikalavimų specifikavimo metodas

Funkcinių reikalavimų specifikacijos metodu projektuojamos informacinės sistemos duomenų saugyklos metamodelis (Butkienė, Butleris, 2001), kuriuo remiantis generuojamas ir *SQL DDL* aprašas bei vaizduojama esybių – ryšių diagrama, sukuriamas tam tikrais etapais. Etapų eigoje palaipsniui surenkama duomenų modeliui (esybių-ryšių diagramai) sudaryti reikalinga informacija:

5. **Išskiriamas kuriamos IS kontekstas**
   Išskiriama kompiuterizuojamos organizacijos veiklos funkcijų aibė. Ši pakopa atliekama vartotojui nurodant veiklos sritį bei pateikiant apibendrintą atliekamų funkcijų aibę toje veiklos srityje apjungiant jas į hierarchiją (panašu į *Oracle* CASE metodą).
6. **Specifikuojami duomenų šaltiniai bei jų struktūra** (čia sudaromas ir lokalus autonomiškas duomenų modelis)
   Remiantis ankstesnės pakopos rezultatais – veiklos funkcijomis, gaunami vartotojo pageidaujami rezultatai (įvairios ataskaitos, analizės rezultatai ir panašiai), kuriuos turėtų pateikti IS. Apdorojamos tik tokios funkcijos (specifikuotos ankstesnėje pakopoje), kurias atliekant suformuojami duomenys, naudojami IS funkcionalumo rezultatų atributų reikšmėms išvesti.
7. **Specifikuojami ryšiai tarp duomenų šaltinių bei jų struktūra**

Specifikuojant duomenų šaltinius bei jų struktūrą nurodoma, iš kur bus paimti duomenys vienokio ar kitokio funkcinio rezultato formavimui, t.y. specifikuojami vartotojo reikalavimai IS įvedamai informacijai. Duomenų šaltiniai (DŠ) – tai organizacijos objektai, kuriuose saugoma informacija, susijusi su organizacijos veikla. Specifikuoti duomenų šaltiniai bus naudojami duomenų modeliui sudaryti. DŠ egzempliorius į specifikaciją įtraukiamas tik tuo atveju, jei yra bent vienas duomenų šaltinio atributas, naudojamas funkcionalumo rezultatų atributų reikšmėms formuoti. (Danikauskas, Butleris, 2004)

Srautas, perduodantis duomenis iš vieno duomenų šaltinio kitam duomenų šaltiniui arba rezultatui, vadinamas duomenų srautu (DS). Jo struktūrą sudarantys elementai nurodo, kokio duomenų šaltinio atributo reikšmės bus perduodamos ir koks išvedamo duomenų srauto arba kito duomenų šaltinio atributas jas gaus.

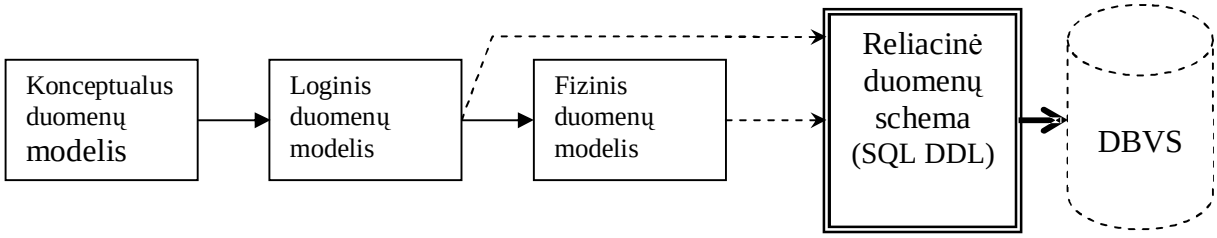
8. **Sudaromas integruotas IS duomenų modelis**

Kiekvienam įvedamus duomenis atitinkančiam objektui – duomenų šaltiniui, sudaromi autonomiški duomenų modeliai (Aleksandravičienė, Danikauskas, Butleris, Šidlauskas, 2005), (A.Jackutė, R.Butleris, 2003). Duomenų modeliuose esybės – duomenų šaltiniai, o ryšiai tarp esybių – duomenų srautai. Vėliau visi atskiriems duomenų šaltiniams sudaryti duomenų modeliai integruojami, sujungiant juos į bendrą duomenų modelį – IS loginę schemą.

Suprojektuotas kuriamos IS duomenų saugyklos meta aprašas bus saugomas *funkcinių reikalavimų specifikacijos* CASE įrankio duomenų saugykloje.

3. Duomenų modeliai

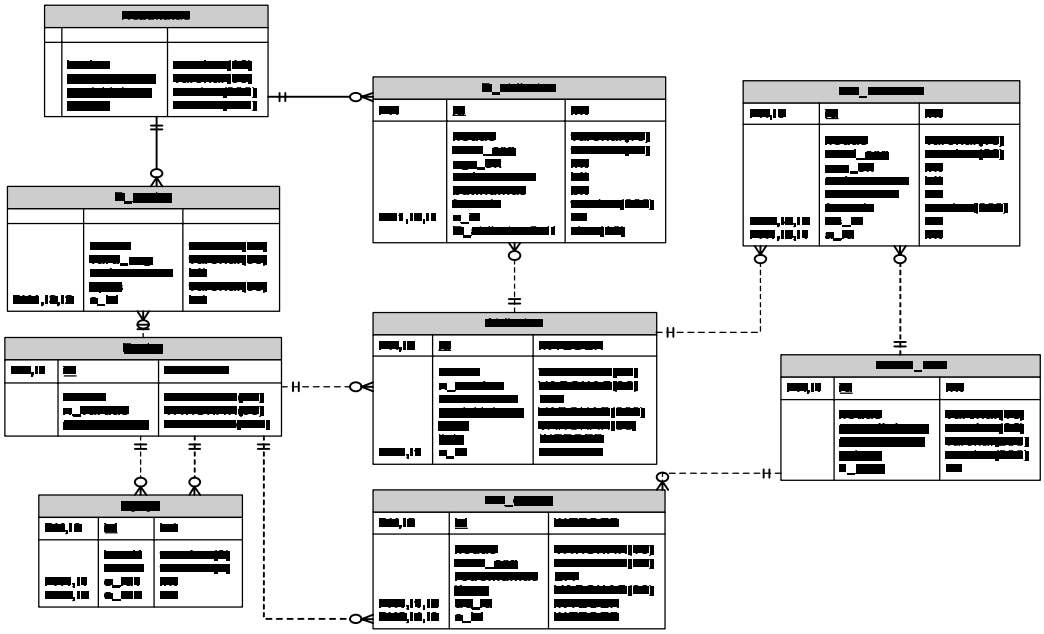
CASE įrankio duomenų saugykloje saugomas IS duomenų saugyklos meta aprašas atitinka *loginį duomenų* modelį. Be loginio duomenų modelio, dar egzistuoja konceptualus bei fizinis duomenų modeliai (Nicewarner, 2005). Taikant funkcinių reikalavimų specifikavimo metodą, konceptualaus duomenų modelio, norint sukurti loginį duomenų modelį, neprireikia. Taipogi nebūtinas ir fizinis duomenų modelis (kuris priartina loginį modelį prie tam tikros, konkrečios reliacinės DBVS) SQL DDL aprašo kūrimui, taigi, loginis duomenų modelis yra pakankamai išsamus reliacinei duomenų schemai generuoti. Aprašomame CASE įrankyje duomenų modeliai (esybių – ryšių diagramos), vaizduojami naudojant *informacijos inžinerijos* notaciją (D.C.Hey, 1999).



1 pav. Duomenų modeliai bei jų tranzityvumas

4. Reliacinės duomenų schemos aprašo generavimas

Šis funkcinių reikalavimų specifikavimo metodo etapas šiek tiek skiriasi nuo 3 skyrelyje aprašytų etapų, kadangi šio etapo rezultatas – pirmas žingsnis link fizinės IS duomenų saugyklos realizacijos. Ankstesniuose etapuose duomenys buvo kaupiami bei apdorojami tik CASE įrankio duomenų saugykloje. Duomenų saugyklos diagrama pateikiama 2 pav.



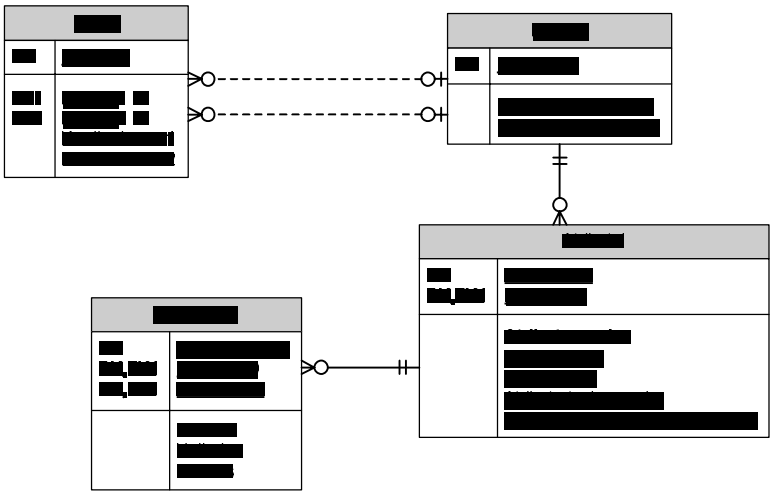
2 pav. Duomenų saugyklos fragmentas, naudojamas SQL DDL aprašui generuoti

Eksportuojant duomenis iš CASE įrankio saugyklos, duomenų loginį modelį siūloma perkelti į tarpinę duomenų struktūrą (3 pav.), kurioje būtų aprašyta informacija, susijusi tik su duomenimis, kuriuos būtina eksportuoti į generuojamą SQL DDL failą bei pateikti duomenų analitikui, kad pastarasis galėtų peržiūrėti ar paredaguoti iš saugyklos įkeltą duomenų modelį. Tai lentelių bei jų atributų, tiek lokalių, tiek ir atkeliamų iš kitų esybių identifikuojančiais ryšiais, aprašai, greta kurių būtų saugoma informacija apie apribojimus generuojamiems duomenims. Šios laikinos struktūros pagalba tikimasi ateityje realizuoti ir egzistuojančios reliacinės duomenų bazės reinžineriją (Abu-Hamdeh, Cordy, Martin, 1994) į funkcinių reikalavimų specifikacijos duomenų saugyklą. Duomenų meta modelio transformavimo į reliacinę schemą apibendrintas algoritmas pateikiamas žemiau :

9. Duomenų surinkimas bei interpretavimas
   - 9.1. Iš metamodelio nuskaitoma informacija apie visas išskirtas esybes
   - 9.2. Kiekviena esybė apdorojama išrenkant jos atributus bei ryšius, kuriuose esybė dalyvauja
   - 9.3. Radus susijusius ryšius, rekursiškai nagrinėjami visi susiję identifikuojantys ryšiai ir į laikiną struktūrą perkeliami susijusių ryšiais esybių atributai
   - 9.4. Iš susijusių duomenų šaltinių bei rezultatų lentelių, išrenkama informacija apie apribojimus atributams, bei papildoma tarpinės struktūros apribojimų lentelė.

10. Išanalizuotų duomenų transformavimas į SQL DDL
    - 10.1. Esybės, esančios laikinoje struktūroje, pradedamos apdoroti po vieną. SQL DDL apraše pirma sukuriamos lentelės su visais atributais (kartu ir su atkeltais atributais iš kitų esybių), bei atributų apribojimais (PRIMARY KEY, UNIQUE, NOT NULL, DEFAULT ir t.t.)
    - 10.2. Kuriant lentelių aprašus, lygiagrečiai yra kuriami išorinių raktų apribojimų SQL DDL aprašai (jeigu esybės atributo laukas „Atributo tėvinė esybė“ nesutampa su esybės numeriu)
    - 10.3. Apdorojus visas esybes, lentelių aprašų dalis papildoma išorinių raktų apribojimais.

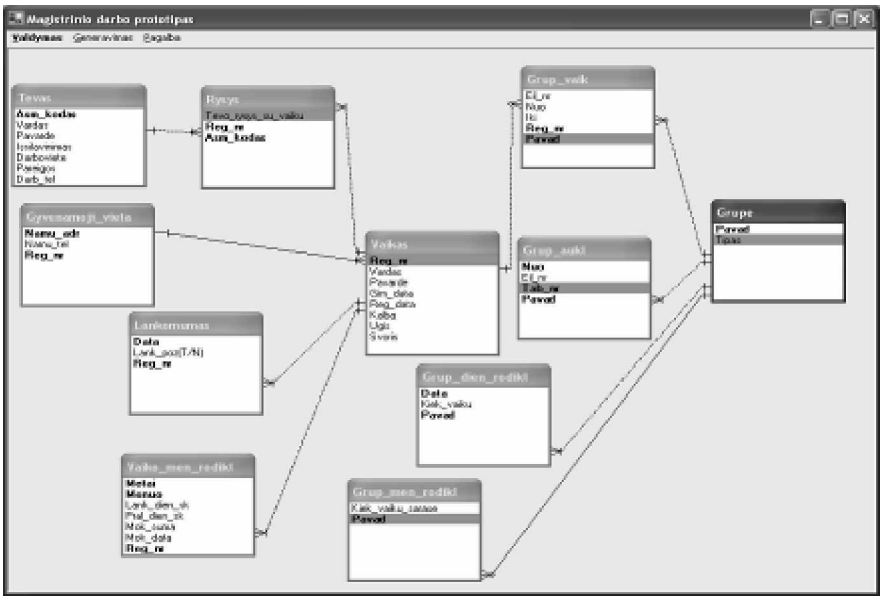
Informacija apie duomenų modelį yra pateikiama ekrane esybių-ryšių modelio pavidalu informacijos inžinerijos notacijoje. Analitikui, norint paredaguoti esybių – ryšių modelį, modifikacijos išlieka tik laikinai, tarpinėje struktūroje. Kitos sesijos metu duomenys vėl bus įkraunami iš CASE duomenų saugyklos.



3 pav. Tarpinė duomenų struktūra tarp loginio duomenų modelio ir reliacinės schemos

5. Realizacija

Projektuojant CASE įrankį buvo tikėtasi panaudoti *Microsoft Visio* paketo duomenų bazių projektavimo šablono funkcines galimybes (pasitelkiant Visual Basic for applications bei *MS Visio SDK*), tačiau, ištyrus, kad Microsoft korporacija neteikia jokių įrankių ar metodų (Miller, 2004), kurių pagalba būtų galima naudoti minėto šablono savybes, buvo pasirinktas sprendimas – naująją funkcinių reikalavimų specifikavimo metodo fazės realizaciją sukurti remiantis Microsoft .NET technologijomis. Ši platforma buvo pasirinkta todėl, kad ADO.NET *DataSet* (MS .NET dalis) pagalba galima lengvai realizuoti tarpinę struktūrą loginiam duomenų modeliui saugoti ir transformuoti į SQL DDL (3 pav.), taipogi dėl realizuotų labai paprastų priemonių Windows programų kūrimui bei integravimui su kitomis sistemomis. CASE įrankio modulį realizuojančios programos vaizdas pateikiamas 4 pav.



4 pav. Duomenų modelio analizės, reliacinės schemos generavimo įrankio vaizdas

6. Išvados

Aprašytas reliacinės duomenų bazės schemos (SQL DDL) generavimo procesas bei duomenų, reikalingų šio proceso funkcionavimui, paruošimas. Ši funkcinių reikalavimų specifikacija paremto metodo IS projektavimo fazė yra pirmoji, o jos rezultatas – pirmasis žingsnis į fizinę projektuojamos IS realizaciją. Straipsnyje trumpai pristatytas CASE įrankio modulis, kurio pagalba realizuojamas duomenų bazės reliacinės schemos generavimas. Ateityje, įrankyje realizuotos tarpinės duomenų struktūros pagalba būtų galima atlikti CASE duomenų saugyklos meta modelio papildymą fragmentais iš kitų funkcinių reikalavimų specifikacijos metodą realizuojančių CASE įrankių duomenų saugyklų, taip sistemoje sukuriant reinžinerijos galimybę.