KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Arjana Matonytė

Užklausų metamodelis paieškos sistemų kūrimui

Magistro darbas

Darbo vadovas
Prof. Dr. L. Nemuraitė

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Arjana Matonytė

Užklausų metamodelis paieškos sistemų kūrimui

Magistro darbas

Recenzentas
Doc. Dr. S. Maciulevičius
2007-05-

Vadovas
Prof. Dr. L. Nemuraitė
2007-05-

Atliko
IFM-1/1 gr. stud.
Arjana Matonytė
2007-05-24

Kaunas, 2007

Turinys

1. ĮVADAS

Duomenų išrinkimas, konvertavimas, transformavimas ir integravimas – gerai žinomos duomenų bazių problemos, kurių sprendimus įgyvendina užklausų kalbos. Jau kelis dešimtmečius dažniausiai naudojama duomenų saugojimo struktūra – pagal E. F. Codd apibrėžtas taisykles [8], kurios, laikui bėgant buvo tobulintos ir pritaikomos prie kitų besivystančių technologijų, kuriamos reliacinės duomenų bazių sistemos. Be daugumai duomenų saugojimo struktūrų būdingų savybių, reliacinės duomenų bazės pasižymi tokiomis savybėmis, kaip saugumu, saugojimo efektyvumu, korektiškumo kontrole ir t. t. [18], kurios atitinka realius organizacijų duomenų saugojimo poreikius, dėl ko dažniausiai praktikoje ir yra naudojamos. Reliacinių duomenų bazių kūrėjai naudoja SQL kalbą, kurią dar 1986 m. standartu patvirtino ANSI (angl. *American National Standards Institute*), o 1987 ir ISO (angl. *International Standards Organization*) standartus kuriančios organizacijos.

Dauguma sistemų naudoja skirtingų formatų duomenis, todėl, siekiant išspręsti duomenų perdavimo tarp sistemų bei integravimo tarp nevienalyčių duomenų šaltinių problemas, išvystytas ir plačiai naudojamas duomenų aprašymo standartas XML (angl. *eXtensible Markup Language*). Navigavimui hierarchine struktūra duomenis saugančiuose XML dokumentuose skirta XPath kalba, kurios dabartinė 2.0 versija yra W3C (angl. *World Wide Web Consortium*) – rekomendacija, o jos ribotam funkcionalumui išplėsti skirta rekomendacijos statusą turinti bei XPath išraiškas palaikanti XQuery užklausų kalba.

Kadangi reliacinė bei XML duomenų saugojimo struktūra skiriasi – reliacinėse duomenų bazėse duomenys saugomi dvimatėse lentelėse, o XML naudoja hierarchinę struktūrą – SQL ir XML dokumentų užklausų kalbų – XQuery ir XPath – struktūra, savybės ir funkcijos taip pat tam tikra dalimi skiriasi. Be to, tarpusavyje skiriasi ir SQL kalbos realizacijos. Tačiau, nepaisant šių skirtumų, visos kalbos turi ir bendrumų. Struktūrinius bendrumus nulemia jau pačios užklausos paskirtis: išrinkti duomenis iš duomenų struktūrų pagal nurodytas sąlygas. Šiuos bendrumus kartu su kitomis sutampančiomis savybėmis galima išreikšti sudarant bendrą užklausų modelį, tuo pačiu perkeliant užklausų vaizdavimą į aukštesnį abstrakcijos lygmenį bei taip suteikiant šiai sričiai aiškumo.

Būtent šios idėjos pagrindu buvo suformuotas pagrindinis **darbo tikslas** – sudaryti bendrinį užklausų modelį. Informacija, reikalinga tokiam modeliui sudaryti, buvo atrinkta atliekant nagrinėjamų užklausų kalbų lyginamąją analizę, kurios metu sugretinami išskirtų lyginimo aspektų rezultatai. Tokiu būdu įvertintos savybės, kurias modelis turi apimti.

Atliekant analizę, susiformavo idėja, kad užklausų kalbos naudojamos paieškos sistemose, o iki šiol nėra metodų, rekomendacijų, nusakančių, kaip reikėtų kurti paieškos

sistemas. Jeigu modelis būtų sudaromas atsižvelgiant ir į paieškos sistemoms būdingas savybes, tai juo būtų galima naudotis kaip metodika tokioms paieškos sistemoms kurti. Taigi, remiantis šia idėja, sudarytas modelis bei juo pagrįsta metodika kurti minėtas sistemas. Sudaryti algoritmai dviems paieškos sistemų kūrimo etapams: paieškos sąsajos apibrėžimui bei užklausų sudarymui. Realizuotas paieškos sistemos prototipas, įrodantis, kad modelį galima naudoti automatizuojant paieškos sistemų kūrimą.

Šio darbo **tyrimo sritis** yra standartinės užklausų kalbos, skirtos labiausiai praktikoje naudojamoms – reliacinėms ir XML – duomenų struktūroms, o **tyrimo objektas** – bendrinis modelis, kurį galima būtų panaudoti užklausų formavimo praktikoje bei paieškos sistemų kūrimo procese.

Iki šiol nėra sukurto panašaus modelio, o ir pati kalbų išsami tarpusavio analizė nebuvo atlikta – egzistuoja tik pavieniai, nenuodugnūs bandymai sulyginti SQL ir XQuery kalbas [14-16]. Tai patvirtina ir Don Chamberlin, vienas iš XML užklausų kalbų specifikacijų redaktorių bei SQL kūrėjų, tuo pačiu pateikdamas tam tikras gaires SQL ir XQuery kalbų palyginimui [7, 16], kuriomis remiamasi ir šiame darbe. Todėl modeliui sudaryti reikalinga išsami užklausų kalbų analizė, kas ir yra vienas iš darbo uždavinių:

- atlikti užklausų kalbų palyginimą ir susisteminti gautus rezultatus;
- pasirinkti modeliavimo aplinką;
- nustatyti ryšius tarp užklausų kalbų elementų;
- remiantis analizės rezultatais, sudaryti užklausų metamodelį;
- parengti metodiką modelio taikymui kuriant paieškos sistemas;
- išbandyti metodikos taikomumą pasirinktai sričiai.

Darbo struktūra:

– Šio darbo **analizės dalyje** pateikta SQL, XPath ir XQuery kalbų duomenų modelio, savybių bei funkcijų, reikšmingų bendrinio modelio sudarymui, analizė. Ji atlikta sujungiant turimą informaciją apie atskiras kalbas, apibrėžtą W3C [21-24] ir ANSI [1, 2, 9] standartais, apie atskiras kalbas bei remiantis anksčiau atliktoje XQuery ir kitų siūlomų XML užklausų kalbų analizėje [14, 16, 19] naudojamais lyginimo aspektais. Modeliavimo aplinka pasirinkta universali modeliavimo kalba UML (angl. *Unified Modeling Language*). Aprašyta užklausų kalbų elementų semantika bei UML klasių diagramos sąvokos ir elementai, taip pat pateikta įrankių, skirtų užklausoms sudaryti ir modeliuoti, analizė. Nuspręsta kurti bendrą užklausų modelį numatant jį panaudoti paieškos sistemų kūrimo procese bei tuo tikslu atlikta egzistuojančių paieškos sistemų savybių analizė.

– **Pagrindinėje darbo dalyje** pateikta užklausų metamodelio sudarymo eiga, grįsta analizės metu gautais rezultatais. Parinkti UML elementai metamodeliui vaizduoti bei

pritaikyti UML teikiami išplėtimo mechanizmai, suteikiantys galimybę standartiniams šios kalbos elementams suteikti naują prasmę – profiliai, stereotipai ir žymėtosios reikšmės. Sudaryti algoritmai, skirti paieškos sistemų sąsajos kūrimo bei paieškos vykdymo eigai apibrėžti. Pateikta metodika, kaip metamodelio informacija taikoma konkrečiai dalykinei sričiai, realizuojant automatinį paieškos sistemų kūrimą, bei išnagrinėtas paieškos sistemai reikalingų savybių poreikis. Aptartos šios realizacijos galimybės bei pasirinkta naudoti XMI standartą, kurio struktūrą apibrėžia OMG (angl. *Object Management Group*). XMI yra XML formato, todėl patogu juo manipuliuoti naudojant XML dokumentams taikomas technologijas.

– **Eksperimentinėje dalyje**, siekiant patvirtinti modelio ir metodikos įgyvendinamumą, sukurtas prototipas. Jame naudojamas sudarytas užklausų metamodelis bei koncepcinės srities modelis paieškos sistemos sąsajai sukurti. Tai atliekama pritaikius minėtus UML išplėtimo mechanizmus bei kūrimui naudojant XSLT transformacijos dokumentą, nuskaitantį modelių duomenis bei apibūdinantį transformavimo taisykles. Funkcionalumo požiūriu prototipas patvirtina, kad modelis gali būti pritaikomas praktikoje kuriant paieškos sistemas.

Pagrindinis šio **darbo rezultatas** – sudarytas modelis bei parengta metodika, apibrėžianti jo taikymą kuriant paieškos sistemas. Vienas iš modelio ir metodikos privalumų globaliu požiūriu – užklausų projektavimas įjungiamas ir į likusios sistemos kūrimo procesą, kam iki šiol nėra sukurtų rekomendacijų ar standartinių priemonių; šis privalumas ateityje galėtų būti išnaudojamas ir kitose sistemose.

Ateityje modelis galėtų būti plečiamas – XQuery kalba tik 2007 m. tapo rekomendacija, todėl numatoma, kad vėlesnėse versijose ji turės papildomų savybių, kurios, jeigu atitiktų SQL savybes, būtų tinkamos perteikti bendrame modelyje. Taip pat, nesusiejant modelio su jo panaudojimo sritimi, jis galėtų apimti daugiau nustatytų bendrų savybių, pavyzdžiui, sąjungų atvejus. Visas struktūrines savybes apimantis modelis galėtų būti naudojamas, pavyzdžiui, sudarant priemonę vartotojams formuoti užklausas nerašant užklausos konkrečia kalba, o parenkant jos dalis iš sąsajoje pateikiamų elementų.

Darbe nagrinėjama tema pristatyta 12-ojoje tarpuniversitetinėje magistrantų ir doktorantų konferencijoje „Informacinė visuomenė ir universitetinės studijos“. Pranešimo medžiaga atspausdinta konferencijos leidinyje ir jos turinys pateiktas 1-ajame priede.

2. UŽKLAUSŲ KALBŲ – SQL, XPATH IR XQUERY - ANALIZĖ

Pradedant nagrinėti užklausų kalbas, visų pirma pateikiama bendra informacija apie duomenų šaltinius ir pristatoma jiems skirtų kalbų analizė, kuri yra vienas iš darbo tikslų. Atliekant analizę, siekiant išvengti daugiaprasmiškumų, galinčių atsirasti dėl nevienodos praktikoje naudojamos terminijos, svarbu tiksliai apibūdinti pagrindines aktualių duomenų šaltinių sąvokas, kas ir pateikiama tolesniuose poskyriuose. Be to, duomenų struktūros taip pat tiesiogiai veikia užklausų kalbas. Aptarus struktūras, nustatomos užklausų kalbų atliekamos operacijos, nuo ko priklauso toliau pateikiama užklausų struktūros nagrinėjimo apimtis. Taip pat nagrinėjamos užklausų struktūrai būdingos savybės ir naudojami operatoriai bei lentelės forma pateikiami apibendrinti atliktos analizės rezultatai. Galiausiai pasirenkama modelio taikymo sritis – paieškos sistemų kūrimas – ir taipogi atliekamas įvairių paieškos sistemų savybių tyrimas.

2.1. Duomenų šaltiniai ir joms skirtos užklausų kalbos

Dauguma šiuolaikinių informacijų sistemų duomenų saugojimui naudoja reliacines duomenų bazes (RDB). Standartinė RDB užklausų kalba – SQL. Nors tiek ANSI, tiek ISO organizacijos yra apibrėžusios SQL užklausų kalbos standartą, konkrečiose reliacinėse duomenų bazių valdymo sistemose (RDBVS) SQL realizacija tam tikra dalimi skiriasi. To priežastimis išskiriami tokie aspektai, kaip SQL standarto sudėtingumas ir dydis – dauguma RDBVS realizuoja ne visą standartą, neapibrėžti tam tikri DB veiksmai keliose svarbiose srityse (pvz., indeksų atžvilgiu) – RDBVS juos realizuoja savo nuožiūra, mažiau apibrėžta kalbos konstruktų semantika – tam tikrose srityse atsiranda daugiaprasmiškumas ir kiti [5].

Duomenų bazių sistemoms apdorojant vis sudėtingesnių tipų informaciją, atsirado ir lankstaus duomenų formato poreikis. Sistemos paprastai tarpusavyje keičiasi duomenimis, duomenų šaltiniai skiriasi, atsiranda nesuderinamumai. Net ir tarp tos pačios rūšies duomenų šaltinių egzistuoja žymūs skirtumai. Duomenų šaltinių nevienalytiškumo problemą sprendžia XML kalba, apibrėžianti duomenų objektų, vadinamų XML dokumentais, klasę ir iš dalies apibrėžianti jų apdorojimo principus [7]. XML dokumentuose saugomi duomenys ir jų turinį aprašančios žymės. Kadangi XML yra savitos struktūros, jai naudojamos atskiros užklausų kalbos, efektyviausiai atliekančios operacijas su šios struktūros duomenimis: XPath ir XQuery.

XPath kalba jos specifikacijoje įvardijama ne kaip užklausų kalba, o kaip sintaksė XML dokumento dalių vietai nustatyti, skirta naudoti kartu su XML dokumentų transformavimo kalba XSLT (angl. *eXtensible Stylesheet Language Transformations*) arba kreipinius į XML

failo dalis nurodančia XPointer kalba [22]. Praktikoje XPath dėl dažnai vadinama išraiškų kalba; tačiau, be pagrindinės paskirties, ji taip pat teikia priemones operacijoms su eilutėmis (angl. *strings*), skaičiais ir loginėmis reikšmėmis atlikti ir gali būti vadinama bei tarp jos naudotojų turi užklausų kalbos statusą [12, 20].

Augant XML formatu saugomos ir vaizduojamos informacijos bei XML formato duomenimis keitimosi apimčiai, išauga ir XML duomenų šaltiniams skirtos užklausų kalbos poreikis. Vienas iš XML kalbos privalumų, lėmusių jos naudojimo išplitimą – lankstumas atvaizduojant įvairialypę, iš skirtingų šaltinių gautą informaciją. XQuery sukurta kaip kalba, skirta šiam XML lankstumui išnaudoti, pateikianti funkcines savybes, susijusias su informacijos iš įvairių šaltinių išrinkimu ir interpretavimu [16]. XQuery užklausų kalba išplečia XPath kalbą bei teikia papildomas, galingesnes savybes, iš kurių išskirtinesnė – naujų elementų kūrimo galimybė. Dabartinė XQuery 1.0 versija, kuri yra W3C rekomendacija, dar neturi kitų dažnai užklausų kalboms būdingų savybių, tokių, kaip duomenų šalinimas ar saugomų duomenų keitimas.

Kadangi XPath užklausų kalba vis dėlto nėra savarankiška, tikslingiau ją nagrinėti kaip papildomas galimybes XQuery kalbai suteikiančią kalbą. Tolesnėje užklausų kalbų analizėje, atliekant kalbų lyginimą, XPath kalba išskirta kaip atskira kalba, tačiau, sudarant užklausų metamodelį, atsižvelgiama į XQuery ir XPath savybių visumą, prieš nusprendžiant, ar ši savybė atitinka SQL teikiamą savybę.

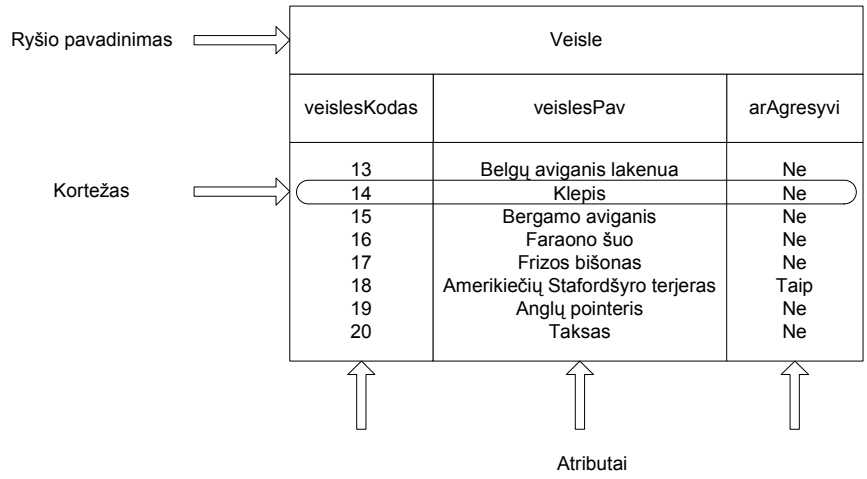
2.2. Reliacinio ir XML duomenų modelių analizė

Konkrečios užklausos kalbos sąvokos sietinos su duomenų struktūra, kuriai užklausų kalba yra skirta. Tiek RDB, tiek XML sąvokos skiriasi, taigi ir jų duomenų modeliai yra nevienalyčiai. Norint, kad užklausų kalbų analizės rezultatai būtų nedaugiaprasmiški, kas gali pasireikšti dėl terminų ir sąvokų sampratos nevienodumo, tikslinga apibrėžti pagrindines RDB ir XML duomenų modelių sąvokas.

Reliacinis duomenų modelis

Reliacinis duomenų modelis (RDM) apibrėžia duomenų atvaizdavimo būdą (duomenų struktūrą), operacijas, kurias galima atlikti su duomenimis, ir duomenų bazės integralumui taikomas taisykles [18]. Duomenų struktūrą sudaro dvi komponentės: esybės ir ryšiai tarp esybių. Esybės vaizduojamos ryšiais (angl. *relations*) arba bazinėmis lentelėmis (angl. *base tables*). Abu terminai reiškia vieną ir tą patį – *ryšys* yra matematinis *lentelės* terminas. Bazinė lentelė apibrėžiama kaip nesurikiuotas (t. y. neturintis specifinės eilės tvarkos) rinkinys, kuriam priklauso nulis, vienas ar daugiau kortežų (dar vadinamų *eilutėmis*), sudarytų iš vieno

ar daugiau nesurikiuotų atributų (dar vadinamų *stulpeliais*). Visus kortežus sudaro vienodas atributų rinkinys. 1 pav. pateiktas RDB lentelės pavyzdys.

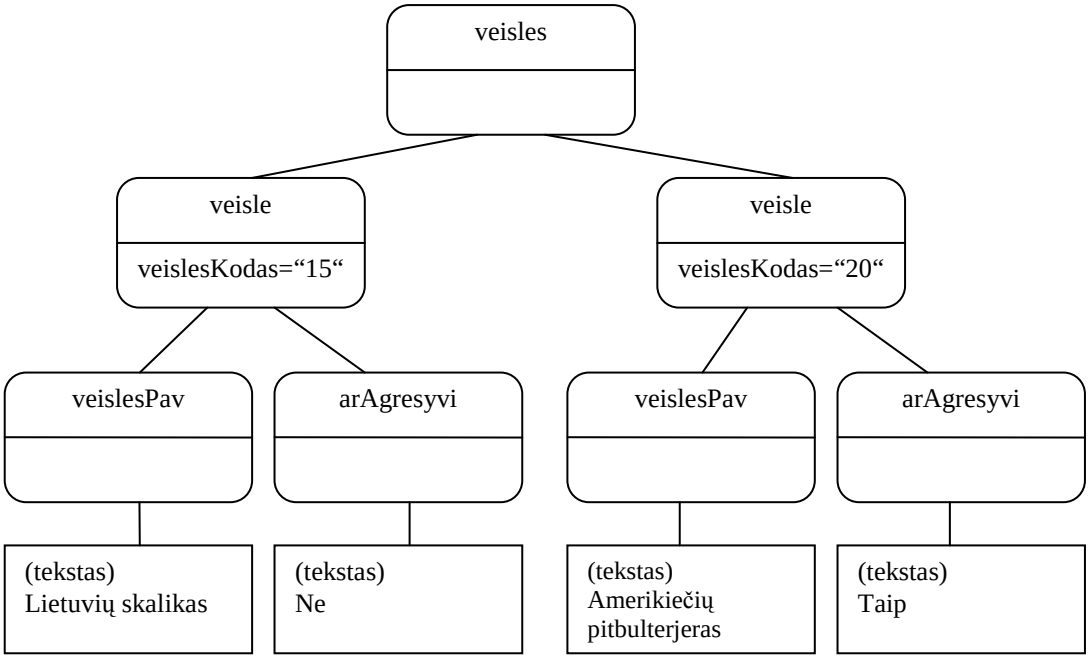


1 pav. Reliacinės duomenų modelio ryšys

XQuery ir XPath duomenų modelis

XML dokumente duomenys saugomi hierarchinėje, medį primenantį struktūroje (2 pav.). Medį sudaro aibė mazgų, kurių kiekvienas gali būti sujungtas su nuliu ar daugiau žemesnio lygio mazgų, vadinamų mazgo vaikais. Kiekvienas mazgas atitinka XML elementą, kuris gali būti tuščias, turėti tekstinę reikšmę arba turėti mazgų-vaikų.

XML teikia tiktai gaires, kuriomis remiantis gali būti kuriami sudėtingesni modeliai, tačiau atskira XML duomenų modelio specifikacija nėra pateikiama. Tačiau W3C apibrėžia XQuery ir XPath duomenų modelį, kuriuo ir bus remiamasi toliau nagrinėjant XML užklausų kalbas.




```
<veisles>
   <veisle veislesKodas ="15">
      <veislesPav>Lietuvių skalikas</veislesPav >
      <arAgresyvi>Ne</arAgresyvi>
   </veisle>
   <veisle veislesKodas="20">
      <veislesPav>Pitbulterjeras</veislesPav >
      <arAgresyvi>Taip</arAgresyvi>
   </veisle>
</veisles>
```

2 pav. Hierarchinė XML dokumento struktūra, pavaizduota grafiškai ir išeities tekstu

XQuery ir XPath duomenų modelis apibrėžiamas formalaus duomenų modelio, o ne XML teksto sąvokomis. Jis apibūdina:

- kokia informacija pateikiama XQuery užklausai apdoroti, informaciją jos apdorojimo metu bei išvedamą informaciją;
- reikšmes, kurias galima naudoti XQuery bei XPath išraiškose.

Pagrindines duomenų modelio charakteristikas įvardija abstraktus duomenų (informacijos) rinkinys, pavadintas XML Information Set (sutrumpintai – Infoset). Jis yra skirtas teisingos struktūros XML dokumentų sąvokoms apibrėžti. Infoset išskiria 11 informacijos elementų: dokumento, elemento, atributo, apdorojimo instrukcijos, neišplėstos esybės nuorodos (angl. *unexpanded entity reference*), simbolių (angl. *character*), komentaro, dokumento tipo paskelbimo (angl. *document type declaration*), neanalizuojamos esybės (angl. *unparsed entity*), žymėjimo (angl. *notation*) ir vardų srities informacijos elementai. Išsamus šių informacijos elementų aprašas pateiktas XML Information Set rekomendacijoje [20]. Taip pat pažymimos kelios Infoset neapibrėžtos duomenų modelio savybės:

- XML schemos tipų palaikymas. XML griežtai specifikuoja sintaksę, tačiau leidžia laisvai apibrėžti XML dokumentų prasmę. Tuo tarpu XML schemos rekomendacija apibrėžia savybes, tokias, kaip struktūras ir paprastuosius duomenų tipus, išplečiančius Infoset tikslia informacija apie tipus. XML skirtos dvi schemos – XML Schema ir duomenų tipo apibrėžtis (DTD).
- Dokumentų rinkinio ir sudėtinių reikšmių vaizdavimas.
- Tipizuotų atominių reikšmių palaikymas. Atominėmis vadinamos atominio tipo reikmės, neapimančios kito tipo reikšmių.
- Rikiuotų nevienalyčių sekų palaikymas. Seka vadinamas rikiuotas rinkinys, sudarytas iš nulio ar daugiau elementų (angl. *items*) – mazgų ar atominių reikšmių.

Detaliau bus aptarti struktūriniai elementai bei jų hierarchinis vaizdavimas.

Kiekviena užklausos įvestis ir kiekvienos užklausos išvestis – duomenų modelio egzempliorius. XQuery/XPath duomenų modelyje kiekvienas dokumentas vaizduojamas kaip iš mazgų sudarytas medis. Egzistuoja šie mazgų tipai: dokumento, elemento, atributo, teksto, vardų srities, apdorojimo instrukcijos ir komentaro. Kiekvienas mazgas turi identifikatorių, pagal kurį jis atsiskiria nuo kitų mazgų – net ir nuo tų, kurie kitais atžvilgiais visapusiškai sutampa [15, 24].

Be mazgų, duomenų modelyje apibrėžiamos atominės reikšmės, t. y. paprastos reikšmės, atitinkančios paprastąjį tipą, apibrėžtą W3C rekomendacijoje „XML Schema, Part 2 [24], tokį, kaip eilučių, loginių reikšmių, dešimtainių, sveikųjų, slankaus kablelio, dvigubo tikslumo ir datos tipus. Šie paprastieji tipai gali būti naudojami bet kuriame dokumente, susietame su XML schema.

Elementas (angl. *item*) yra paprastasis mazgas arba atominė reikšmė. Elementų eilė vadinama seka. XQuery kiekviena reikšmė yra seka, ir nėra skirtumo tarp atskiro elemento ir sekos, kuriai priklauso vienas elementas. Sekoms gali priklausyti mazgai ir atominės reikšmės, tačiau joms negali priklausyti kitos sekos.

Pirmasis mazgas dokumente – dokumento mazgas, kurį sudaro visas dokumentas. Dokumento mazgas neatitinka jokios matomos elemento dalies; jis vaizduoja patį dokumentą. Elemento, komentaro ir apdorojimo instrukcijos mazgai rikiuojami tokia tvarka, kokia jie randami XML dokumente. Elemento mazgai eina prieš savo vaikus (t. y. žemesnio lygio medyje, elemento apimamus kitus elementus) – elemento, teksto, komentaro ir apdorojimo instrukcijos mazgus. Atributai nelaikomi elemento vaikais, tačiau jie turi apibrėžtą poziciją dokumento tvarkoje: jie eina po elemento, kuriame jie buvo rasti, prieš elemento vaikus.

Nors tiek XQuery/XPath, tiek reliacinis duomenų modelis yra labai bendri savo galimybėmis vaizduoti visų tipų duomenis, egzistuoja skirtumai tarp duomenų, paprastai saugomų šiais formatais, tipų [7, 11]:

- XML duomenys dažnai būna išsklaidyti, t. y. jie neturi reguliarios struktūros, kaip reliaciniai duomenys – dalis elementui priklausančių žemesnio lygio elementų gali būti nepateikti;
- reliaciniai duomenys neturi vidinės tvarkos, nepriklausančios nuo jų reikšmių, tačiau XML dažnai naudojama saugoti sutvarkytiems duomenims, tokiems, kaip knygos paragrafai;
- lyginant su tipine reliacinės duomenų bazės schema, apibrėžiančia fizinę duomenų bazės struktūrą, XML schemos informacija dažnai yra sudėtingesnė ir didesnė jos keitimosi tikimybė.

Atskirai bus aptarti ir sulyginti reliacinių ir XML struktūrų tipizavimo mechanizmai, ryšiai tarp jų duomenų elementų bei naudojamos identifikavimo priemones.

Tipizavimo mechanizmai

XML elementus galima suklasifikuoti pagal tipus ir domenus, t. y. atributų reikšmių sritis (1 lentelė). Pirmajame stulpelyje pažymėti iš atominės srities reikšmes įgyjantys elementai, antrajame – iš sudėtinės srities. Elementai gali būti atominiai, sudėtiniai, t. y. sudaryti iš elementų, sudėtiniai su mišriu turiniu arba tušti. Tuo tarpu reliacinė duomenų bazių sistema (RDBS) leidžia apibrėžti reikšmių sritis tik stulpeliams, o ne lentelėms [3, 25].

1 lentelė. Elementų tipai

Elementų tipai (ET)	Atominė sritis	Sudėtinė sritis
Atominis ET	+	-
Sudėtinis ET, sudarytas iš elementų	-	+
Sudėtinis ET su mišriu turiniu	+	+
Tuščias ET	-	-

Identifikavimas

RDBS unikalaus ryšio egzemplioriaus – lentelės eilutės – identifikatorius yra pirminio rakto (angl. *primary key*) reikšmė. Lentelės eilutė yra unikali lentelės ribose, tai yra ją galima išskirti pagal raktą nuo kitų tos pačios lentelės eilučių. XML elemento mazgas gali būti identifikuojamas naudojant unikalų identifikatorių, tai yra, atributo, apibrėžto duomenų tipo apibrėžties identifikatoriumi arba paskelbto XML schemoje raktu, reikšmę. Jokie du dokumento elementai negali turėti to paties unikalaus identifikatoriaus. Jeigu dokumentas neturi schemos, joks dokumento elementas neturi unikalaus identifikatoriaus [21]. RDBS identifikatoriumi – pirminiu raktu – gali būti tiek atskiras lentelės stulpelis, tiek kelių stulpelių

derinys. XML identifikatoriumi, jeigu naudojama DTD, gali būti nurodomas tik vieno elemento tipo atributas, o jeigu naudojama XML schema – vienas ar keli atributo tipo elementai.

Ryšiai

Ryšiai tarp RDBS lentelių išreiškiami išoriniais raktais (angl. *foreign keys*) – stulpeliais, kuriuose nurodomas tos pačios ar kitos lentelės pirminis raktas. Lentelės eilučių kardinalumą, tai yra galinčių dalyvauti ryšyje eilučių skaičių, galima nurodyti apibrėžiant išorinį raktą kaip ne nulinį (angl. *not null*) ir (arba) unikalų (angl. *unique*) [3].

Analogiški XML ryšiai tarp elementų vėlgi gali būti apibrėžiami taikant schemas. XML schema palaiko *keyref* sąvoką, kuri panaši į RDBS išorinio rakto sąvoką, kas reiškia, kad tam tikras elemento/atributo derinys nurodo atitinkamą elemento/atributo derinio kuriamą raktą. Taikant DTD, ši teikia du būdus ryšiams tarp elementų tipų apibrėžti, t. y. *IDREF(S)* atributus ir sudėtinius elementų tipus.

2.3. Užklausų atliekamos operacijos

Apibrėžus pagrindines nagrinėjamų duomenų struktūrų – reliacinių ir XML – sąvokas, svarbu išsiaiškinti, kokias operacijas gali atlikti kiekviena užklausų kalba, kadangi skirtingoms operacijoms atlikti skirtų užklausų struktūra skiriasi, o tai savaime susiję ir su užklausų savybėmis, taigi ir su tikslinga nagrinėti savybių aibe.

Užklausos gali būti naudojamos informacijai išrinkti, jai keisti, šalinti ar naujam duomenų elementui sukurti. Visų trijų kalbų atliekamos operacijos pažymėtos 2 lentelėje, kurioje galima matyti, kad visos užklausos turi tik informacijos išrinkimo galimybę. Tai lemia, kad toliau, nagrinėjant užklausų struktūrą ir jai būdingas savybes, bus orientuojamasi į informacijos išrinkimo užklausas.

2 lentelė. Galimos užklausų kalbų operacijos su duomenimis

Funkcinė savybė/užklausų kalba	SQL	XPath	XQuery
Duomenų išrinkimas	+	+	+
Naujų duomenų elementų kūrimas	+	-	+
Duomenų šalinimas	+	-	-
Duomenų keitimas	+	-	-

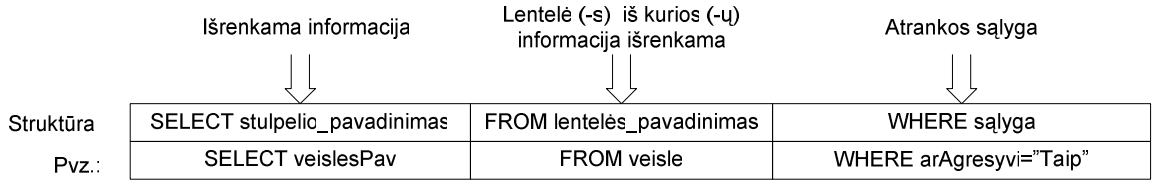
2.4. Užklausų struktūra

Išskirtinos trys užklausas sudarančios pagrindinės dalys: dalis, įvardijanti išrenkamą (ar keičiamą, šalinamą, kuriamą) informaciją, dalis, įvardijanti duomenų saugojimo elementą (-us), iš kurio (-ų) informacija turi būti išrenkama, bei dalis, kurioje apibrėžiama informacijos atrankos (ar keitimo, šalinimo) sąlyga. Paskutinioji dalis nėra būtina, tačiau praktikoje dažniausiai naudojama. Abstrakti SQL, XQuery ir XPath išrinkimo užklausų struktūra,

išskiriant paminėtas užklausos sudėtines dalis, pateikiama 3-5 paveiksluose, kartu su kiekvienos kalbos užklausų struktūros aprašymu.

2.4.1. SQL užklausos struktūra

SQL užklausos struktūroje (3 pav.) aiškiai atskiriamos pagrindinės užklausos dalys: informacijos išrinkimo dalyje įvardijami stulpelis (-ai), iš kurių informacija išrenkama, prasidedanti žodžiu SELECT, antroje užklausos dalyje – lentelė (-ės), iš kurių informacija išrenkama, prasidedanti FROM, o WHERE sakinyje nurodoma sąlyga, lemianti reikšmės pateikimą arba nepateikimą rezultatų aibėje.



3 pav. Abstrakti SQL užklausos struktūra ir pavyzdys[1]

Atrankos sąlyga gali būti atominė arba sudėtinė, tai yra lyginama gali būti pagal vieną arba daugiau negu vieną kriterijų. Atrankos sąlygos gali ir nebūti.

2.4.2. XPath užklausos struktūra

XPath duomenų dalies vietai XML dokumente nustatyti naudoja kelio išraiškas. Kelio išraiška – seka žingsnių, kurie atliekami pereinant nuo vieno XML mazgo prie kito mazgo ar mazgų aibės. Šis perėjimas, kitais žodžiais, tai XML dokumento medžio mazgų nagrinėjimas pradedant nuo aukštesnio lygio mazgo ir einant prie žemesnio, įvertinant, ar mazgą užklausą atrenka, ar ne. Žingsniai atskiriami pasviruoju brūkšniu („/“). Kiekvieną žingsnį sudaro trys komponentai: ašies žymeklis, mazgo testas ir predikatas. Ašies žymeklis – tai ašies pavadinimas, reiškiantis kontekstinio mazgo santykį su ankstesniojo žingsnio mazgu. Santykių pavyzdžiai – tėvo mazgas, vaiko mazgas. Mazgo testo atveju patikrinamas mazgo tipas. Predikatu apribojama išrenkamų reikšmių aibė. Rezultatas po kiekvieno kelio žingsnio – atrinktų mazgų seka. Kelio išraiškos reikšmė – mazgų seka, gaunama po paskutiniojo kelio žingsnio.

Kiekvienas žingsnis įvertinamas specialaus mazgo, vadinamo kontekstiniu mazgu, kontekste. Paprastai žingsniu gali būti bet kokia išraiška, grąžinanti mazgų seką. Viena svarbi

[1] Reliacinė išrinkimo operacija yra vadinama projekcija (angl. *projection*), o atrankos – selekcija (angl. *selection*) [11]. Siekiant suvienodinti naudojamas sąvokas, tiek XML užklausų kalbų, tiek SQL kalbos atveju projekcija bus vadinama išrinkimu, o selekcija – atranka.

žingsnio rūšis, vadinama ašiniu žingsniu, gali būti suvokiama kaip prasidedanti kontekstiniu mazgu ir judanti mazgų hierarchija tam tikra kryptimi, vadinama ašimi. Judant numatytąja ašimi, kiekviename žingsnyje atrenkami kriterijus atitinkantys mazgai. Atrankos kriterijus gali atrinkti mazgus pagal jų pavadinimus, poziciją kontekstinio mazgo atžvilgiu ar nurodant predikato dalyje mazgo atrinkimo reikšmę. XPath apibrėžia 13 ašių [22]. Įvertinus kelio išraišką, pagal kiekvieną žingsnį atrinkti mazgai patys tampa kontekstiniais mazgais kitame žingsnyje. Jeigu žingsnyje yra keli kontekstiniai mazgai, mazgo testas atliekamas kiekvieno šių kontekstinių mazgų atveju, o gauta mazgų seka apjungiama naudojant sąjungos operatorių, kad būtų suformuotas žingsnio rezultatas. Žingsnio rezultatas visada yra skirtingų mazgų (neturinčių dublikatų mazgo identifikatoriaus atžvilgiu) dokumente rasta eilės tvarka seka.

Kelio išraiškas galima rašyti arba nesutrumpinta, arba sutrumpinta ašių žingsnių sintakse. Nesutrumpintą ašies žingsnio sintaksę sudaro dviem dvitaškiais atskirti ašis ir atrankos kriterijus (4 pav. 1 pavyzdys.), o sutrumpintoje, priklausomai nuo ašies tipo, jos pavadinimas arba praleidžiamas, arba žymimas atitinkamais simboliais (4 pav., 2 pavyzdys).

Struktūra	/kelias_ik _elementc/išrenkama_informacija[atrankos_sąlyga]
Pvz 1	/child::veisles/child::veisle/child::veislesPav[child::arAgresyv=”Taip”]
Pvz 2	/veisles/veisle/veislesPav[arAgresyv=”Taip”]

4 pav. Abstrakti XPath užklausos struktūra ir pavyzdys, naudojant pilną ir sutrumpintą ašių žingsnių sintaksę

2.4.3. XQuery užklausos struktūra

XQuery užklausos struktūra yra laisvesnė už XPath ar SQL kalbų užklausų struktūras. XQuery užklausoje duomenų elemento vietai XML dokumente nustatyti naudojamos šešių rūšių XPath kelio išraiškos [10]. 5 pav. pateikta XQuery užklausa, naudojanti tik XPath kelio išraiškas.

Struktūra	XML_dokumento_pavadinimas/Xpath_išraiška
Pvz.:	doc(„veisles xml“)/veisles/veisle[arAgresyv =”Taip”]

5 pav. Abstrakti XQuery užklausos struktūra ir pavyzdys

Alternatyvi XQuery užklausos struktūra naudoja savybę, vadinamą *FLWOR* išraiška, suteikiančia galimybę atlikti iteracijas bei susieti kintamuosius su tarpiniais rezultatais (6 pav.) [22].

Struktūra		Pvz.:	
for	Kintamųjų paskelbimas kintamųjų reikšmių priskyrimas XML dokumentas Xpath kelio išraiškos	for	$i in doc(„veisles xml")/veisles/veisl
let			
where atrankos_sąlyga		where $ /arAgresyv ="Taip"	
return išrenkama_informacija		return veislesPav	

6 pav. Abstrakti XQuery užklausos struktūra, naudojant FLWOR išraišką, ir jos pavyzdys

Išskiriamos penkios *FLWOR* išraiškos dalys:

- „for“ ir „let“ sakiniai, generuojantys rikiuotą susietų kintamųjų kortežų seką, vadinama kortežų srautu (angl. *tuple stream*);
- „where“ sakinys, skirtas kortežų srautui atrinkti; kaip ir SQL atveju, sakinys gali būti atominis arba sudėtinis;
- „order by“ sakinys, kuris gali būti naudojamas kortežų srautų eiliškumui pakeisti;
- „return“ sakinys, sudarantis FLWOR išraiškos rezultatą.

2.5. Užklausos rezultato semantika

SQL užklausa tiek išrenka informaciją iš iš stulpelių sudarytos lentelės, tiek ir grąžina rezultatą iš stulpelių sudarytos lentelės forma. Grąžinamą lentelę sudaro stulpeliai, nurodyti išrenkamos informacijos dalyje. XQuery ir XPath išrenka informaciją iš XML dokumentų, o grąžina seką – surikiuotą rinkinį iš nulio ar daugiau elementų, kuriais gali būti XML dokumento mazgai arba atominės reikšmės. XQuery grąžinama seka – nebūtinai XML dokumentas, tačiau ji optimizuota ir dažniausiai sudaroma taip, kad jos rezultatas būtų XML dokumentas [4].

2.6. Užklausų savybės, operatoriai ir išraiškos

Svarbi ne tik užklausų struktūra, tačiau ir papildomos savybės, nuo kurių labai priklauso užklausų sandara ir formavimas. Aptartos duomenų struktūros turi specifinių bruožų, kurių nevienodumas lemia ir užklausų veiksenos bei struktūrinį nevienodumą. Toliau bus atlikta lyginamoji užklausų kalbų savybių bei išraiškų analizė pagal šiuos požymius: užklausų kalbų galimybę išlaikyti pirminę duomenų elementų tvarką, nulinių reikšmių apdorojimo pobūdį, kvantifikuotas (*kvantifikavimo* sąvoka apibrėžta 2.6.3 skyriuje) išraiškas, neigimo išraiškas, duomenų elementų sujungimo mechanizmus, agregatines, rekursines funkcijas, vidinių užklausų naudojimą, aibių operacijas bei operatorių tipus.

2.6.1. Pirminės eilės tvarkos išlaikymas ir rikiavimas

Pirminės eilės tvarkos išlaikymas – užklausų savybė atrinkti rezultatus tokia eilės tvarka, kokie jie buvo rasti duomenų struktūroje. XPath bei XQuery išlaiko pirminę eilės tvarką (jeigu specialiai nenurodoma kitokia rezultatų grąžinimo tvarka), tuo tarpu reliacinėms sistemoms nebūdinga vidinė tvarka, taigi tai netaikoma ir SQL [17].

Atrinkti rezultatai gali būti rikiuojami didėjančia arba mažėjančia tvarka pagal kurį nors duomenų struktūros elementą, t. y. stulpelį SQL atveju ar mazgą ar jo atributą XML užklausų atveju. SQL ir XQuery užklausų kalbos suteikia rikiavimo galimybę ir abiejų kalbų rikiavimo kriterijus nurodomas naudojant *ORDER BY* sakinį. Kiekviena užklausa gali neturėti rikiavimo kriterijų, turėti vieną arba daugiau negu vieną rikiavimo kriterijų.

2.6.2. Nulinė reikšmė

Nulinė (*null*) reikšmė – reikšmė, skirta reikšmės nebuvimui pažymėti. SQL turi *NULL* reikšmę, kurios reikia, kadangi kiekviena lentelės eilutė turi turėti vienodą stulpelių rinkinį. XQuery ir XPath tiesioginio analogo neturi, kadangi XML, jeigu informacija nepateikiama, elementas gali būti tiesiog tuščias arba jo ir visai nebūti. Artimiausias nulinės SQL reikšmės analogas – tuščia XQuery ar XPath seka.

SQL nulinė reikšmė yra apdorojama kitaip, negu paprastos reikšmės: nulinės reikšmės nėra tarpusavyje lygios, jų nepaisoma vykdant agregatines funkcijas bei yra specifiškai apdorojamos sujungimų, kurie aptariami 2.6.6 poskyryje, atvejais [13].

2.6.3. Kvantifikuotos išraiškos

Visos trys užklausų kalbos palaiko kvantifikuotas išraiškas. Kvantifikacija – konstruktas, pažymintis predikato (kintamojo) galiojimo ribas [13]. Kvantifikuotos išraiškos rezultatas visada yra loginis „taip“ arba „ne“. Yra dvi pagrindinės kvantifikacijos rūšys: universali, kurios rezultatas yra loginis „taip“, jeigu visos atrenkamos reikšmės tenkina nurodytą sąlygą, ir egzistencinė, kurios rezultatas yra loginis „taip“, jeigu egzistuoja bent viena nurodytą sąlygą tenkinanti reikšmė. XPath/XQuery kvantifikavimo sintaksė naudoja *some*/ *every* ir *satisfies* kombinacijas, o SQL – *SOME*, *EVERY*, *ANY*, kaip pažymėta 3 lentelėje.

3 lentelė. Kvantifikacijos sintaksė

Kvantifikacijos tipas/Kalba	SQL	XPath/XQuery
Egzistencinė kvantifikacija	SOME	some <kintamasis> satisfies <atrankos sąlyga>
Universali kvantifikacija	EVERY, ANY	every <kintamasis> satisfies <atrankos sąlyga>

2.6.4. Agregatinės funkcijos ir grupavimas

Tiek SQL, tiek XPath/XQuery turi galimybę vykdyti agregatines funkcijas, kurios apdoroja grupę skaitinių reikšmių, tačiau grąžina vieną reikšmę – skaitinį rezultatą. Galimos agregatinės funkcijos pateiktos 4 lentelėje. XPath ir XQuery agregatinės funkcijos sutampa ir yra taikomos atominėms reikšmėms; jeigu, sudarant užklausą, nurodoma, kad agregatinė funkcija turi įvertinti seką mazgų, automatiškai išrenkama jų reikšmė.

4 lentelė. Agregatinių funkcijų tipai ir jų naudojimas užklausų kalbose[2]

Funkcija	**Aprašas**	**SQL**	**XPath**	**XQuery**
AVG	Vidutinė reikšmė	Vidutinė stulpelio reikšmė	Vidutinė atominių reikšmių reikšmė	
COUNT	Rastų reikšmių skaičius	Stulpelių eilučių skaičius (be nulinės reikšmės)	Atominių reikšmių skaičius, neįskaitant tuščių elementų	
MAX	Maksimali rasta reikšmė	Maksimali stulpelio reikšmė	Maksimali atominė reikšmė	
MIN	Minimali rasta reikšmė	Minimali stulpelio reikšmė	Minimali atominė reikšmė	
SUM	Rastų reikšmių suma	Stulpelio reikšmių suma	Rastų reikšmių suma	

Su agregatinėmis funkcijomis naudojamas grupavimas, t. y. galimybė jungti duomenis į grupes, kurioms ir taikomos agregatinės funkcijos. SQL grupavimą atlieka naudodama *GROUP BY* konstruktą. Tiesioginio XQuery/ XPath atitikmens neturi, tačiau XQuery grupavimas gali būti atliekamas naudojant įdėtas (angl. *nested*) užklausas; XPath galima grupę duomenų perduoti konkrečiai agregatinei funkcijai kaip argumentus, tačiau tiesioginio grupavimo konstrukto taip pat nėra.

2.6.5. Užklausų įdėjimas

Dar viena užklausoms būdinga savybė - vienos užklausos įdėjimas į kitą (angl. *nesting*). Šią galimybę taip pat teikia tiek SQL, tiek XPath bei XQuery. Vidinė – pagrindinės užklausos kurioje nors dalyje naudojama – užklausa turi grąžinti rezultatus, kuriems keliama sąlyga – jie turi būti tokios struktūros ir tipo, kad grąžinamą reikšmę (ar reikšmes) galėtų apdoroti pagrindinė užklausa. 5 lentelėje pateikta informacija, pažyminti, kurioje kiekvienos konkrečia užklausų kalba sudarytos užklausos dalyje gali būti naudojama vidinė užklausa.

[2] Konkrečiose SQL realizacijose pateikiama daugiau agregatinių funkcinių, tačiau [1-2] standartas apibrėžia tik 4-oje lentelėje nurodytąsias, todėl į kitas agregatines funkcijas neatsižvelgiama.

5 lentelė. Užklausų įdėjimas sudėtinių užklausos dalių atžvilgiu

Kalba/Užklausos dalis	Užklausos rezultatas	Duomenų elementai, iš kurių rezultatas išrenkamas	Atrankos sąlyga
SQL	-	+	+
XPath	Visur, kur galima naudoti išraiškas		
XQuery	+	Pagal XPath	+

2.6.6. Sujungimai

Reliacinėse duomenų bazėse viena iš dažniausiai naudojamų operacijų – sujungimai (angl. *joins*), naudojami rezultatų aibei suformuoti, išrenkant duomenis iš vienos ar kelių lentelių. ANSI standartas apibrėžia, kad sujungimai realizuojami naudojant *JOIN* sakinį, kuris turi kelias variacijas [12]. XPath/XQuery užklausose taip pat realizuota sujungimo operacija, skirta informacijai iš skirtingų to paties dokumento ar kelių dokumentų dalių apjungti. XML užklausų kalbose sujungimai neturi tiesioginio SQL *JOIN* atitikmens, o įgyvendinami (XQuery) naudojant *for/let* kombinaciją arba (XPath) naudojant teta sujungimą (tokiu būdu realizuojamas tik tam tikras sujungimo atvejis) arba *for* iteracijas [12].

2.6.7. Operacijos su aibėmis

Užklausos gali atlikti aibių operacijas. SQL realizuoja tris tokias operacijas: sąjungą (angl. *union*), sankirtą (angl. *intersect*) ir skirtumą (angl. *minus*). Operacijoms su lentelėmis atlikti keliama sąlyga, kad abi lenteles turi sudaryti vienodas stulpelių skaičius bei kiekvieno lentelių stulpelio duomenų tipai turi sutapti. Taip pat kiekvienas sekų operatorius turi du režimus: vieną – kuomet į rezultatą įtraukiami besidubliuojantys įrašai, kitą – kuomet dublikatai pašalinami. XPath ir XQuery palaiko visas tris aibių operacijas: sąjungą, sankirtą ir skirtumą (šių kalbų atveju angl. vadinamą *except*): sąjungos operatorius pateikia seką, kurią sudaro visi operandų mazgai; sankirtos operatorius pateikia seką, kurią sudaro visi mazgai, kurie yra abiejuose operanduose; skirtumo operatorius sudaro seką, kurią sudaro visi mazgai, kurie yra pirmajame operande, bet kurių nėra antrajame.

2.6.8. Funkcijų perklotis, rekursija ir vartotojo apibrėžtos funkcijos

SQL suteikia galimybę naudoti perklotas (angl. *overloaded*) funkcijas [13] ir parenka funkciją pagal operandų duomenų tipus. XPath ir XQuery, priešingai, nepalaiko funkcijų perklojimo. Kiekviena XPath ir XQuery funkcija (identifikuojama pagal funkcijos pavadinimą ir parametrų skaičių) turi gerai apibrėžtą parašą, ir funkcinis kreipinys bando pakeisti jos argumentus į numatomus tipus. Ši strategija suderinama su XQuery principu

manipuliuoti netipizuotais duomenimis ir keisti juos į numatytą tipą atsižvelgiant į duomenų naudojimą.

SQL ir XQuery teikia rekursinių funkcijų vykdymo galimybę bei leidžia sudaryti vartotojo apibrėžtas funkcijas.

2.6.9. Lyginimo operatoriai

XQuery turi dvi grupes lyginimo operatorių, iš kurių išskiriamos reikšmių lyginimo (operatoriai, kurie lygina pavienes reikšmes) bei bendrųjų lyginimo (operatoriai, kurie manipuliuoja su reikšmių grupėmis ir ieško bent vienos atitinkančios reikšmės tarp šios grupės reikšmių) operatorių grupės. Reikšmių lyginimo operatoriai traktuoja, kad lyginamos reikšmės atitinka viena kitą, jeigu bent vienas rezultatas sutampa su lyginamąja reikšme. Tuo tarpu bendrieji lyginimo operatoriai traktuoja, kad lyginamos reikšmės atitinka viena kitą, jeigu su lyginamąja reikšme sutampa vienas ir tik vienas rezultatas. SQL atveju reikšmės išraiška visada grąžina vieną reikšmę.

2.6.10. Loginiai operatoriai

Loginiai SQL operatoriai – *ir*, *arba*, *ne* (angl. and, or ir not) – yra trireikšmiai dėl nulinės reikšmės egzistavimo (lyginant nulinę reikšmę su bet kokia kita reikšme, grąžinama „nežinoma“ loginė reikšmė). Tuo tarpu XQuery ir XPath naudoja tradicinę dviejų reikšmių logiką. Šių užklausų kalbų operatoriai *and* ir *or* tuščią seką traktuoja kaip false.

Operatorių *ne* galima išskirti kaip atskirą atvejį. XML kalbų terminais neigimas reiškia galimybę neįtraukti duomenų, atsižvelgiant į struktūrą. Kitaip sakant, tai galimybė užklausai išrinkti elementus, kurie neturi tam tikrų vaikinių elementų ar atributų. Neigimui atlikti XQuery ir XPath naudojama *not*, kuri yra daugiau funkcija negu operatorius. Ji invertuoja loginę reikšmę, *true* pakeisdama į *false* ir atvirkščiai. SQL kalboje neigimas reiškia, kad užklausa suranda ir atrenka visas rastiems rezultatams nepriklausančias reikšmes. SQL kalboje neigimas turi analogišką prasmę – į rezultatų aibę neįtraukiamos atrankos sąlygą atitinkančios eilutės, o neigimas atliekamas naudojamas *not* operatorių.

2.6.11. Aritmetiniai operatoriai

Kaip ir SQL, XPath ir XQuery teikia įprastus aritmetinius operatorius - +, -, *, div ir mod. Dalybos operatorius pavadintas *div*, siekiant išskirti jį nuo pasvirojo brūkšnio, naudojamo kelio išraiškose.

Aritmetiniai operatoriai apibrėžiami su skaitinėmis reikšmėmis (arba agregatinių funkcijų, sekų ar skaitinių reikšmių atvejais). Jeigu aritmetinio operatoriaus operandas – mazgas, automatiškai išrenkama jo reikšmė su esamu tipu.

Aritmetinių operatorių veiksmai su tuščiomis sekomis – svarbus specialus atvejis. XQuery tuščia seka kartais naudojama trūkstamai ar nežinomai informacijai pavaizduoti, panašiai kaip ir nulinė reikšmė reliacinėse sistemose. Dėl šios priežasties +, -, *, div ir mod operatoriai apibrėžti grąžinti tuščią seką, jeigu bent vienas iš jų operandų – tuščia seka.

2.6.12. Lyginimo rezultatai

Atlikto užklausų kalbų lyginimo rezultatai pagal visus nagrinėtus kriterijus pateikti 6 lentelėje.

6 lentelė. Užklausų kalbų savybių palyginimas

Lyginimo kriterijus		**SQL**	**XPath**	**XQuery**
Užklausų rezultato semantika		Išrenkami duomenys iš stulpelių sudarytos lentelės forma	Atominių reikšmių seka	Atominių reikšmių arba XML dokumento mazgų seka
Operatoriai	Lyginimo	aritmetiniai (>, <, >=, <=, =, <>) IN between like	aritmetiniai (>, <, >=, <=, =, <>); IN, between atitikmenis galima realizuoti naudojant sudėtinę sąlygą	
	Loginiai	And, Or, Not	And, Or; Not funkcija	
	Aritmetiniai	+, -, *, /	+, -, *, div, mod	
Pirminės eilės tvarkos išlaikymas rezultate		Neišlaiko	Išlaiko	
Kvantifikuotos išraiškos		Egzistencinė ir universali kvantifikacija	Egzistencinė ir universali kvantifikacija	
Nulinė reikšmė		Turi nulinę reikšmę	Neturi	Neturi; artimiausias analogas – tuščia seka
Agregatinės funkcijos		Teikiama galimybė	Teikiama galimybė	
Grupavimas		Teikiama galimybė	Nėra galimybės; galima perduoti reikšmes kaip agregatinės funkcijos parametrus	Palaiko naudojant įdėtas užklausas
Užklausų įdėjimas (angl. *nesting*)		Teikiama galimybė	Teikiama galimybė	
Sujungimai		Palaiko įvairių rūšių sujungimus	Tiesiogiai nepalaiko; atskiram sujungimo atvejui galima naudoti theta sujungimą	Pasiekiamas analogiškas efektas naudojant for-let kombinaciją
Operacijos su aibėmis		Sąjunga, sankirta, skirtumas	Sąjunga, sankirta, skirtumas	
Funkcijų perklotis		Teikiama galimybė	Nėra galimybės	
Rekursija ir vartotojo apibrėžtos funkcijos		Teikiama galimybė	Teikiama galimybė	

2.7. Užklausų vykdymas

Kaip jau buvo minėta, visos trys užklausų kalbos teturi vieną bendrą – informacijos išrinkimo – galimybę. Tokių užklausų paskirtis – grąžinti pagal nurodytus kriterijus atrinktų duomenų rezultatą. Ši užklausų savybė išnaudojama būtent paieškos sistemose. Taigi tikslinga būtų modeliu apibendrintus analizės rezultatus pritaikyti šio tipo sistemoms.

Duomenų išrinkimo užklausų naudojimo sritis yra platesnė: tokios užklausos naudojamos, pavyzdžiui, vykdant mainus tarp skirtingų sistemų ar formuojant iš esamų duomenų įvairios struktūros dokumentus, ataskaitas. Tačiau paieškos sistemas kaip modelio taikymo objektą tikslinga pasirinkti, kadangi iki šiol dar nėra rekomendacijų, metodų, skirtų paieškos sistemų kūrimui apibrėžti, o pačios paieškos sistemos yra plačiai naudojamos.

Kaip minėta, atsižvelgiant į XPath ir XQuery savybių visumą sudarant bendrinį modelį, naujų elementų sudarymo galimybė taip pat būtų teikiama tiek SQL, tiek XML užklausų kalbų, todėl šią operaciją galima priskirti prie bendrų, tačiau šiame darbe bus apsiribota paieškos sistemų kūrimu.

Paieškos sistemose paprastai vartotojui pateikiama sąsaja, kurioje šis įveda formuojamos užklausos kriterijus, nurodo rezultatų rikiavimo tvarką ir pan. Taigi realiai vartotojas, nesusidurdamas tiesiogiai su konkrečios užklausų kalbos konstruktais, suformuoja užklausą, ir tokia užklausa gali būti skirta tiek reliaciniam, tiek XML duomenų šaltiniui.

Remiantis šia išvada, naudinga turėti tokį modelį, kuris būtent tiktų tiek bet kurios nagrinėjamos užklausų kalbos užklausoms reikšti, tiek būtų jį galima tiesiogiai susieti su vartotojui pateikiama vizualia užklausos išraiška. Būtent šioms dviems – paieškos sąsajos ir užklausų sudarymo ir vykdymo – sritims modelis ir skiriamas.

2.8. Esamų paieškos sistemų teikiamos galimybės bei numatomos paieškos galimybės

Siekiant įvertinti daugumą paieškos sistemos poreikių tenkinančias funkcines galimybes, atliktas egzistuojančių sudėtingesnių paieškos sistemų teikiamų galimybių palyginimas. Lyginamoji informacija pateikta 7-ojoje lentelėje, kurioje išvardytas įprastinių ir sudėtingesnių paieškų galimybių sąrašas bei pažymėta, ar ši galimybė kiekvienoje konkrečioje paieškos sistemoje teikiama, ar ne. Sukurtas užklausų modelis turi taip pat būti tinkamas šioms galimybėms įgyvendinti, taigi lentelėje pateikiama informacija apie numatytus įgyvendinimo būdus. Be joje pateiktų sistemų savybių, taip pat, formuojant paieškos kriterijų, gali būti naudojami papildomi lyginamieji operatoriai, apibūdinti 2.6.9 skyriuje.

7 lentelė. Paieškos sistemų galimybių palyginimas[3]

Savybė/Paieškos sistema	**IEEE**	**UWFL**	**EBSCO**	**Amazon**	**Paieškos sistema, sukurta pagal siūlomą modelį**
Pasirenkamas paieškos kriterijų sąrašas	+ pasirenkamų reikšmių sąraše	+ pasirenkamų reikšmių sąraše	+ pasirenkamų reikšmių sąraše	-	Paieškos kriterijai pateikiami pasirenkamų reikšmių sąraše
Galima pasirinkti – vykdyti tikslią paiešką ar ne	-	+	-	+ kai kuriems kriterijams galima pasirinkti	-
Paieškos laukai jungiami pasirenkamais loginiais operatoriais	+	+	+	-	Loginis operatorius pasirenkamas iš sąrašo
Tekstiniame lauke formuojama paieška	+	-	-	+	+
Pasirenkamas rikiavimo kriterijus ir rikiavimo tvarka	+	-	-	+	Iš sąrašo pasirenkamas kriterijus, nurodoma rikiavimo tvarka
Paieška tarp rezultatų	+	+	+	-	+
Rezultatų kiekio apribojimas	+	-	-	-	Pasirenkamos fiksuotos reikšmės
Nurodoma paieškos sritis	+	+	+	+	Paieškos objektai pateikiami pasirenkamų reikšmių sąraše
Įvertinamas rezultato tinkamumas	-	-	+	-	Pateikiamas metodas tinkamumui įvertinti

2.9. Bendro modelio savybės, reikalingos paieškai

6 ir 7 lentelėse pateikti užklausų kalbų ir paieškos sistemų savybių tyrimo rezultatai. Norint, kad kuriamas modelis būtų tinkamas paieškos sistemoms kurti, jis turi perteikti tokias bendras užklausų kalbų savybes, kurios naudojamos paieškos sistemose, kitaip tariant, turi būti parinktos bendros užklausų kalbų savybės, naudojamos paieškos sistemų savybėms

[3] *Pastaba. Naudotų paieškų adresai internete: IEEE – http://ieeexplore.ieee.org/search/advsearch.jsp , UWFL – University of West Florida Library, http://wf.aleph.fcla.edu/F/CKU1U4IDXRS6T41C7X649AMRK3D8CFJRK2G6V3H5424V9IJPDN-00492?func=file&file_name=find-d EBSCO – http://search.ebscohost.com, Amazon - http://www.amazon.com/b/ref=sv_b_0/103-6390796-6735032?ie=UTF8&node=241582011 .*

įgyvendinti. Struktūrines savybes, apimančias paieškos objektus (sritis), kriterijus bei duomenų elementus, turi įgyvendinti bet kokia užklausa. Kitos reikalingos savybės yra šios:

- paieškos kriterijų skaičius;
- rikiavimo kriterijai;
- rikiavimo tvarka;
- loginiai operatoriai;
- lyginamieji operatoriai.

Papildomos savybės, tiesiogiai nesusijusios su užklausų formavimu

- Paieškos tarp rezultatų savybė susijusi su duomenų elementais, iš kurių išrenkami duomenys, todėl į ją turi būti atsižvelgta kaip į alternatyvius duomenų elementus.
- Tiesioginio rezultatų kiekio apribojimo konstrukto nėra nei SQL, nei XQuery/XPath kalbose. SQL kalbų realizacijos teikia savo būdus, skirtus šiam kiekiui apriboti, o XQuery atveju tai galima netiesiogiai atlikti skaičiuojant rezultato eilučių skaičių ir atmetant ribą viršijančias eilutes, tačiau tai taipogi netiesioginė galimybė.

2.10. Modeliavimo forma ir priemonės

Informacinių sistemų (IS) projektavime svarbų vaidmenį atlieka automatizuoto projektavimo (CASE) priemonės, kurių daugelis naudoja vieningą modeliavimo kalbą UML, skirtą objektinėms sistemoms vaizduoti, specifikuoti, kurti bei dokumentuoti. Ši kalba, grafiškai pateikianti elementus, suteikia vaizduojamiems elementams aiškumo, taip turėdama pranašumą prieš daugelį kitų modeliavimo būdų, pavyzdžiui formalųjį modeliavimą. Be to, būdama standartu, UML kalba suteikia galimybę didesniam projektuotojų ratui tiksliai interpretuoti diagramų elementų prasmę bei, prireikus, atlikti pakeitimus modelyje. Dėl šių priežasčių modeliui kurti numatyta naudoti UML kalbą.

2.10.1. UML klasių diagrama

Siekiant modeliavimui naudoti UML kalbą, tikslinga aprašyti pagrindinius jos elementus ir išskirti jų poaibį, turintį prasmę sudarant numatomą modelį.

UML standartas (2.0 versija), kurį kuria OMG (angl. *Object Management Group*) – atviras organizacijų konsorciumas, apibrėžia 13 diagramų tipų: klasių, sekų, objektų, panaudojimo atvejų, paketų, diegimo, būsenų, veiklos, bendradarbiavimo, komponentų, laiko, sudėtinės struktūros ir būsenų diagramas [19].

UML diagramos skirstomos į statines, dinamines ir architektūrines diagramas. Statinė diagrama vaizduoja sistemos struktūrą, kurią sudaro smulkesnės fizine struktūra susijusios posistemės. Dinaminės diagramos vaizduoja veiksminio pobūdžio sąveiką tarp struktūrinių elementų, o architektūrinės diagramos skirsto sistemą į vykdomus (angl. *running*) ir vykdomuosius (angl. *executable*) komponentus.

Kadangi modeliui svarbus struktūrinis užklausų aspektas, bus trumpai apibūdintos pagrindinės struktūrinio tipo elementų sąvokos. UML apibrėžia trijų rūšių struktūrines diagramas: klasių, objektų ir panaudojimo atvejų. Panaudojimo atvejų diagrama naudojama funkciniams sistemų reikalavimams specifikuoti; objektų diagrama pavaizduoja sistemos objektus tam tikru laiko momentu – egzempliorius; klasių diagramoje vaizduojami sistemos objektų tipai (klasės) bei tarp jų egzistuojantys statiniai ryšiai (asociacijos). Modeliui sudaryti bus naudojama klasių diagrama, kadangi modelis turi perteikti užklausų elementus bei tarp jų esančius ryšius.

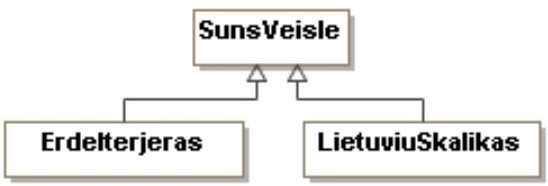
Klasių diagrama neparodo laikinos informacijos, ji aprašo tik klasifikaciją. Be minėtų objektų tipų ir ryšių, klasių diagramoje taip pat vaizduojamos klasių savybės ir operacijos, kurias gali atlikti klasės tipo objektai (t. y. egzemplioriai), bei apribojimai.

Klasių diagramos pagrindinis struktūrinis vienetas – klasė. Klasės gali būti jungiamos į paketus. Klasės aprašą sudaro klasės pavadinimas, atributai ir operacijos. Atributu išreiškiama klasės savybė. Yra galimybė, kaip ir programavimo kalbose, priskirti atributui tipą, numatytąsias reikšmes, nurodyti kardinalumą ar apibrėžti jo matomumą.

Tarp klasių, kaip minėta, gali egzistuoti ryšiai. Išskiriami kelių tipų ryšiai:

- asociacija,
- apibendrinimas,
- priklausomumas,
- agregavimas,
- rekursyvus ryšys,
- kompozicija.

7 pav. pateiktas UML klasių diagramos pavyzdys, kuriame pavaizduota abstrakti klasių struktūra bei tarp klasių egzistuojantys ryšiai.



7 pav. UML klasių diagramos pavyzdys

2.10.2. Modeliavimo priemonės pasirinkimas

UML diagramoms kurti egzistuoja aibė priemonių, tokių kaip „Rational Rose“, „ArgoUML“, „SmartDraw“ paketai. „MagicDraw“ paketas – UML priemonė, palaikanti visą UML 2.0 standartą, programų tekstų tiesioginę ir atvirkštinę inžineriją į vienas dažniausiai vartojamų programavimo kalbų (pvz., Java, C++). Šios funkcinės galimybės svarbios tiek praplečiant UML kalbos elementų prasmę, tiek naudingos esant poreikiui papildyti patį įrankį nauju funkcionalumu, kas ir lemia priemonės pasirinkimą modeliui kurti.

2.11. Analizės išvados

1. Nustatyta, kad iki šiol dar nėra atliktos išsamios reliacinių ir XML šaltinių standartinių užklausų kalbų analizės.
2. Bendras užklausų kalbų modelis, kuris neperteiktų su konkrečios kalbos konstruktais susijusių savybių, suteiktų galimybę formuoti užklausas abstraktesniame lygyje.
3. Sulyginti reliacinis ir XQuery/XPath duomenų modeliai, nes nuo jų labai priklauso užklausų struktūra ir pagrindinės savybės: duomenų modeliai nusako pagrindinę duomenų struktūrą, tipizavimo mechanizmus, duomenų elementų identifikatorius bei ryšius tarp elementų.
4. Nustatyta, kad visoms joms būdinga tik viena – informacijos išrinkimo – funkcinė savybė, taigi tolesnė analizė buvo vykdoma informacijos išrinkimo požiūriu.
5. Apibrėžta bendra užklausoms būdinga struktūra bei nustatyta, kad užklausų kalbos turi daug panašių – tiek struktūrinių, tiek funkcinių – savybių bei požymių: visos trys – SQL, XPath ir XQuery – užklausų kalbos turi savo būdą nurodyti išrenkamiems elementams, esybėms (duomenų objektams), iš kurių duomenys išrenkami, bei rezultatų atrankos (filtravimo) galimybę; visos kalbos leidžia naudoti vidines užklausas, agregatines, rekursines funkcijas, teikia elementų sujungimo, grupavimo, rikiavimo galimybes.
6. Remiantis panašumu tarp duomenų išrinkimo užklausų ir paieškos sistemų, numatyta, kad bendrą modelį galima būtų naudoti kuriant paieškos sistemas.
7. Aptartos esamos užklausų modeliavimo galimybės ir pasirinktas grafinis modeliavimo būdas naudojant UML standartą, kadangi UML diagramos suteikia modeliui aiškumo, o standartas didina modelio plėtimo bei galimybę bei suteikia nepriklausomumą nuo konkrečios modeliavimo priemonės.

Numatyta tolesnė veikla

Atlikus užklausų kalbų – SQL, XPath ir XQuery – analizę, buvo surinkta pakankamai duomenų bendriniam modeliui sukurti. Taigi bus kuriamas modelis, o užtikrinti jo panaudojimui praktikoje bus sukurta metodika paieškos sistemų kūrimui automatizuoti. Taip pat numatoma sukurti programinį prototipą, pademonstruojantį, kaip modeliu galima remtis kuriant paieškos sistemas.

3. UŽKLAUSŲ METAMODELIS

Remiantis atliktos analizės rezultatais, formuojamas užklausų metamodelis. Įvesta metamodelio sąvoka, kadangi modelis saugo metaduomenis, tai yra duomenis apie užklausas. Metamodelis turi apimti struktūrines bei su užklausų vykdymu susijusias savybes, todėl išskiriamos dvi modelio dalys. Taip pat, siekiant parodyti modelio pritaikymo principus, pasirinktai dalykinei sričiai bus sudaromas koncepcinis modelis ir, panaudojant metamodelio elementus, pažymima užklausoje dalykinėje srityje naudojama informacija. Galiausiai pateikiama metodika, skirta paieškos sistemų vartotojo sąsajai generuoti bei užklausoms formuoti.

3.1. Užklausų metamodelio sudarymas

Numatyta užklausų modelį sudaryti naudojant UML klasių diagramą, kurios naudojimo pagrįstumas pateiktas 2.10 skyriuje, todėl visų pirma bus parinkti konkretūs klasių diagramos elementai, reikalingi formuojant modelį. Klasių diagrama saugo informaciją apie tam tikros srities struktūrines ir elgsenos savybes. Modelis turi perteikti struktūrines užklausų savybes, todėl atitinkamai bus naudojamos tik struktūrinės klasių diagramos savybės. Struktūrinėms savybėms priklauso klasės, atributai, ryšiai bei ryšių kardinalumai [25].

Formuojant užklausas, išskirtinos trys pagrindinės užklausų dalys:

- išrenkama informacija, kuri bus vadinama r*ezultatu*,
- atrankos sąlyga, kuri bus vadinama *sąlyga*,
- požymis, pagal kurį išrenkama informacija išrikiuojama, kuris bus vadinamas *rikiavimo kriterijumi*.

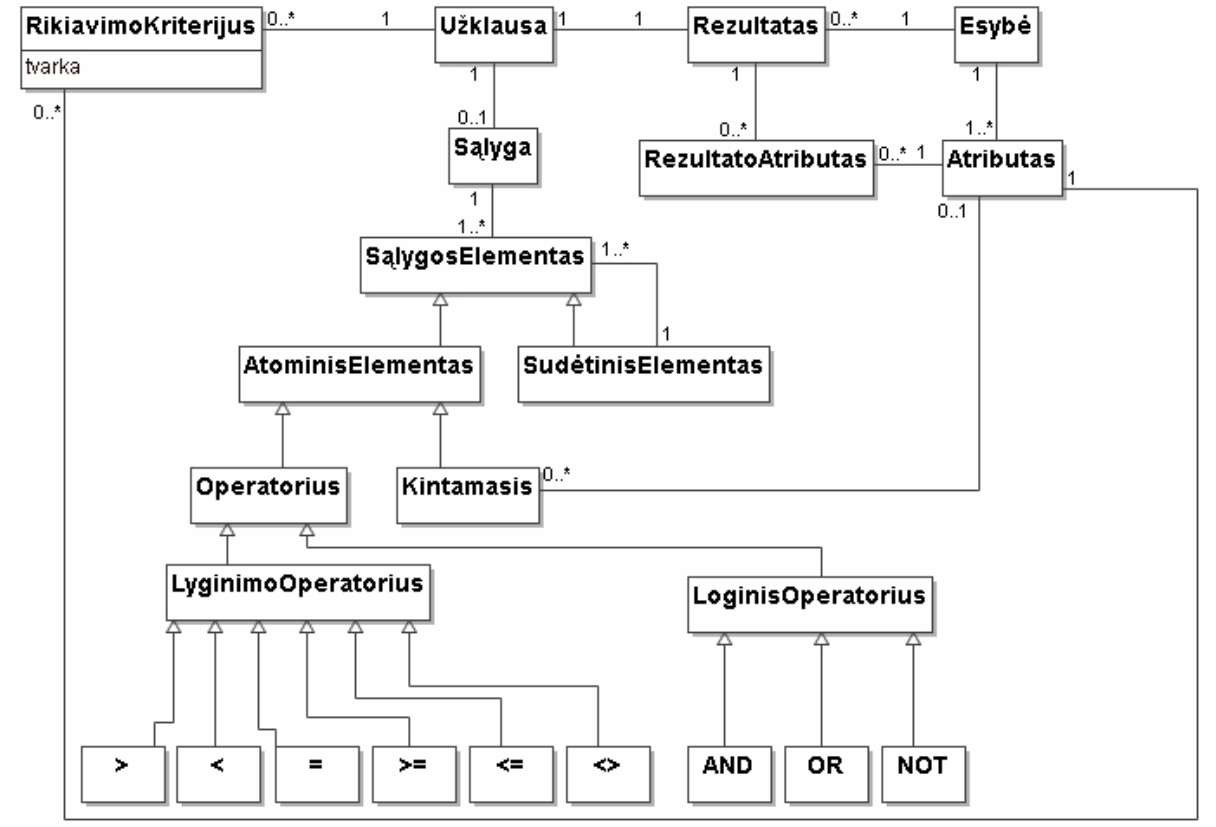
Kiekviena iš šių užklausos dalių bus vaizduojama klase. Užklausos dalys susijusios, todėl šiems ryšiams pažymėti naudojami ryšiai – asociacijos ir agregatiniai.

Užklausos rezultatas – tai esybės atributų aibė reliacinės duomenų bazės atveju arba mazgo atributų aibė arba mazgo elementų aibės tekstinis turinys (angl. *text content*) XML dokumento atveju. Diagramoje tiek esybes, tiek mazgai apibendrinami klase „Esybė“, o esybės ar mazgo atributai arba mazgo elementai – klase „Atributas“.

Kita užklausos sudėtinė dalis, sąlyga – tai užklausos atrankos kriterijų aibė, turinti bent vieną elementą, kuris pažymimas klase „SąlygosElementas“. Sąlygos elementą sudaro operatoriais jungiami esybės atributai. Operatorius galima vaizduoti klase „Operatorius“, o sąlygos atributus - klase „Kintamasis“. Abiems šiems sąlygos elementams apibendrinti bus naudojama klasė „AtominisElementas“, o ryšiui tarp „AtominisElementas“ bei „Operatorius“ ir „Kintamasis“ klasių pažymėti – UML klasių agregavimo ryšys. Kiekvieną sąlygos elementą

sudaro aibė operatorių ir kintamųjų – jie bus jungiami į klasę „SudėtinisElementas“. Taigi sąlygos elementas gali būti arba atominis, arba sudėtinis, ir šiam ryšiui vėlgi naudojamas agregavimo ryšys. Sudėtinio elemento dalimi yra atominiai elementai, todėl panaudojamas rekursinis ryšys tarp „SąlygosElementas“ ir „SudėtinisElementas“ klasių. Atominio sąlygos elemento operatoriumi gali būti loginiai, aritmetiniai ar papildomi operatoriai „IN“ (reiškiantis, kad užklausos kintamasis lyginamas su viena iš sąraše nurodytu reikšmių) ir „between“ (reiškiantis, kad užklausos kintamasis gali įgyti reikšmes iš nurodytos srities, intervalo). Visi šie operatoriai bei galutinis užklausos modelis pateikti 8 paveiksle.

Galiausiai rikiavimo kriterijų galima vaizduoti klase, kuriai priskiriamas atitinkamas pavadinimas „RikiavimoKriterijus“, o pats rikiavimo kriterijus – tai esybės atributas. Užklausos rezultatų rikiavimas gali būti organizuojamas didėjančiai arba mažėjančiai, todėl šis rikiavimo tvarkos požymis prilyginamas UML klasės atributui.



8 pav. Užklausų metamodelis

3.2. Užklausų posistemės metamodelio sudarymas

Sudarytame modelyje dar neatsižvelgta į šiuos aspektus:

- kiekvienas vartotojas formuoja savo užklausą, t. y. konkretus užklausos egzempliorius siejamas su jos autoriumi, ir šiam pateikiami konkretaus užklausos egzemplioriaus rezultatai;

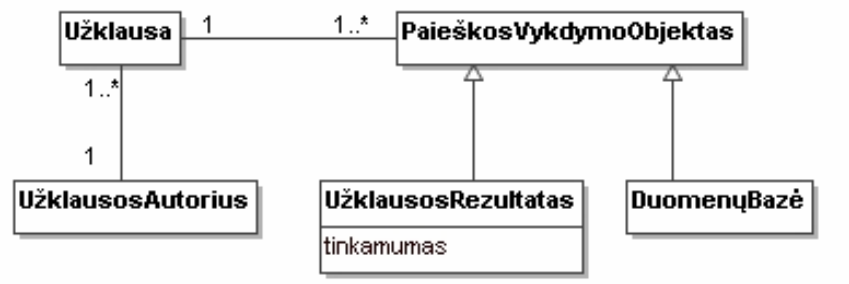
- sykį atlikus užklausą ir gavus jos rezultatus, kita užklausa (ar kitos užklausos) gali būti vykdoma (vykdomos) tarp gautų paieškos rezultatų. Taigi turi būti saugoma kiekvienos užklausos rezultatų versija;
- skaičiuojamas kiekvienos užklausos rezultatų atitikimas nurodytiems kriterijams.

Papildant modelį šiais aspektais, formuojamas užklausų posistemio duomenų modelis.

Užklausos autorius-vartotojas gali pateikti vieną ar kelias užklausas. Šie objektai atitinkamai vaizduojami klasėmis „UžklausosAutorius“ ir „Užklausa“. Čia „Užklausa“ – tai užklausų metamodelio užklausa. Galimi du užklausos vykdymo variantai:

- Užklausa gali būti vykdoma kiekvieną kartą atliekant paiešką tarp duomenų bazėje (reliacinėje ar XML dokumentuose) saugomų duomenų.
- Atlikus užklausą, išsaugoma jos rezultatų kopija. Vykdant užklausą antrą kartą, gali būti nurodoma, ar paieška atliekama paskutinėje rezultatų versijoje, ar vėl kreipiamasi į duomenų bazėje saugomas pagrindinių duomenų struktūras (t. y. ne į ankstesnius rezultatus).

Abstrakčiai duomenų struktūrai, iš kurios išrenkami duomenys, užklausos vykdymo metu skiriama klasė „PaieškosVykdymoObjektas“, apibendrinanti dvi klases – „UžklausosRezultatas“ bei „DuomenųBazė“, skirtas atitinkamai užklausų rezultatų versijoms bei duomenų bazėje saugomų pagrindinių duomenų struktūroms pažymėti. Galutinė paieškos posistemės modelio diagrama pateikta 9-ajame paveiksle. Taip pat reikia atkreipti dėmesį, kad saugomos užklausos rezultatų versijos. Vartotojui turi būti galimybė vykdyti paiešką tarp ankstesnių užklausų rezultatų. Kad rezultatų identifikavimas turėtų vartotojui prasmę, prie jų papildomai gali būti pateikiama suformuota užklausa vartotojui suprantama išraiška. Papildomai „UžklausosRezultatas“ turi tinkamumo (angl. *relevance*) atributą – kiekvieniems užklausos rezultatams galima taikyti požymį, parodantį, kiek rasta informacija atitinka paieškos kriterijus.

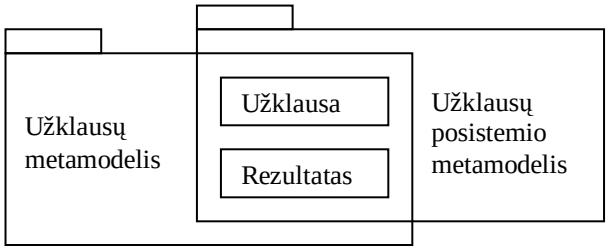


9 pav. Užklausų posistemės metamodelis

Taigi išskirti du susiję modeliai: užklausų metamodelis bei užklausų posistemės metamodelis. Pirmasis skirtas struktūrinei individualios užklausos informacijai saugoti: jį

sudaro užklausų sandarą atitinkantys elementai. Nestruktūriniams užklausų aspektams saugoti skirtas užklausų posistemio metamodelis.

9-ajame paveiksle pateikta diagrama nėra atskira nuo užklausų metamodelio. Kaip jau buvo minėta, „Užklausa“ klasė yra bendra abiems 8-ajame ir 9-ajame paveiksluose pateiktiems modeliams. Taip pat užklausos rezultatas (klasė „UžklausosRezultatas“) sutampa su užklausų metamodelio rezultatu (klase „Rezultatas“). Kaip šie modeliai atrodo bendrai, pateikta 10-ajame paveiksle.

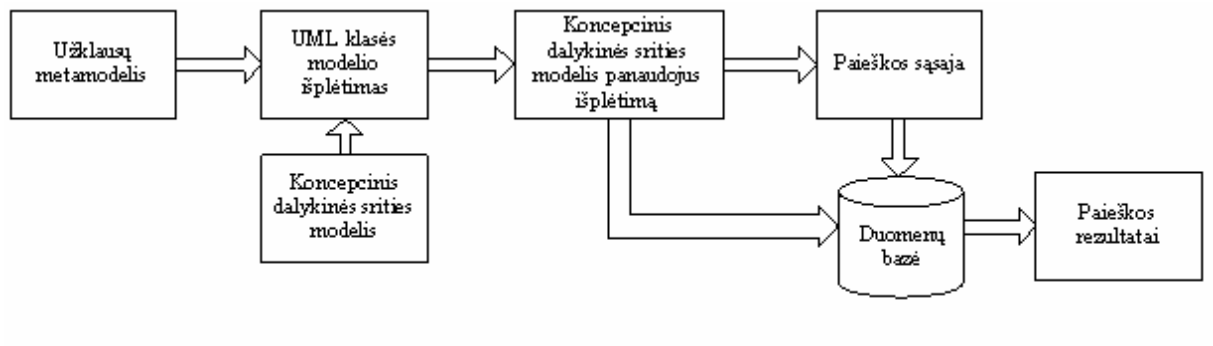


10 pav. Sąsaja tarp pagrindinio užklausų metamodelio bei užklausų posistemės metamodelio

Sudaryto metamodelio elementai naudojami modeliuojant paieškos sistemą. Tai yra, kiekvienas metamodelio elementas saugo savyje informaciją apie paieškos sistemos savybes. Pavyzdžiui, metamodelio elementas „RikiavimoKriterijus“ paieškos sistemos elementams suteikia papildomą prasmę – yra paieškos sistemos elemento, pagal kurį rezultatai gali būti rikiuojami, tipas. Šis taikymas įgyvendinamas naudojant UML išplėtimo mechanizmus, kurie detaliau apibūdinti 3.3 skyriuje – pastarajame būtent aptariamas išplėtimo mechanizmų naudojimas paieškos modelio atveju. Koks išplėtimą žymintis elementas bus naudojamas, priklauso nuo elemento ryšio su kitais metamodelio elementais bei jo įgyjamų reikšmių aibės.

3.3. Užklausų modelio naudojimas paieškai specifinėje srityje suformuoti

Sudaryto užklausų modelio pagrindu kuriamas UML išplėtimas, kurį būtų galima pritaikyti užklausai dalykinėje srityje formuoti turint koncepcinį šios srities modelį. Dalykinei sričiai vaizduoti naudojama UML klasių diagrama. Norint iš šios diagramos sukurti dalykinei sričiai reikalingą paieškos sistemą, diagramoje turi būti saugojama papildoma informacija. Tuo tikslu UML klasių diagrama praplečiama papildomais elementais bei požymiais. Praplėtimus panaudojus koncepciniame modelyje, klasės diagramoje saugoma informacija panaudojama paieškos sąsajai generuoti. Praplėsto koncepcinio modelio informacija kartu su vartotojo nurodytais paieškos parametrais panaudojama užklausai suformuoti bei įvykdyti. Šios veiksmų eigos pradiniai, tarpiniai ir galutiniai rezultatai pateikti 11-ajame paveiksle. Detaliau veiksmai aptarti ir iliustruoti pavyzdžiais 3.3.1-3.5 skyriuose.



11 pav. Paieškos sąsajos gavimo, naudojant suformuotą užklausų modelį bei koncepcinį dalykinės srities modelį, eiga

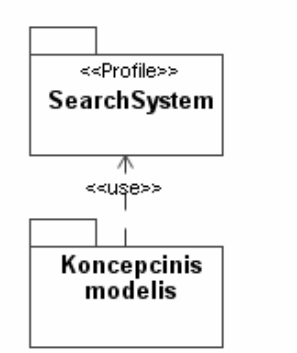
3.3.1. UML išplėtimo mechanizmai

UML gali būti išplėstas, kaip buvo minėta, pritaikant jį naudojimui konkrečioje srityje naudojant profilius. Profilyje sukuriami stereotipai, žymėtosios reikšmės (angl. *tagged values*) bei apribojimai – UML kalbos išplėtimo mechanizmai, kurie pritaikomi standartiniams UML diagramų elementams, atributams, metodams, ryšiams ar ryšių galams. Tokiu būdu standartiniams elementams suteikiama papildoma prasmė, nusakyta išplėtimo mechanizmais.

Kiekvienas stereotipas susietas su viena ar keliomis UML konstrukcijomis, kurios yra modifikuojamos. Stereotipas apibrėžia, su kokiu užklausų modelio elementu siejamas UML modelio elementas. Žymėtąsias reikšmes galima naudoti tokiais pačiais atvejais, kaip ir stereotipus, tačiau taip pat tikslinga jas taikyti užklausų modelio elementams, kurių nėra galimybės pažymėti koncepciniame modelyje tiesiogiai, stereotipais. Tuo tarpu apribojimai apibrėžia sąlygas, kurios turi būti patenkintos, norint stereotipą priskirti UML elementui [24].

3.3.2. Profilio kūrimas metamodelio pagrindu

Taigi sukurto užklausų metamodelio elementų prasmę suteikti koncepcinio modelio elementams naudojami UML išplėtimo elementai. Bendra šio proceso eiga apibrėžiama, kad visų pirma sukuriamas profilis ir, remiantis sudarytu modeliu, parenkami profilio elementai. Tuomet sudaryto koncepcinio modelio elementams nurodomi profilio elementai. Kad būtų aiškiau, 12-jame paveiksle parodyta, kad sukurtas profilis *SearchSystem*, o jis naudojamas koncepciniame modelyje.



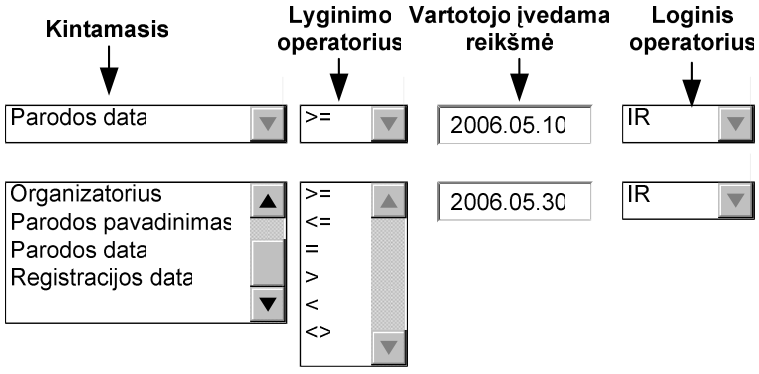
12 pav. Paieškos profilio SearchSystem pritaikymas koncepciniam modeliui

Taigi visų pirma turi būti sukuriamas paieškos sistemos profilis *SearchSystem*. Šis profilis skirtas visiems galimiems paieškos sistemos sričiai būdingiems elementams apibūdinti. Kuriamai paieškai atskiram paieškos dalykinėje srityje atvejui sukuriamas *SearchSubsystem* stereotipas. Šiam stereotipui sukuriami žemesnio lygio stereotipai, atitinkantys užklausų metamodelio elementus.

Šiems metamodelio elementams galima sukurti tokius stereotipus:

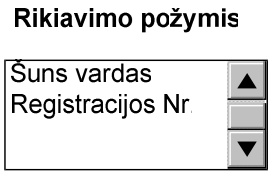
- <<SearchCriterion>> – paieškos kriterijus, klasė „Kintamasis“- taigi kintamojo elementas atvaizduotas <<SearchCriterion>> stereotipu,
- <<SearchObjectID>> – paieškos objekto identifikatorius, įvardija esybės identifikatorių,
- <<SearchResult>> - rezultatą sudarantys atributai (reliacinių duomenų bazių lentelių atributai ar XML dokumentų mazgai ar atributai),
- <<SortCriterion>> - atributas, pagal kurį rezultatai yra rikiuojami, atitinkantys klasę „RikiavimoKriterijus“.

Kad būtų galima aiškiau įsivaizduoti, kaip susiję išplėtimo elementai, dalykinė sritis ir paieškos sąsaja, 13 paveiksle pateikta numatomas galimas koncepcinio modelio elementų poaibio atvaizdavimo į sąsają pavyzdys. Visi elementai, kuriems priskirtas kintamojo stereotipas „SearchCriterion“, pateikiami pasirenkamų reikšmių sąraše.



13 pav. Sąsajos ir metamodelio elementų poaibio atitikmuo: kintamieji ir operatoriai

14 paveiksle pateikta stereotipu „SortCriterion“ pažymėtų koncepcinės schemos elementų vaizdavimo sąsajoje galimybė.



14 pav. Sąsajos ir metamodelio elementų poaibio atitikmuo: rikiavimo kriterijus

Kaip parodyta 13-ajame paveiksle, paieškos sąsajoje pateikiamos kelios eilutes, į kurias galima įvesti kriterijus. Praktikoje šį eilučių skaičių tikslinga sutapdinti su paieškos kintamųjų skaičiumi, kad vartotojas galėtų, esant poreikiui, atrinkti rezultatus pagal visus įmanomus kriterijus. Šiam kiekiui pažymėti sukuriama žymėtoji reikšmė „CriteriaCount“.

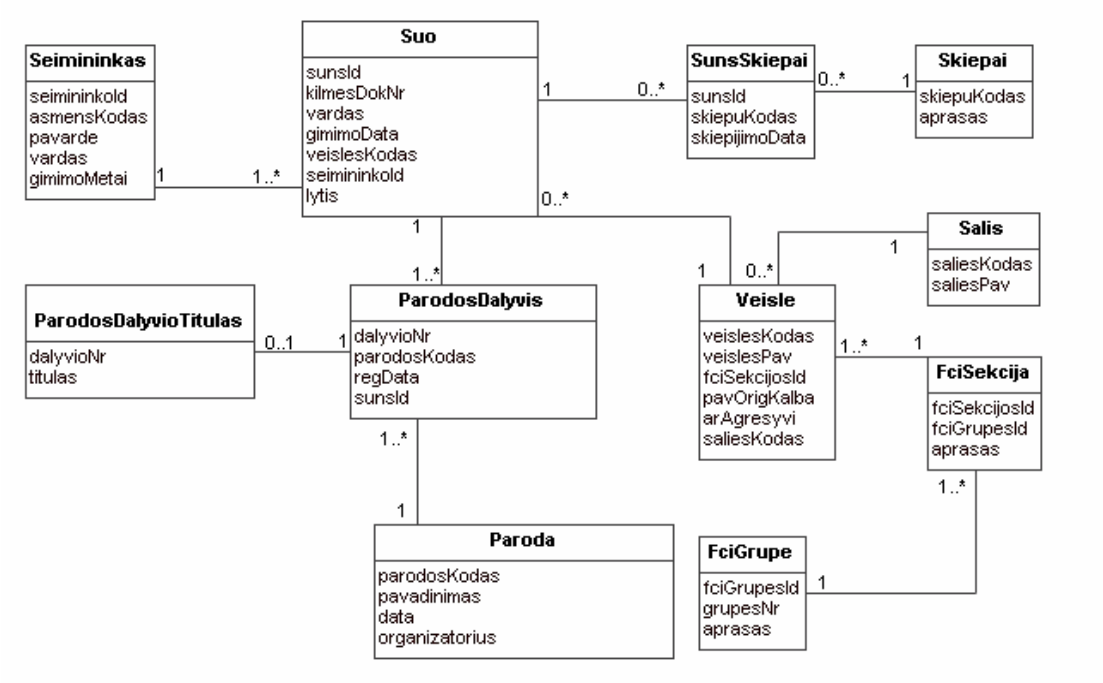
Visi paieškos profilio *SearchSystem* stereotipai ir žymėtosios reikšmės pateikiami 8 lentelėje.

8 lentelė. Paieškos sistemos profilis SearchSystem

Stereotipo pavadinimas	Bazinė klasė	Tėvinis elementas	Žymėtosios reikšmės	Aprašas
<<SearchSubsystem>>	Package	Nėra	CriteriaCount	Pažymi, kad paketo elementai priklauso paieškos posistemės sričiai
<<SearchCriterion>>	Attribute	<<SearchSubsystem>>	Nėra	Pažymi paieškos kriterijaus atributą
<<SortCriterion>>	Attribute	<<SearchSubsystem>>	Nėra	Pažymi rikiavimo kriterijaus atributą
<<SearchObjectID>>	Attribute	<<SearchSubsystem>>	Nėra	Pažymi paieškos objekto identifikatorių
<<SearchResult>>	Attribute	<<SearchSubsystem>>	Nėra	Pažymi paieškos rezultato atributą

3.4. Metodikos taikymas UML įrankiuose

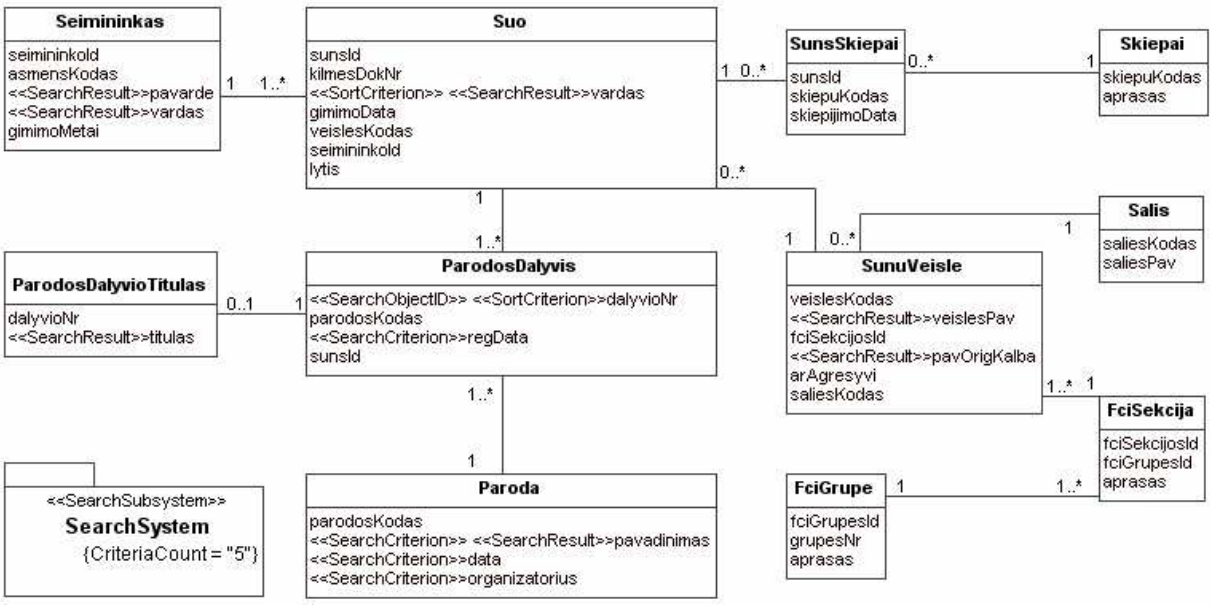
Norint taikyti aprašytą metodiką kuriant paieškos sistemą, visų pirma reikia sukurti dalykinės srities koncepcinę klasių diagramą, naudojant pasirinktą UML priemonę. Srities, kuriai taikoma metodika, koncepcinė klasių diagrama pateikta 15-ajame paveiksle.



15 pav. Koncepcinė dalykinės srities diagrama

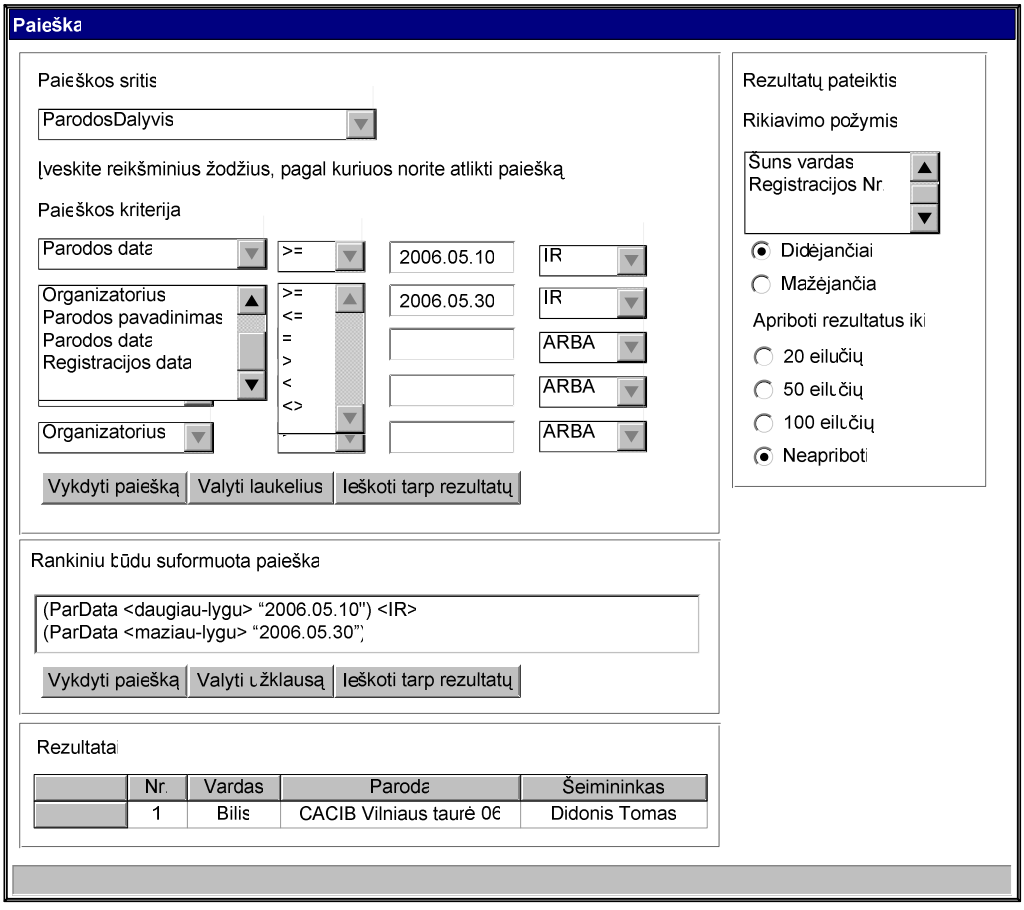
Kaip buvo minėta, papildoma reikšmė srities elementams suteikiama naudojant UML praplėtimo mechanizmus. Koncepcinė klasių diagrama, vaizduojanti dalykinės srities elementus ir kuriai yra kuriama paieškos sistema, sukuriama atskirame pakete, kuriam priskiriamas <<SearchSubsystem>> stereotipas. Toliau pateikta stereotipų ir žymėtųjų reikšmių priskyrimo eiga nebūtinai turi būti atliekama tokia tvarka, kokia veiksmai yra aprašyti.

Priskyrus <<SearchSubsystem>> stereotipą, pasirenkamas reikiamas srities paieškos objektas ir pažymimas atitinkamu stereotipu <<SearchObjectID>>. Taip pat atributai, pagal kuriuos numatoma galimybė atlikti rikiavima, pažymimi stereotipu <<SortCriterion>>, o visam paieškos, skirtos dalykinei sričiai, posistemiui nurodomas paieškos kriterijų skaičius, kurį galima pasirinkti vykdant paiešką, įvedant žymėtosios reikšmės „CriteriaCount“ reikšmę. Galiausiai nurodomi atributai, kurie naudojami kaip paieškos kriterijai, ir pažymimi stereotipu <<SearchCriterion>>. Pasirinktais paieškos elementais papildyta dalykinės srities diagrama pateikta 16-ajame paveiksle.



16 pav. Koncepcinė dalykinės srities diagrama su praplėtimais

Remiantis pažymėta informacija, koncepcinės diagramos elementai, pažymėti naudojant stereotipus ir žymėtąsias reikšmes, atvaizduojami į paieškos sąsają, kaip buvo parodyta 13-ajame paveiksle. 17-ajame paveiksle, kuriame pateikiama numatoma paieškos sąsaja, kurią norima sukurti, <<SearchCriterion>> pažymėti atributai, transformuoti į sąsajos paieškos kriterijų sąrašus, o <<SortCriterion>> - į rikiavimo požymių sąrašą.



17 pav. Paieškos dalykinėje srityje sąsaja

Statiniai ir dinaminiai sąsajos elementai

Dalis pateiktoje sąsajoje vaizduojamų elementų yra statiniai, t. y. jie neturi priklausyti nuo modeliuojamos srities. Kaip statinius elementus galima išskirti mygtukus „Vykdyti paiešką“, „Valyti laukelius“, „Ieškoti tarp rezultatų“, išrenkamų rezultatų skaičiaus apribojimo grupės sritis su apribojimo reikšmėmis bei rezultato panašumo reikšmę. Šiai rezultato panašumo reikšmei apskaičiuoti gali būti taikomos įvairios skaičiavimo schemos:

- gali būti skaičiuojama, kiek užklausoje nurodytų kriterijų atitinka išrinktas rezultatas, imant atitinkančių kriterijų rezultatų skaičiaus ir bendro skaičiaus santykį (gali būti išreiškiamas procentais);
- kita alternatyva – santykinio panašumo skaičiavimas. Jam įvertinti turi būti saugoma papildoma informacija apie sąvokų panašumą. Jeigu turime dvi sąvokas S1 ir S2, tarp kurių panašumas yra 70 proc., o yra ieškoma reikšminio žodžio, apimančio sąvoką S1, tuomet paieškos rezultatas, radus sąvoką S1, bus vertinamas kaip 100 proc. panašus į reikšminį žodį, t. y. pilnai jį atitinkąs. Jeigu bus rasta sąvoka S2, tuomet rezultato įvertis bus lygus 70 proc.

Loginiai operatoriai bei lyginamieji ir paieškos tikslumą apibrėžiantys operatoriai taip pat priskiriami prie statinio tipo elementų – juos galima būtų naudoti visose paieškos sąsajose.

3.5. Prototipo kūrimo priemonių bei metodų alternatyvos bei pasirinkimas

Realizacijos metodai

Paieškos sistemą galima realizuoti dviejais būdais:

- naudoti XMI failą. XMI faile saugomi metaduomenys apie sukurtą diagramą. Pats XMI standartas skirtas keitimuisi duomenimis, naudojant XML formatą, o XML formato universalumą galima išnaudoti įvairiomis priemonėmis. Taigi, analizuojant XML pateiktą informaciją, galima programiškai sudaryti atvaizdavimo į sąsają algoritmą;
- padaryti savo metamodelį ir aprašyti jį XML schema.

Metodų įgyvendinamumas naudojant MagicDraw

MagicDraw paketas suteikia galimybę sukurtus paketus ir profilius eksportuoti ir išsaugoti OMG siūlomu XMI formatu, kurį galima naudoti UML modelį transformuojant į paieškos sąsają. Taip pat naudojant šį paketą kuriami projektai saugojami archyvuotoje XMI faile. Papildomai MagicDraw paketas teikia galimybę susidaryti XML schemą bei ją generuoti, kas yra patogu siekiant gauti savo sudaromą modelį XML formatu. MagicDraw teikia tiesioginės ir atvirkštinės kodo inžinerijos naudojimo galimybes, taigi egzistuojančias

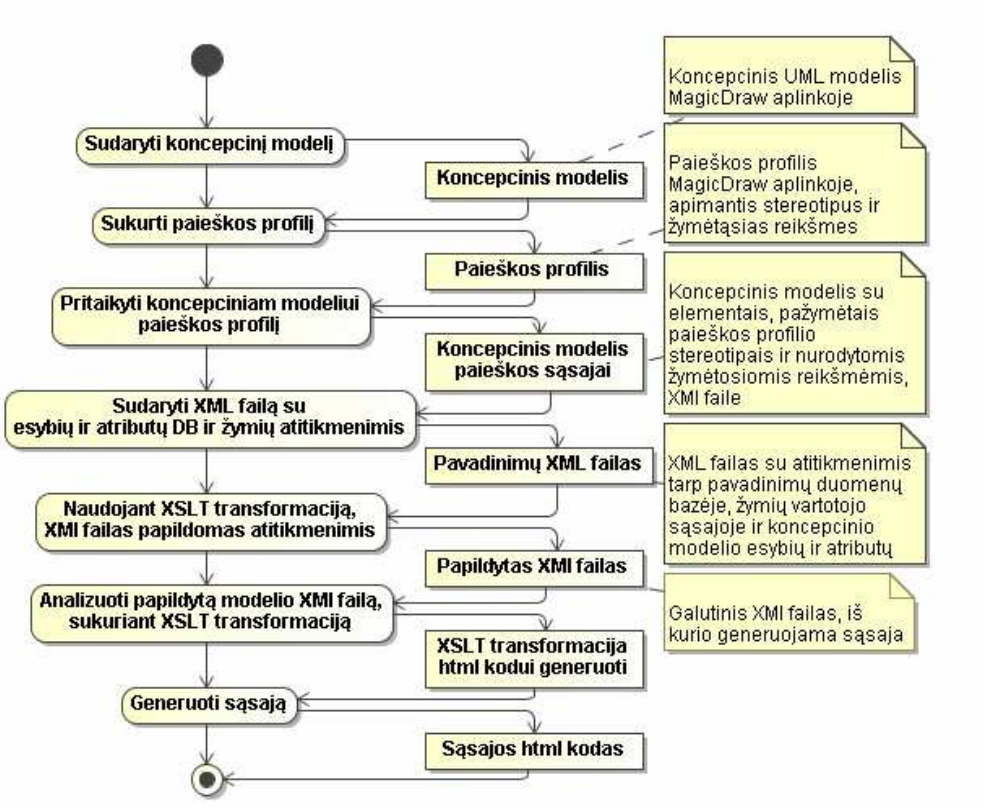
UML klasių diagramas galima transformuoti į XML schemą ir, atvirkščiai, turint schemą, galima iš jos generuoti schemos klasių diagramą.

Pasirenkama alternatyva ir pasirinkimo pagrįstumas

Realizacijai pasirinktas pirmasis variantas, kadangi šiuo atveju nereikia kurti papildomos schemos, be to, tokiu atveju modelis nepriklauso nuo konkretaus naudojamo įrankio – XMI failo formatą apibrėžia OMG standartas. Kadangi XMI failas – XML struktūros dokumentas, sąsajai generuoti tikslinga naudoti XSLT transformacijas – tai taip pat (žiniatinklio) standartas, sukurtas W3C grupės, skirtas XML dokumentus transformuoti į XML, HTML dokumentus ar paprastą tekstą.

Paieškos sąsajos kūrimo metodikos apibendrinimas

Atliktų veiksmų visumą galima apibendrinti kaip metodiką paieškos sąsajai kurti ir užklausoms formuoti naudojant pasiūlytą metamodelį.. Bendra sąsajos kūrimo eiga pateikta 18 pav.



18 pav. Sąsajos kūrimo eiga

Jos metu:

- sudaromas koncepcinis modelis sričiai, kurioje turi būti atliekama paieška;
- panaudojamas sukurtas paieškos profilis ir pritaikomas koncepcinio modelio elementams;

- kadangi koncepciniame modelyje, duomenų bazėje bei paieškos sąsajoje klasių ir atributų atitikmenų pavadinimai dažniausiai skiriasi, sukurtas XML failas, kuriame išvardytas visas pavadinimų trejetas;
- analizuojamas modelio XMI failas ir papildomas atributais ir klasėmis iš XML failo;
- gautas XMI failas analizuojamas naudojant XSLT stiliaus dokumentą;
- XSLT transformacijos rezultatas – sugeneruota HTML sąsaja.

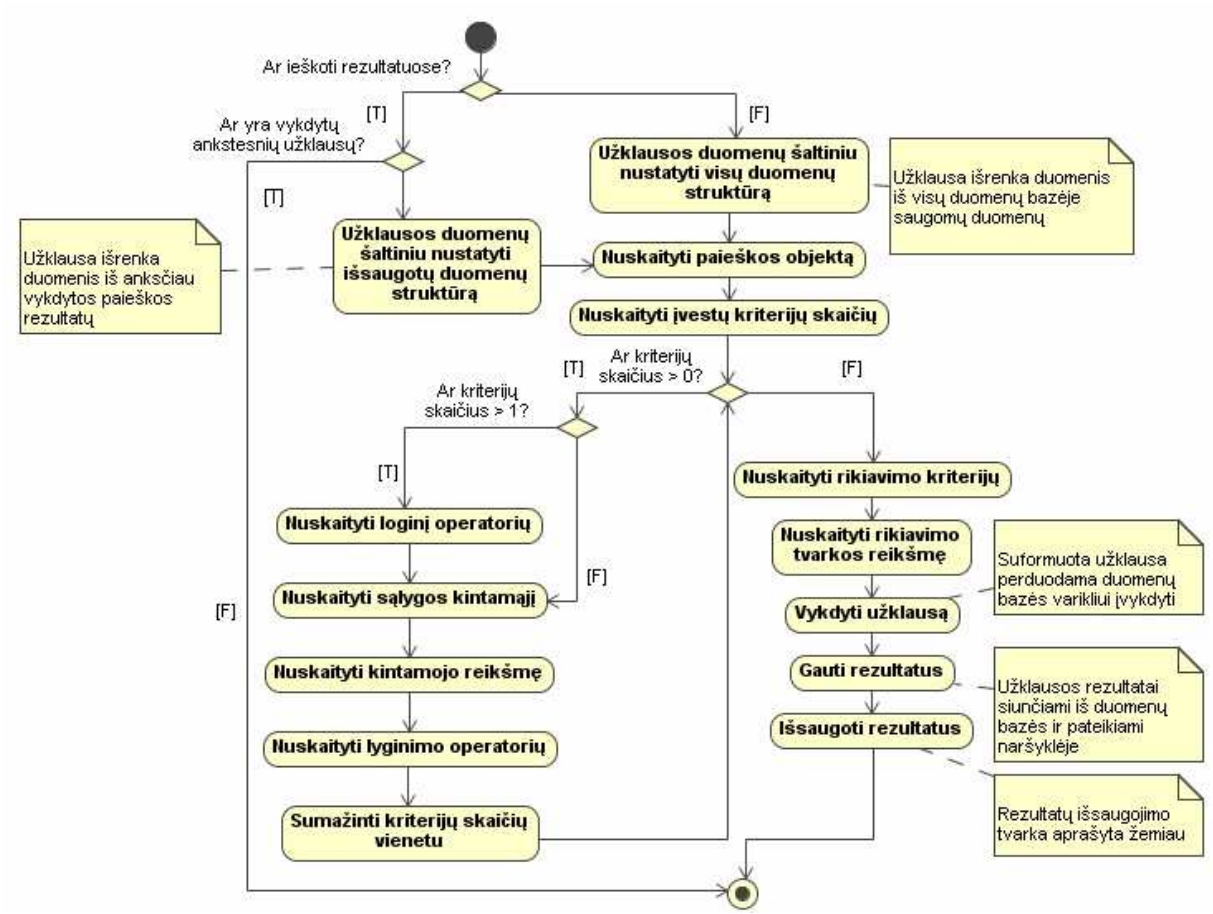
Naudojamos XSLT transformacijos pateikiamos 2-ajame priede.

Šiame algoritme pateikiama, kaip gaunamas paieškos sąsajos turinys, tačiau jis neapibrėžia nuo turinio nepriklausomų savybių, tokių, kaip sąsajos elementų išdėstymas ar jų formatas. Elementų išdėstymą galima nurodyti kuriant XSLT dokumentą, o jų stilius paprastai lanksčiausiai valdomas naudojant CSS (angl. *Cascading Style Sheet* – pakopinis stilius) dokumentą. Kadangi šio darbo tikslas yra apibrėžti užklausoms ir jų naudojimo paieškos sistemose būdingas savybes, stiliaus ar išdėstymo priemonių alternatyvos nenagrinėjamos.

Užklausų formavimo algoritmas

Pačioms užklausoms formuoti ir vykdyti sudaromas atskiras algoritmas. Atvaizdavimas į paieškos sąsają yra visiškai nepriklausomas nuo duomenų struktūrų. Tuo tarpu užklausų formavimo algoritmas turi panaudoti tiek praplėsto koncepcinio modelio informaciją, tiek informaciją apie duomenų struktūros elementus. Algoritmas apima dalį anksčiau pateiktų etapų, tai yra, norint, kad būtų suformuota užklausa, reikia sukurti paieškos sąsają; tuomet atliekami papildomi veiksmai (pateikiami ir pirmieji veiksmai, sutampantys su sąsajos kūrimo veiksmais):

- sudaromas koncepcinis modelis UML;
- panaudojamas sukurtas paieškos profilis, taip gaunant reikiamą modelio XMI failą;
- sukuriamas XML failas su atributų, klasių, duomenų bazių elementų pavadinimų atitikmenimis;
- naudojant XSLT transformaciją, modelio XMI failas papildomas XML atitikmenų informacija;
- naudojant XSLT transformaciją, generuojama vartotojo sąsaja;
- vartotojas įveda duomenis ir pasirenka paieškos vykdymo mygtuką;
- nuskaitomi vartotojo, šiam naudojantis sąsaja, įvesti užklausos parametrai;
- apdorojama užklausa – šio proceso diagrama pateikiama 19 pav..



19 pav. Paieškos užklausų sudarymo ir vykdymo algoritmas

Užklausos formavimo metu susiejamos duomenų struktūros elementų reikšmės su vartotojo nurodytais parametrais. Šis formavimas bei suformuotos užklausos perdavimas vykdymui atliekamas naudojant įvairias papildomas programavimo technologijas.

Paieškos tarp rezultatų ir ranka formuojamos paieškos galimybių detalus aprašymas pateikiamas atskirai, siekiant neperpildyti 19 paveiksle pateikto algoritmo.

Paieška tarp rezultatų

19 paveiksle pateiktame algoritme pirmuoju žingsniu patikrinama, ar paieška atliekama tarp rezultatų. Paieškos tarp rezultatų idėja SQL ir XML užklausų kalbų atveju sutampa: įvykdžius užklausą, išsaugomi rezultatai. Jeigu duomenų struktūra – reliacinė, rezultatams išsaugoti sukuriama papildoma lentelė, kurioje stulpelių skaičius atitinka rezultato eilutės stulpelių skaičių. Taip pat konkrečios paieškos rezultatų eilutėms priskiriamas identifikatorius (taigi ir papildomas stulpelis), skirtas rezultatų versijoms pažymėti.

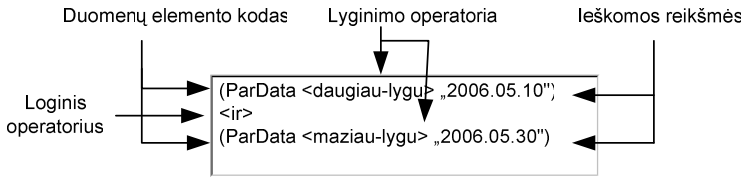
XML dokumentų atveju rezultatai įrašomi į XML failą. Šio failo struktūra atitinka pagrindinių XML dokumentų, iš kurių duomenys išrenkami pirmąją kartą vykdant užklausą, struktūrą. Kitaip tariant, tarp jo mazgų egzistuoja tokia pati hierarchija, kaip ir tarp pagrindinių duomenų XML faile. Papildomai sukuriamas atributas, žymintis rezultato versiją. Baigiant paieškos seansą, rezultatų duomenys naikinami.

Toks duomenų struktūrų formatas ir eiga nėra vieninteliai, kuriuos galima taikyti atliekant paiešką tarp rezultatų. XML rezultato failo hierarchinė struktūra priklauso nuo konkretaus kelio iki mazgų žymėjimo užklausos. Rezultatų duomenų naikinimas pasibaigus seansui – taip pat tik viena iš alternatyvų. Gali būti saugoma visada tik paskutinė rezultatų versija – tokiu atveju nereikalingas rezultatų versijų saugojimas. Galiausiai rezultatai apskritai gali būti nenaikinami (arba tai atliekama po tam tikro ilgo laiko tarpo) – tuomet papildomai be versijos būtų priskiriamas identifikatorius ir seansui.

Rankiniu būdu formuojama užklausa

Užklausai formuoti rankiniu būdu reikalingos taisyklės, apibrėžiančios, kaip užklausos dalis turi būti nurodoma paieškos autoriaus, kad ji būtų teisingai interpretuojama. Tai yra vartotojui turi būti pateikta aibė raktinių žodžių, kuriuos jis gali naudoti savo paieškos eilutėje. Šios taisyklės konkrečių paieškos sistemų atvejų gali skirtis, tačiau dažniausiai bendra paieškos eilutės formavimo struktūra panaši, parenkama tik skirtinga jų sintaksė.

Rankiniu būdu gali būti formuojama visa paieškos eilutė arba tik jos sąlygos dalis. Pastaroji situacija pavaizduota 20 paveiksle.



20 pav. Rankiniu būdu formuojamos paieškos elementai

Paveiksle taip pat pateikiama galima sintaksė. Taisyklės yra reikalingos šiems elementams:

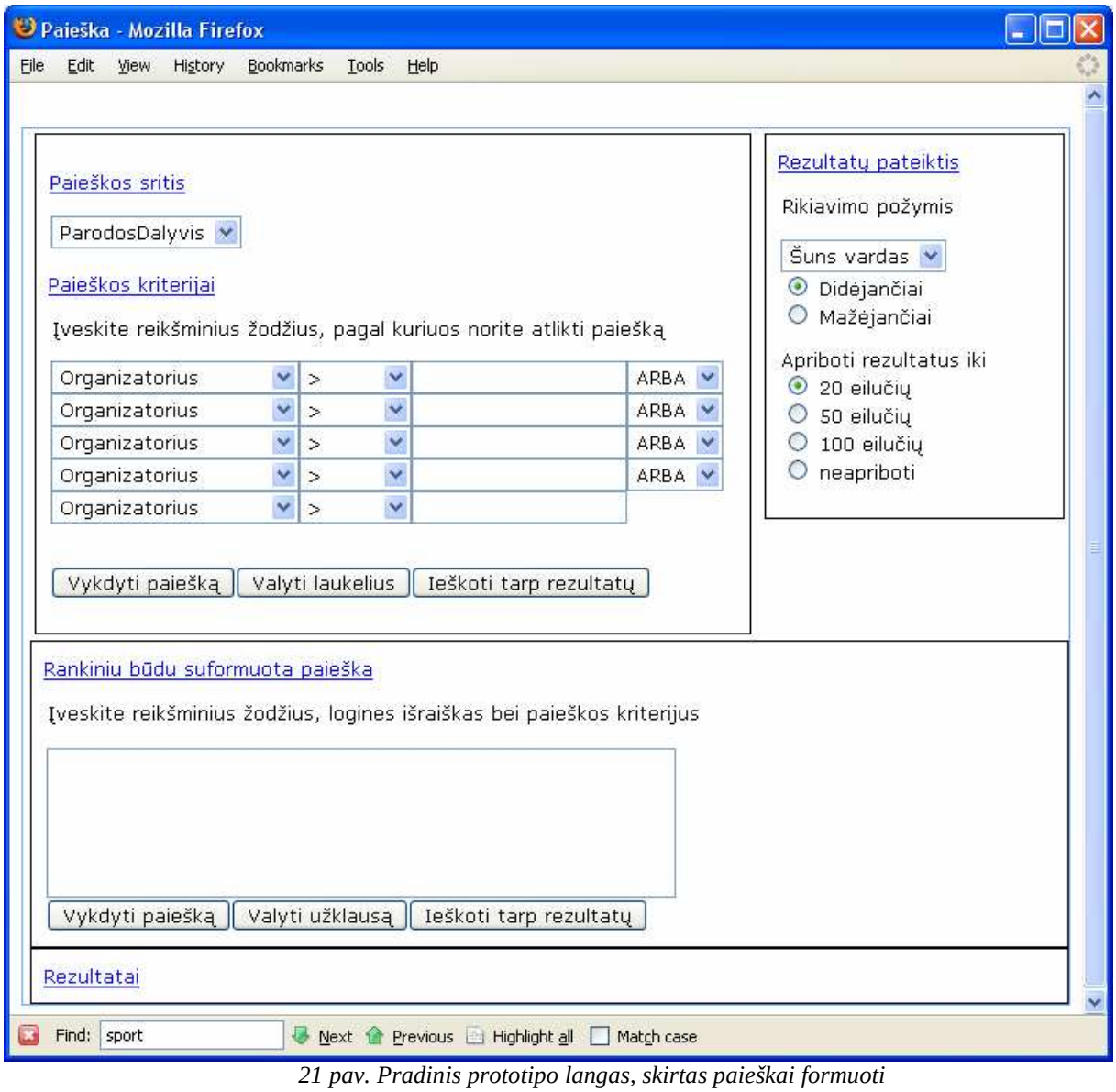
- duomenų elementams (SQL atveju – atributams, XML užklausų atveju – mazgams ar jų atributams) – vartotojas turi žinoti, kaip jam įvardyti duomenų elementą, pvz., naudojamą atrankos sąlygoje. Sudaromas sąrašas duomenų elementų kodų ir pateikiamas vartotojui. Šie kodai, perdavus užklausą apdoroti, atvaizduojami į realius duomenų elementų pavadinimus duomenų bazėje;
- operatoriams (lyginimo bei loginiams) – jeigu vartotojui pateikiami ne tiesioginiai duomenų bazėje naudojami operatoriai, kaip parodyta paveiksle „daugiau-lygu“ ir „maziau-lygu“ pavyzdžiu, tuomet reikalingos šių operatorių atvaizdavimo į tikrąjį duomenų bazės operatorių taisyklės;
- atrankos sąlygos grupėms išskirti.

4. PAIEŠKOS SISTEMOS PROTOTIPAS

Remiantis modeliu ir aprašyta metodika, sukurtas paieškos sistemos prototipas, leidžiantis formuoti paiešką pagal įvairius kriterijus įvairiose sistemos objektų aibėse. Tolesniuose skyriuose pateiktas prototipo aprašymas bei pateikti ekraniniai jo vaizdai. Galiausiai, siekiant įvertinti jo kokybę, aptariamos prototipo savybės. Prototipo duomenų bazė yra reliacinė, ir jos esybių-ryšių diagrama pateikta 3-ajame priede.

4.1. Trumpas prototipo aprašymas ir ekraniniai vaizdai

Sąsajos elementai, skirti veiksmams, kuriuos vartotojas turi galimybę atlikti formuodamas paiešką, vykdyti pateikiami viename naršyklės lange (21 pav.). Čia išskiriamos atskiros sritys, numatytos susijusiai operacijų grupei atlikti ar požymiams pasirinkti.

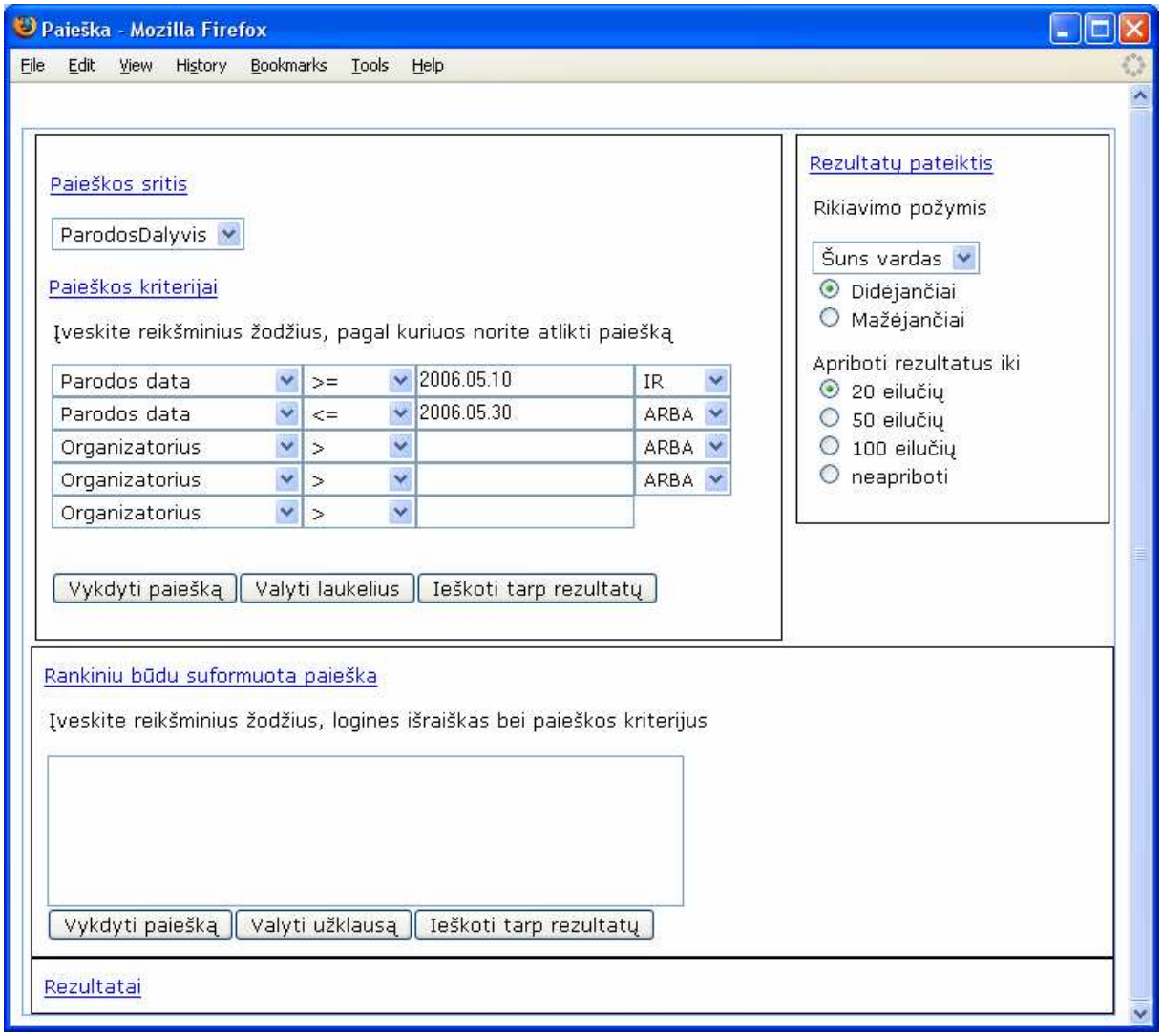


21 pav. Pradinis prototipo langas, skirtas paieškai formuoti

Prototipo sąsaja yra sugeneruojama taikant pateiktą metodiką. Joje pateikti visi elementai, kurie koncepcinėje srityje pažymėti stereotipais ar žymėtosiomis reikšmėmis: paieškos, rikiavimo kriterijai, paieškos objektai bei kriterijų eilučių skaičius.

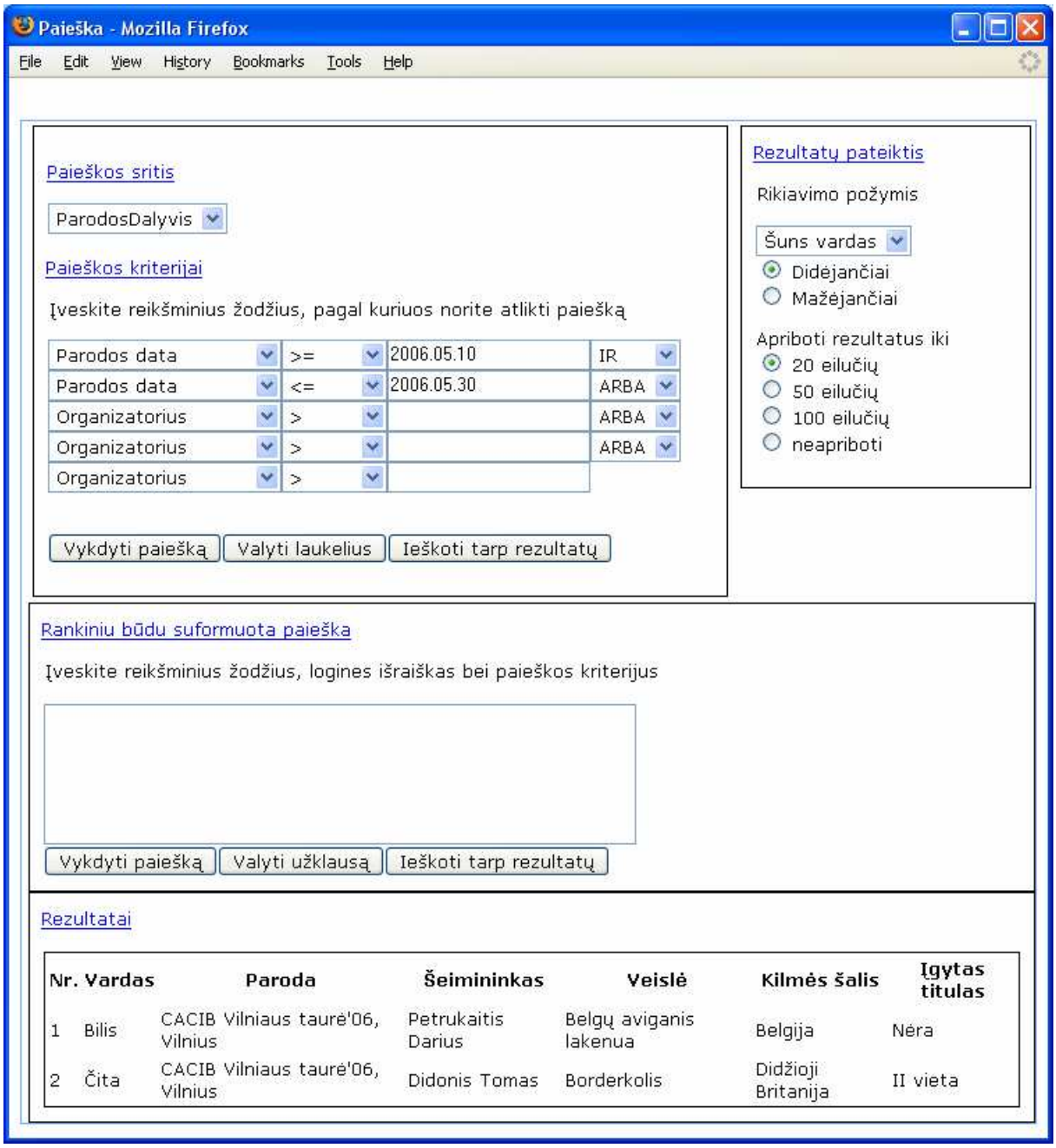
Norint atlikti paiešką naudojantis sąsajos elementais – pasirenkant paieškos sritį, paieškos kriterijus, operatorius bei nurodant rikiavimo kriterijaus ir rikiavimo tvarkos reikšmę, paspaudžiamas atitinkamoje „Paieškos kriterijai“ srityje esantis mygtukas „Vykdyti paiešką“. Rankiniu būdu formuojama paieška atliekama įvedant paieškos sąlygos tekstą į „Rankiniu būdu suformuota paieška“ sritį ir vėlgi paspaudžiant mygtuką „Vykdyti paiešką“, esantį po tekstu (22 pav.).

Papildomi paieškos požymiai – atributo, pagal kurį atliekamas rikiavimas, rikiavimo tvarka bei rezultatų apribojimo požymis nustatomi „Rezultatų pateiktis“ srityje.



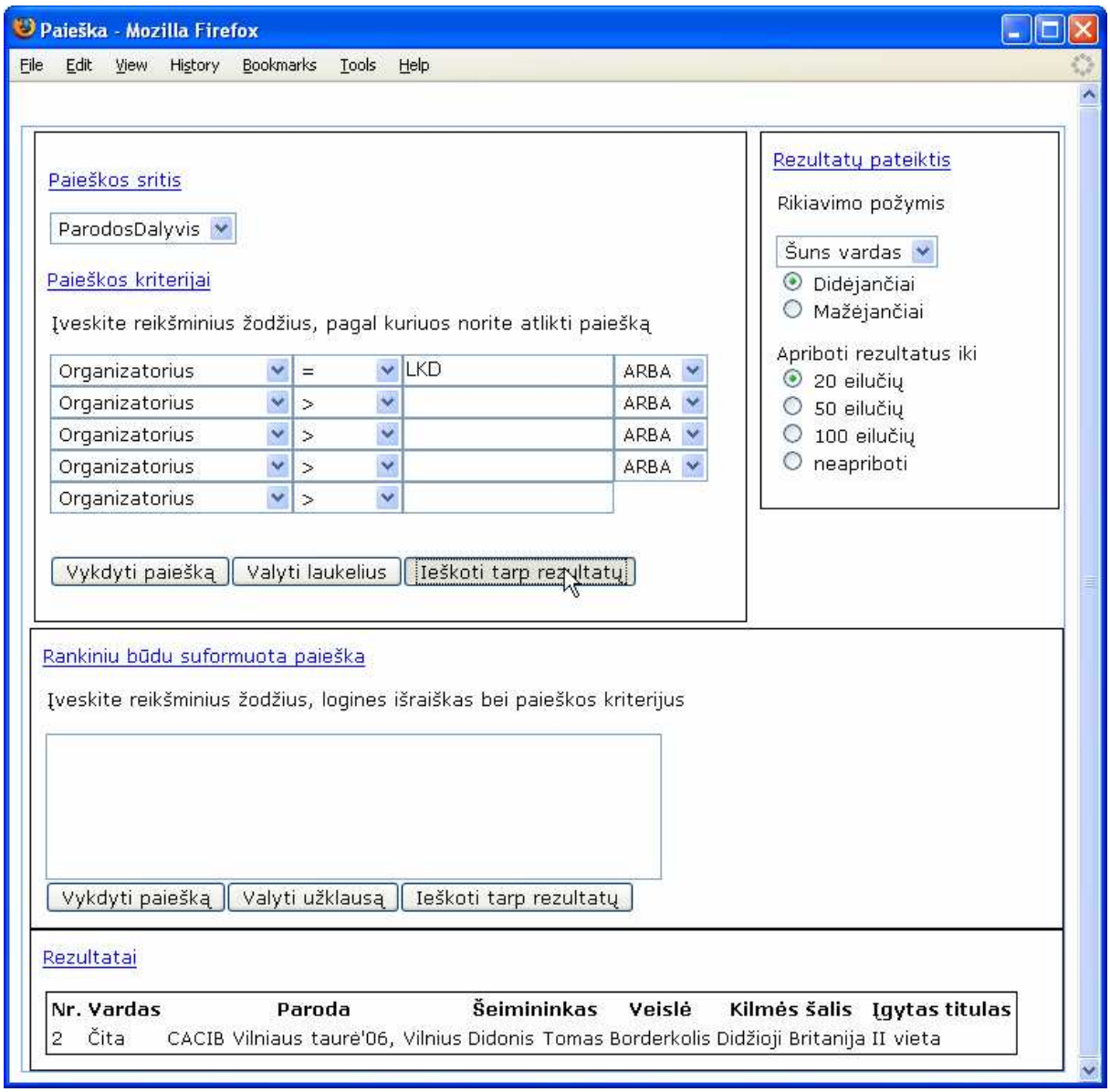
22 pav. Paieškos formavimas

Galiausiai, paspaudus mygtuką „Vykdyti paiešką“, atliekama paieška, ir „Rezultatai“ skiltyje pateikiami išrinkti duomenys (23 pav.)



23 pav. Rezultatai, gauti įvykdžius užklausą

Paspaudus mygtuką „Ieškoti tarp rezultatų“ ir nurodžius paieškos kriterijus, pateikiami patikslinti anksčiau vykdytos užklausos rezultatai (24 pav.).



24 pav. Paieškos rezultatai, gauti vykdant paiešką tarp rezultatų

4.2. Metodikos naudojimas kuriant prototipą ir prototipo vertinimas

Sukurta paieškos sistema apima didžiąją daugumą savybių, kurias teikia kitos paieškos sąsajos, pavyzdžiai kurių buvo pateikti 7 lentelėje. Dalis jų buvo realizuotos pagal pateiktą metodiką. Kita dalis, kurios modelis neišreiškia, realizuota alternatyviais būdais, siekiant, kad paieška apimtų visumą galimų savybių. 9 lentelėje pateiktas paieškos sistemų savybių sąrašas bei būdai, naudoti arba pasiūlyti joms įgyvendinti.

9 lentelė. Naudoti paieškos sistemos savybių realizavimo būdai

Savybė	Realizavimo būdas
Pasirenkamas paieškos kriterijų sąrašas	Remiantis metodika, paieškos kriterijais pažymėti elementai pateikiami sąlygos kriterijų sąraše
Paieškos laukai jungiami pasirenkamais loginiais operatoriais	Pasirenkami loginiai operatoriai, nurodyti modelyje klase „LoginisOperatorius“
Tekstiniame lauke formuojama paieška	Pateiktas metodas paieškai formuoti
Pasirenkamas rikiavimo kriterijus ir rikiavimo tvarka	Remiantis metodika, rikiavimo kriterijais pažymėti elementai pateikiami rikiavimo kriterijų sąraše; rikiavimo tvarka, kaip nurodyta modelyje, visada gali įgyti dvi reikšmes, todėl pateikiama kaip statinis elementas; rikiavimo tvarka, formuojant užklausą, nuskaitoma pagal pateiktą algoritmą
Paieška tarp rezultatų	Remiantis metodika
Rezultatų kiekio apribojimas	Pateikiamas kaip statinis sistemos elementas; metodika jo neapibrėžia
Nurodoma paieškos sritis	Remiantis metodika, paieškos objektais pažymėti elementai pateikiami paieškos srities sąraše
Lyginimo operatoriai	Pasirenkami lyginimo operatoriai, nurodyti modelyje klase „LyginimoOperatorius“
Įvertinamas rezultato tinkamumas	Pateikiamas metodas rezultatui įvertinti

Prototipo sąsaja yra tipiška paieškos sistemoms, todėl tiek pradedantysis vartotojas intuityviai, be papildomų mokymų gali formuoti užklausas naudodamasis pagrindinėmis paieškos savybėmis, tiek daugiau patyręs vartotojas gali formuoti užklausas rankiniu būdu.

Pasikeitus dalykinei sričiai, pavyzdžiui, ją papildžius naujais elementais, pakeitimai egzistuojančioje sistemoje būtų atliekami pažymėjus naujus elementais UML stereotipais, o programiniu būdu pakeitimų atlikti nebereikėtų. Pakaktų įvykdyti aprašytas XSLT transformacijas, kas pagreitina programinės įrangos atnaujinimo procesą.

Prototipo savybes galima plėsti keliuose lygiuose:

- vizualias jo savybes, tokias, kaip spalvas, rėmelius, šriftą galima keisti keičiant stiliaus dokumentą;
- sąsajos elementų išdėstymą bei turinį galima keisti keičiant transformacijos dokumente aprašytas taisykles;
- duomenų struktūrų elementų prasmę galima keisti keičiant arba papildant paieškos UML profilį.

Taigi egzistuoja įvairios prototipo plėtimo galimybės, tipiška jo sąsaja patogi vartotojams įsisavinant jos teikiamas savybes, o teikiamos funkcinės galimybės atitinka didžiąją dalį paieškos sistemų teikiamų galimybių. Remiantis tuo, galima teigti, kad prototipas yra kokybiškas funkcionalumo požiūriu ir sudarytą modelį tinka naudoti įgyvendinant dažniausiai paieškos sistemose naudojamas funkcines galimybes.

5. IŠVADOS IR REKOMENDACIJOS

5.1. Išvados

1. Darbo metu atlikta plačiausiai paplitusiems duomenų saugojimo formatams skirtų užklausų kalbų analizė parodė, kad šios kalbos turi daug bendrų savybių, todėl galima sudaryti jas apibendrinantį modelį.
2. Atliekant analizę, susiformavo idėja, kad užklausų kalbos ir paieškos sistemos veikia panašiai, todėl galima sudaryti modelį, kuris leistų tokių sistemų kūrimą automatizuoti.
3. Paieškos sistemos modelį tikslinga išvesti iš dalykinės srities, kurioje bus atliekama paieška, koncepcinio modelio, papildant jį paieškos sistemos stereotipais. Kad paieška būtų efektyvesnė, paieškos sistemą reikia ne tik kurti pagal užklausų struktūrą, bet ir papildyti ją nuoseklaus vykdymo elementais.
4. Nustatyta, kad modelyje tikslinga išskirti dvi sritis – viena jų susijusi su struktūrinėmis užklausų savybėmis, kita su jų vykdymu, kadangi struktūrinės savybės tiesiogiai susijusios su paieškų sistemų ir pačių užklausų formavimu, o paieškos galimybėms padidinti yra svarbios su užklausų vykdymu susijusios savybės.
5. Modeliavimui pasirinkta UML kalba, kuri leidžia su paieška susijusios sistemos dalies projektavimą nuosekliai įjungti į visos sistemos projektavimo procesą. Be to, UML modelis gali būti saugomas standartiniu XMI formatu, kuris leidžia automatizuoti paieškos sistemų kūrimą ir suteikia nepriklausomumą nuo konkrečių UML CASE įrankių.
6. Sudarytas užklausų metamodelis ir jo taikymo metodas nepriklauso nuo konkrečios užklausų kalbos ir dalykinės srities, todėl gali būti taikomas praktikoje automatizuojant ir pagerinant paieškos sistemų kūrimą įvairioms dalykinėms sritims.
7. Realizuotas prototipas patvirtino metodo veiksmingumą, o lyginamoji analizė su kitomis paieškos sistemomis parodė, kad sudarytas modelis padidina semantines paieškos galimybes, kadangi vartotojas gali panaudoti rezultatus tolesniam užklausų patikslinimui.

5.2. Rekomendacijos

Aptarta ir pademonstruota tik viena modelio taikymo galimybė. Papildomi tyrimai galėtų pritaikyti analizės metu surinktą informaciją praplečiant modelį kitoms taikymo

sritims. Be to, nagrinėjamu metu XQuery kalba dar tik įgijusi rekomendacijos statusą, ir ją numatyta plėsti įgyvendinant daugiau operacijų, kurias ji galėtų atlikti [6], taigi modelis, apimdamas daugiau bendrų savybių, taptų universalesnis. Praktikoje naudojamos užklausų formavimo priemonės, suteikiančios galimybę užklausas sudaryti naudojantis specialiai tam skirta sąsaja, o ne jas rašant tiesiogiai. Tai patogu tiek didelės patirties užklausų rašymo srityje turintiems vartotojams, tiek tokios priemonės labai patogios, jeigu jas reikia atnaujinti arba panaudoti kaip šabloną kitų užklausų formavimui ir patyrusiems programuotojams.

Šiuo metu, norint, kad modelio informaciją galima būtų taikyti ir betarpiškam užklausų ir paieškos sąsajos generavimui, koncepcinio modelio klasių ir atributų pavadinimai turi sutapti su duomenų struktūrų elementų pavadinimais bei paieškos sąsajoje naudojamais pavadinimais. Modelyje galėtų būtų naudojami sričiai tinkami pavadinimai, tačiau jis taip pat galėtų saugoti ir atvaizdavimui į kitas duomenų struktūras reikalingą informaciją bei sąsajos pavadinimus.

SUMMARY

Query Metamodel for Search System Development

SQL, XPath, and XQuery are the query languages used to extract data from the most widely-used relational and XML data sources. As both of them are intended for different purposes, both of them have their niche in the world of database technologies.

In this work the analysis of the SQL, XPath, and XQuery query languages has been carried out, as the research of the available information has proved that no comprehensive comparative analysis of the languages exists. Also, a generic model has been developed that encompasses the properties inherent to all of the query languages based on the results of the conducted languages' analysis. The model is developed with the intended purpose in mind to facilitate the automation of the search system development process. Being a standard language for system specification and providing visual elements, UML has been used as an implementation medium which also helps involve search interface specification into the overall design process.

The feasibility and practical usage of the model has been tested by developing a conceptual model of a specific application domain. Using extension mechanisms UML provides, the conceptual model data has been mapped by applying transformations to the search system elements, and query model elements have been used in construction process of the queries within the domain, thus proving the model being suitable for its predefined purpose.

NAUDOTA LITERATŪRA

[1] ANSI/ISO/IEC International Standard (IS) Database Language SQL – Part 1: SQL/Framework, September 1999

[2] ANSI/ISO/IEC International Standard (IS) Database Language SQL – Part 2: Foundation (SQL/Foundation), September 1999

[3] **Beaulieu A.,** Learning SQL. O'Reilly & Associations, 2005, p. 52-74

[4] **Bonifati A., Ceri S.**, Comparative Analysis of Five XML Query Languages. ACM SIGMOD Record, 2000, Vol. 29, No 1. p. 68-79

[5] **Bourret R.** XML and Databases. [interaktyvus] 2005, September [žiūrėta 2006-05-10]. Prieiga per internetą: http://www.rpbourret.com/xml/XMLAndDatabases.htm

[6] **Conrad R., Scheffner D., Freytag J. Ch.** XML Conceptual Modeling Using UML: Conceptual Modeling – ER 2000: 19th International Conference on Conceptual Modeling, Proceedings. Vol. 1920/2000, p. 291-307.

[7] **Chamberlin, D.** XQuery: An XML query language. IBM Systems Journal, 2002, Vol. 41, No. 4, p. 597-615

[8] **Codd E. F.** A Relational Model of Data for Large Shared Data Banks: Communications of the ACM, Vol. 13, No. 6, June 1970, p. 377-387, Association of Computing Machinery

[9] **Eisenberg A., Kulkarni K., Melton J., Michels J.-E., Zemke F.** SQL:2003 Has Been Published. SIGMOD Record, 2004, Vol. 33, No. 1., p. 119-126

[10] **Grinev M. N.** UQL – A Query Language in Terms of UML, Translation from UQL into XQuery and XQuery Logical Optimization. Ph. D. Thesis at Computational Mathematics and Cybernetics Department in Moscow State University, 2003.

[11] **Howard Katz, Don Chamberlin, Denise Draper and others:** XQuery from the Experts: A Guide to the W3C XML Query Language, 2003, Addison Wesley

[12] **Kay M.**, XPath 2.0: Programmer‘s Reference. Wiley Publishing, 2004, p. 5-19

[13] **Kline K., Kline D.** SQL in a Nutshell: O‘Reilly & Associates, Inc., 2001, p. 3-25

[14] **Melton J.**, SQL, XQuery and SPARQL: What‘s Wrong With This Picture? [interaktyvus] XTech, 2006 [žiūrėta 2006-05-01]. Prieiga per internetą: http://xtech06.usefulinc.com/schedule/paper/119

[15] **North B.** XQuery and SQL: Vive la Difference. [interaktyvus] DB2 Magazine, 2003, Vol. 8, Issue 3. [žiūrėta 2006-03-31]. Prieiga per internetą: http://www.db2mag.com/story/showArticle.jhtml?articleID=12803228

[16] **Ozcan F., Chamberlin D., Kulkarni K., Michels J.-E.** Integration of SQL and XQuery. IBM Systems Journal, 2006, Vol. 45, No. 2, p. 245-270

[17] **Pal S., Cseri I., Seeliger O., Rys M., Schaller G., Yu W., Tomic D., Baras A., Berg B., Churin D., Kogan E**. XQuery Implementation in a Relational Database System: Proceedings of the 31st international conference on Very large data bases. VLDB Endowment, Trondheim, 2005, p. 1175-1186.

[18] **Riordan R. M.** Designing Relational Database Systems 1999, Microsoft Press, p. 53-77

[19] **Spiegel J.** A Survey of XML Query Languages [interaktyvus]. Department of Computer Science University of California, 2004 [žiūrėta 2006-05-11]. Prieiga per internetą: http://citeseer.ist.psu.edu/731008.html

[20] Unified Modeling Language [interaktyvus], Wikipedia [žiūrėta 2006-06-17]. Prieiga per internetą: http://en.wikipedia.org/wiki/Unified_Modeling_Language

[21] XML Information Set (Second Edition) [interaktyvus], W3C, 2004 [žiūrėta 2007-05-03]. Prieiga per internetą: http://www.w3.org/TR/xml-infoset/#infoitem.rse

[22] XML Path Language [interaktyvus], W3C, 1999 [žiūrėta 2006-02-07]. Prieiga per internetą: http://www.w3.org/TR/XPath

[23] XML Query (XQuery) Requirements [interaktyvus], W3C, 2005 [žiūrėta 2006-06-02]. Prieiga per internetą: http://www.w3.org/TR/xquery-requirements

[24] XQuery 1.0: An XML Query Language [interaktyvus], W3C, 2006 [žiūrėta 2006-02-24]. Prieiga per internetą: http://www.w3.org/TR/xquery/

[25] **Žederštreimaitė Ilma.** XML taikymo metodika kuriant duomenų apdorojimo sistemas internete, 2004 Kauno Technologijos Universitetas. Informacinės technologijos'2004 iš konferencijų ciklo „Lietuvos mokslas ir pramonė“.

SANTRUMPŲ IR TERMINŲ ŽODYNAS

ANSI (American National Standards Institute) – pelno nesiekianti organizacija, kuri vykdo (prižiūri) produktų, paslaugų, procesų ir sistemų JAV standartų kūrimą.

CASE (Computer-aided Software Engineering) – programinės įrangos priemonių naudojimas kuriant ir tvarkant sukurtą programinę įrangą

DTD (Data Type Definition) – duomenų tipo apibrėžtis; schema, apibrėžianti XML dokumento sintaksę ir semantiką

FCI (Federation Cynologique Internationale) – Tarptautinė kinologų federacija.

IS (Information System) – informacijos sistema.

ISO (International Organization for Standardization) – tarptautinis standartų nustatymo organas, kurį sudaro šalių standartų organų atstovai.

OMG (Object Management Group) – standartus objektiškai orientuotoms sistemoms kuriantis organas.

RDBS (Relational Database System) – reliacinės duomenų bazių sistema.

RDBVS – reliacinė duomenų bazių valdymo sistema

RDM (Relational Data Model) – reliacinis duomenų modelis.

UML (Unified Modelling Language) – vizuali kalba, apibrėžianti grafinę notaciją, skirtą įvairiems programinės įrangos architektūros aspektams modeliuoti.

W3C (World Wide Web Consortium) – WWW konsorciumas, skirtas pasaulinio interneto tinklo tobulinimui ir protokolų standartizacijai.

XMI (XML Metadata Interchange) – OMG standartas metaduomenims, naudojant XML formatą, keistis.

XML (eXtensible Markup Language) – išplėstinė žymėjimo kalba.

XSL (eXtensible Stylesheet Language) – XML dokumentų atvaizdavimas į skirtingas terpes skirtingais formatais

XSLT (eXtensible Stylesheet Language Transformation) – kalba, suteikianti galimybę transformuoti XML dokumentus iš vienos struktūros į kitą.

PRIEDAI

1 priedas. Straipsnis „Užklausų metamodelis automatizuotam paieškos sistemų kūrimui“

UŽKLAUSŲ METAMODELIS AUTOMATIZUOTAM PAIEŠKOS SISTEMŲ KŪRIMUI

Arjana Matonytė, Lina Nemuraitė

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

Straipsnyje nagrinėjama plačiausiai praktikoje paplitusiems duomenų šaltiniams – reliacinėms duomenų bazėms ir XML dokumentams – skirtų užklausų kalbų - SQL, XQuery ir XPath - užklausų struktūra. Remiantis analizės rezultatais, sudarytas bendrinis užklausų modelis, kurio paskirtis – padėti formuoti vartotojų užklausas nepriklausomai nuo to, kokiomis technologijomis realizuoti duomenų šaltiniai. Sukurto modelio praktinio taikomumo galimybėms pademonstruoti pateikta juo paremta užklausų sąsajos kūrimo metodika.

1. Įvadas

Duomenų išrinkimas, konvertavimas, transformavimas ir integravimas – gerai žinomos duomenų bazių problemos, kurių sprendimus įgyvendina užklausų kalbos. Jau kelis dešimtmečius dažniausiai yra naudojama reliacinė duomenų bazių struktūra, su kurios duomenimis operacijoms atlikti naudojama SQL užklausų kalba. Tuo tarpu XML standartas buvo sukurtas siekiant išspręsti egzistuojančią duomenų perdavimo tarp sistemų bei integravimo tarp nevienalyčių duomenų šaltinių problemas. XML kalba turi savitą struktūrą, todėl jai sukurtos užklausų kalbos XPath ir XQuery, tapusios žiniatinklio standartus kuriančios organizacijos W3C rekomendacijomis, efektyviausiai galinčios atlikti operacijas su šios struktūros duomenimis.

Kai reikia atlikti operacijas su skirtingos struktūros duomenimis, natūralu, kad pačių konkrečia kalba pateikiamų užklausų struktūra, savybės ir funkcijos tam tikra dalimi skiriasi. Perkelti užklausų sudarymą ir vaizdavimą į aukštesnį abstrakcijos lygmenį ir taip suteikti šiai sričiai aiškumo ir išvengti daugiaprasmiškumų galima sudarant bendrą kalbų modelį. Iki šiol nėra sukurto panašaus modelio, o ir pati išsami visų kalbų tarpusavio analizė, kurios informacija būtų naudinga šį modelį sudarant, nebuvo atlikta – egzistuoja tik pavieniai, nenuodugnūs bandymai sulyginti SQL ir XQuery kalbas [2-3], [5-7].

Siekiant užpildyti šią spragą, straipsnyje visų pirma atliekama nuodugni kalbų analizė, nepriklausoma nuo jos rezultatų panaudojimo srities. Modelis pasirinktas patogia forma susistemintiems analizės rezultatams pateikti bei sukurtas naudojant standartinę UML kalbą, kurios vizualus elementų pateikimas suteikia modeliui aiškumo. Be to, modelis skirtas padėti formuoti vartotojų užklausas nepriklausomai nuo to, kokiomis technologijomis realizuoti duomenų šaltiniai.

Taip pat, atliekant analizę, pastebėtas panašumas tarp užklausų kalbų ir paieškos sistemų veikimo, todėl pateikta praktinė modelio pritaikymo galimybė: pasiūlyta metodika, kaip sukurtas modelis gali būti panaudojamas automatizuojant paieškos sistemų kūrimą.

2. Užklausų kalbų analizė

Dauguma šiuolaikinių informacijos sistemų duomenis saugo reliacinėse duomenų bazėse (RDB). Standartinė RDB užklausų kalba – SQL, kurios standartas nors ir yra apibrėžtas ANSI ir ISO organizacijų, konkrečiose reliacinių duomenų bazių valdymo sistemose (RDBVS) SQL realizacija tam tikra dalimi skiriasi.

Duomenų bazių sistemoms apdorojant vis sudėtingesnių tipų informaciją ir iškylant nesuderinamumams tarp duomenų šaltinių, atsirado ir lankstaus duomenų formato poreikis – šiuo formatu tapo XML. Kadangi šis formatas yra savitos struktūros, jam reikalingos ir buvo sukurtos specialios užklausų kalbos, efektyviausiai atliekančios operacijas su XML formato duomenimis: XPath ir XQuery, abi tapusios žiniatinklio standartus kuriančios W3C organizacijos rekomendacijomis.

Visos užklausų kalbos tarpusavyje turi tam tikrų panašumų bei skirtumų. Iki šiol buvo atlikta įvairių pavienių užklausų kalbų analizė, taipogi ir pavieniai bandymai sulyginti SQL ir XQuery kalbas [3], [5-6]. Straipsnyje pateikiama visų trijų kalbų analizė tam tikra dalimi remiasi šių analizių metu išskirtais ir naudotais lyginimo kriterijais bei sukaupta informacija apie atskiras kalbas. Analizės rezultatai papildyti duomenimis iš kalbų standartų ir rekomendacijų dokumentų [1], [4], [8-9]. Lyginimo metu siekta atsižvelgti daugiausia į struktūrinius kiekviena kalba formuojamų užklausų panašumus ir skirtumus bei su ja susijusias savybes.

2.1. Užklausų struktūros ir savybių palyginimas

Konkrečios užklausų kalbos sąvokos sietinos su duomenų formatu, kuriam užklausų kalba yra skirta. Tiek RDB, tiek XML sąvokos skiriasi, taigi ir jų duomenų modeliai nevienalyčiai. Norint, kad užklausų kalbų analizės rezultatai būtų nedaugiaprasmiški, tikslinga apibrėžti pagrindines RDB ir XML duomenų modelių sąvokas, susijusias su struktūriniais duomenų aspektais, kadangi duomenų struktūros elementai naudojami formuojant užklausas.

2.1.1. Duomenų modeliai

Reliacinio duomenų modelio atveju duomenys vaizduojami ryšiais. Ryšys apibrėžiamas kaip aibė kortežų, kurie savo ruožtu yra sudaryti iš nesurikiuotų atributų reikšmių aibės. Atributo reikšmę sudaro jo pavadinimas ir tipo pavadinimas. Atributo reikšmė – konkreti galima atributo tipo reikšmė. Praktikoje ryšiai vizualiai paprastai vaizduojami lentelėmis.

XML duomenų modelis yra abstraktus – XML teikia tiktai gaires sudėtingesniems modeliams kurti. XML dokumente duomenys saugomi hierarchinėje, medį primenantį struktūroje. Mazgais vadinamos medžio viršūnės. Egzistuoja kelių tipų mazgai: elemento, dokumento, apdorojimo instrukcijos, komentaro, atributo, vardų srities ir teksto. Kiekvienas mazgas gali turėti vaikų, tai yra būti sujungtas su žemesnio lygio mazgais.

2.1.2. Struktūra ir operacijos

Išskirtinos trys užklausas sudarančios pagrindinės dalys:

- dalis, įvardijanti išrenkamą (ar keičiamą, šalinamą, kuriamą) informaciją;
- dalis, įvardijanti duomenų saugojimo elementą (-us), iš kurio (-ų) informacija turi būti išrenkama;
- dalis, kurioje apibrėžiama informacijos atrankos (ar keitimo, šalinimo) sąlyga; ši dalis užklausoje nėra būtina, tačiau dažniausiai naudojama.

Skirtingos užklausų kalbos vykdo ne visas galimas operacijas – galimybė užklausų kalboms vykdyti konkrečias operacijas pateikta 1 lentelėje.

1 lentelė. Galimos užklausų kalbų operacijos su duomenimis

Funkcinė savybė/ užklausų kalba	SQL	XPath	XQuery
Duomenų išrinkimas	+	+	+
Naujų duomenų elementų kūrimas	+	-	+
Duomenų šalinimas	+	-	-
Duomenų keitimas	+	-	-

Galima atkreipti dėmesį, kad visos užklausų kalbos gali vykdyti tik duomenų išrinkimo operaciją, todėl tolesnėje analizėje ir jos rezultatų panaudojime koncentruojamasi į išrinkimo funkcijas.

2.1.3. Savybių palyginimo rezultatai

Toliau išskirta aibė kriterijų, pagal kuriuos lyginamos užklausų kalbos: operatoriai, reikšmių grupavimo, agregavimo, operacijų su aibėmis galimybės ir kitos. Straipsnyje pateikti tik atlikto palyginimo visų užklausų lyginimo kiekvieno kriterijaus atveju rezultatai (2 lentelė). Jeigu kriterijų – funkcinę užklausų galimybę ar jų savybę yra galimybė kalboje galimybė naudoti, tuomet lentelėje pažymėtas „+“; priešingu atveju – „–„.. Tam tikrais atvejais, kur tikslinga, pateikiamas kalbos teikiamos galimybės ar savybės paaiškinimas.

2 lentelė. Užklausų kalbų savybių palyginimas

Lyginimo kriterijus		SQL	XPath	XQuery
Užklausų rezultato semantika		Išrenkami duomenys iš stulpelių sudarytos lentelės forma	Atominių reikšmių arba XML dokumento mazgų seka	Atominių reikšmių arba XML dokumento mazgų seka
Operatoriai	Lyginimo	aritmetiniai (>, <, >=, <=, =, <>) IN between like	aritmetiniai (>, <, >=, <=, =, <>; IN, between atitikmenis galima realizuoti naudojant sudėtinę sąlygą	aritmetiniai (>, <, >=, <=, =, <>); IN, between atitikmenis galima realizuoti naudojant sudėtinę sąlygą
	Loginiai	And, Or, Not	And, Or; Not funkcija	And, Or; Not funkcija
	Aritmetiniai	+, -, *, /	+, -, *, div, mod	+, -, *, div, mod
Pirminės eilės tvarkos išlaikymas rezultate		Neišlaiko	Išlaiko	Išlaiko

Kvantifikuotos išraiškos	Egzistencinė ir universali kvantifikacija	Egzistencinė ir universali kvantifikacija	Egzistencinė ir universali kvantifikacija
Nulinė (null) reikšmė	Turi Null reikšmę	Neturi	Neturi; artimiausias analogas – tuščia seka
Agregatinės funkcijos	+	+	+
Grupavimas	+	–	Galima atlikti naudojant specialią distinct-values() funkciją
Užklausų įdėjimas (angl. *nesting*)	+	+	+
Sujungimai (angl. *joins*)	Naudojami įvairių rūšių sujungimai	Tiesioginio atitikmens nėra; galima naudoti theta sujungimą	Pasiekiamas analogiškas efektas naudojant for-let kombinaciją
Operacijos su aibėmis	Sąjunga, sankirta, skirtumas	Sąjunga, sankirta, skirtumas	Sąjunga, sankirta, skirtumas
Funkcijų perklotis	+	–	–
Rekursija ir vartotojo apibrėžtos funkcijos	+	+	+

2.2. Užklausų vykdymas

Kaip jau buvo minėta, visos trys užklausų kalbos teturi vieną bendrą – informacijos išrinkimo – galimybę. Tokių užklausų paskirtis – grąžinti pagal nurodytus kriterijus atrinktų duomenų rezultatą. Šia savybe išsiskiria būtent paieškos sistemos. Taigi tikslinga būtų modeliu apibendrintus analizės rezultatus pritaikyti šio tipo sistemoms.

Paieškos sistemose paprastai vartotojui pateikiama sąsaja, kurioje šis įveda formuojamos užklausos kriterijus, nurodo rezultatų rikiavimo tvarką ir pan. Taigi realiai vartotojas, nesusidurdamas tiesiogiai su konkrečios užklausų kalbos konstruktais, suformuoja užklausą, ir tokia užklausa gali būti skirta tiek reliaciniam, tiek XML duomenų šaltiniui.

Remiantis šia išvada, naudinga turėti tokį modelį, kuris būtent tiktų tiek bet kurios nagrinėjamos užklausų kalbos užklausoms reikšti, tiek būtų jį galima tiesiogiai susieti su vartotojui pateikiama vizualia užklausos išraiška. Taigi modelis kuriamas, turint idėją jį panaudoti paieškos sistemos kūrimo proceso automatizavime.

3. Metamodelio sudarymas

3.1. Modeliavimo formos pasirinkimas

Siekiant suteikti modeliui aiškumo, jis kuriamas naudojant standartinę UML kalbą – būtent dėl vizualus modelio elementų pateikimo ji turi pranašumą prieš kitus modeliavimo būdus. Be to, būdama standartu, ji suteikia galimybę didesniam projektuotojų ratui tiksliai interpretuoti diagramų elementų prasmę bei, prireikus, atlikti pakeitimus modelyje. Pats kuriamas modelis tuo pačiu yra ir metamodelis – jis išreiškia duomenis apie duomenis, todėl toliau straipsnyje taip ir vadinamas.

3.2. Struktūrinio ir dinaminio metamodelių derinys

SQL, XPath ir XQuery analizės metu daugiausia dėmesio skirta užklausų struktūrai. Tačiau metamodelis yra kuriamas atsižvelgiant į jo panaudojimą automatizuojant paieškos sistemų kūrimą. Paieškos sistemos gali vartotojams teikti šias galimybes:

- nurodyti tam tikrą kiekį paieškos kriterijų,
- galimybę paieškos kriterijus pasirinkti iš sąrašo,
- galimybę pasirinkti – vykdyti tikslią paiešką ar ne,
- paieškos kriterijai jungiami pasirenkamais loginiais operatoriais,
- vartotojui pateikiamas sąrašas lyginimo operatorių,
- tekstiniame lauke galima ranka įvesti užklausos eilutę (supaprastinta kalba) – tai paprastai daugiau skiriama patyrusiems vartotojams, kuriems patogiau naudotis ne vizualiais sąsajos elementais, o pateikti užklausos tekstą,
- pasirenkamas rikiavimo kriterijus ir rikiavimo tvarka,
- išrenkamų rezultatų kiekio apribojimas,
- išrinktų rezultatų tikslumo įvertis,
- suteikiama galimybė atlikti paiešką tarp rezultatų.

Visos nurodytos savybės, išskyrus dvi paskutiniąsias, susijusios su struktūrinėmis užklausų savybėmis. Tuo tarpu paskutiniosios susijusios su užklausos vykdymu, ir jas galima priskirti prie dinaminių paieškos sistemų savybių. Taigi tikslinga išskirti ir dvi metamodelio dalis: pagrindinį užklausų metamodelį, išreiškiantį užklausų struktūrą, ir posistemės metamodelį, perteikiantį su vykdymu susijusias savybes.

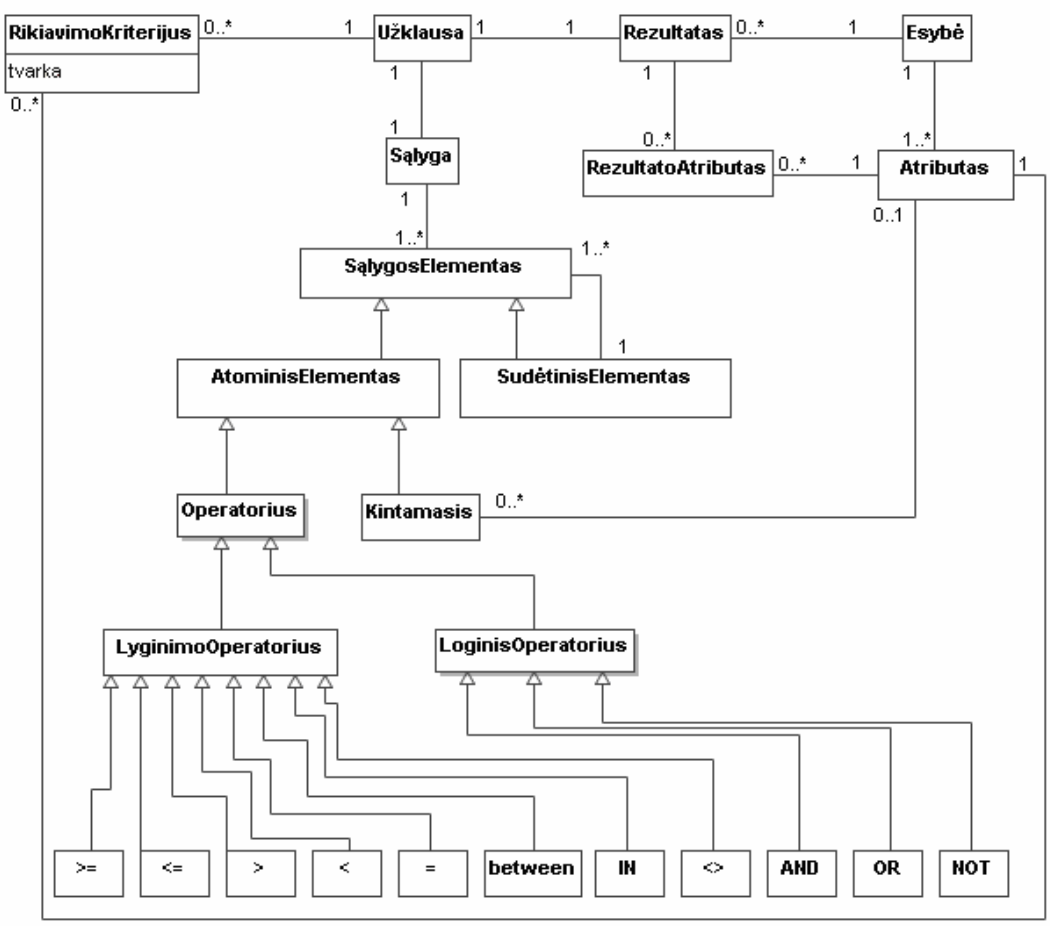
3.3. Užklausų metamodelio sudarymas

Analizės dalyje jau išskirtos pagrindinės užklausų dalys:

- išrenkama informacija, kuri bus vadinama *rezultatu*,
- atrankos sąlyga, kuri bus vadinama *sąlyga*,
- duomenų struktūra, iš kurios išrenkamas rezultatas, kuris bus vadinamas *esybe*, atitinkančia lenteles SQL atveju (arba ryšius reliacinio duomenų modelio kontekste) bei elementus XML dokumento atveju.

Užklausos *rezultatas* – tai esybės atributų aibė reliacinės duomenų bazės atveju arba mazgo atributų aibė arba mazgo elementų aibės tekstinis turinys (angl. *text content*) XML dokumento atveju.

Kita užklausos sudėtinė dalis, *sąlyga*, - tai užklausos atrankos kriterijų aibė, turinti bent vieną elementą, kuris pažymimas klase *SąlygosElementas* (1 pav.).



1 pav. Užklausų metamodelis

Sąlygos elementą sudaro operatoriais jungiami esybės atributai. Operatorius apibendrina klasė *Operatorius*, o sąlygos atributus – *Kintamasis*. Abiems šiems sąlygos elementams apibendrinti naudojama klasė *AtominisElementas*, o ryšiui tarp *AtominisElementas* bei *Operatorius* ir *Kintamasis* – UML klasių agregavimo ryšys. Kiekvieną sąlygos elementą sudaro aibė operatorių ir kintamųjų – jie jungiami į klasę *SudėtinisElementas*. Taigi sąlygos elementas gali būti arba atominis arba sudėtinis, ir šiam ryšiui pažymėti vėlgi naudojamas agregavimo ryšys. Sudėtinio elemento dalimi yra aibė sąlygos elementų, todėl panaudojamas rekursinis ryšys tarp *SąlygosElementas* ir *SudėtinisElementas* klasių. Išskiriami dviejų tipų operatoriai, naudojami kaip atominiai sąlygos elementai: lyginimo ir loginiai operatoriai. Jų naudojimas ir galimos reikšmės buvo pateiktos užklausų kalbų lyginimo rezultatus apibendrinančioje 2 lentelėje.

Galiausiai užklausai galima nurodyti rikiavimo kriterijus, kuriems skiriama klasė *RikiavimoKriterijus*. Pats rikiavimo kriterijus yra esybės atributas. Rikiavimas gali būti organizuojamas didėjančia arba mažėjančia tvarka, todėl rikiavimo tvarkos požymis prilyginamas UML klasės atributui.

3.4. Užklausų posistemės metamodelio sudarymas

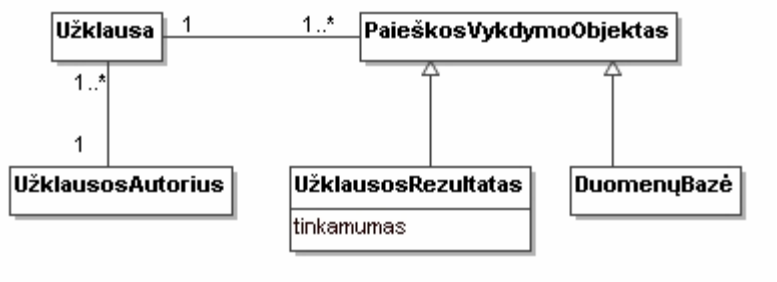
Sudarytame metamodelyje dar neatsižvelgta į šiuos aspektus:

- kiekvienas vartotojas formuoja savo užklausą, t. y. konkretus užklausos egzempliorius siejamas su jos autoriumi, ir šiam pateikiami konkretaus užklausos egzemplioriaus rezultatai. Autorius neatitinka fizinio vartotojo, kadangi tas pats vartotojas gali vienu metu (pavyzdžiui, keliuose naršyklės languose) pateikti kelias užklausas;
- sykį atlikus užklausą ir gavus jos rezultatus, kita užklausa (ar kitos užklausos) gali būti vykdoma (vykdomos) tarp gautų paieškos rezultatų. Taigi turi būti saugoma kiekvienos užklausos rezultatų versija;
- skaičiuojamas kiekvienos užklausos rezultatų atitikimas nurodytiems kriterijams.

Būtent į juos atsižvelgiant formuojamas užklausų posistemės metamodelis. Užklausos autorius-vartotojas gali pateikti vieną ar kelias užklausas. Šie objektai atitinkamai vaizduojami klasėmis *UžklausosAutorius* ir *Užklausa*. Galimi du užklausos vykdymo variantai:

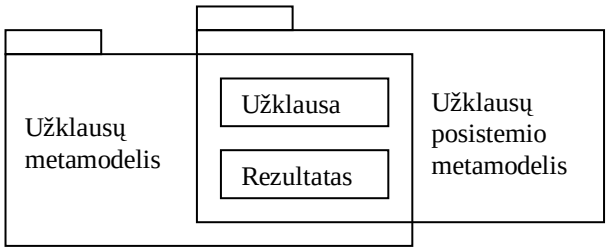
- Užklausa gali būti vykdoma kiekvieną kartą atliekant paiešką tarp duomenų bazėje (reliacinėje ar XML dokumentuose) saugomų duomenų.
- Atlikus užklausą, išsaugoma jos rezultatų kopija. Vykdant užklausą antrąjį kartą, gali būti nurodoma, ar paieška atliekama paskutinėje rezultatų versijoje, ar vėl kreipiamasi į duomenų bazėje saugomas pagrindinių duomenų struktūras (tai yra ne į ankstesnius rezultatus).

Abstrakčiai duomenų struktūrai, iš kurios išrenkami duomenys užklausos vykdymo metu, skiriama klasė *PaieškosVykdymoObjektas*, apibendrinanti *UžklausosRezultatas* ir *DuomenųBazė* klases, skirtas atitinkamai užklausų rezultatų versijoms bei duomenų bazėje saugomų pagrindinių duomenų struktūroms pažymėti. Galutinė paieškos posistemės metamodelio diagrama pateikta 2 paveiksle. Papildomai *UžklausosRezultatas* turi tinkamumo atributą – kiekvieniems užklausos rezultatams galima taikyti požymį, parodantį, kiek rasta informacija atitinka paieškos kriterijus.



2 pav. Užklausų posistemės metamodelis

Abu – užklausų bei jos posistemės – metamodeliai nėra atskiri: abiejuose iš jų vaizduojama ta pati užklausa, pažymint ją *Užklausa* klase. Taip pat užklausų metamodelio *Rezultatas* klasė turi tą pačią prasmę, kaip ir posistemio metamodelio *UžklausosRezultatas* klasė. 3 paveiksle pavaizduota, kaip metamodeliai atrodo visumoje, bei bendri jų elementai.



3 pav. Sąsaja tarp pagrindinio užklausų metamodelio bei užklausų posistemės metamodelio

4. Paieškos sistemų kūrimo automatizavimas

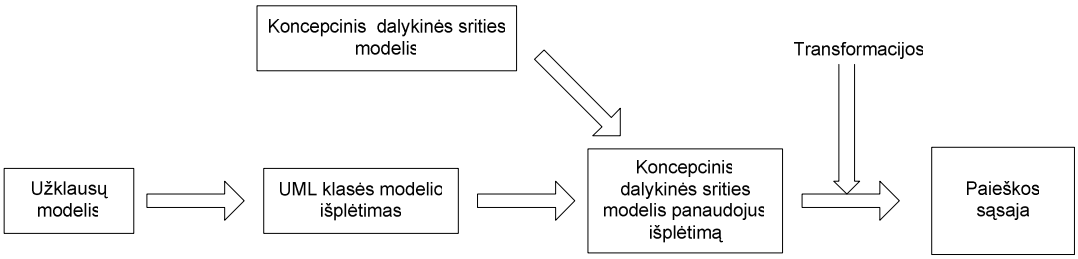
Sudarytas metamodelis gali būti taikomas paieškos sistemoms kūrimo procesui automatizuoti. Paieškos sąsaja, leidžianti vartotojui atsiriboti nuo fizinės užklausos išraiškos, gali būti tiesiogiai gaunama taikant tam tikras taisykles. Tai yra įmanoma naudojant užklausų metamodelio teikiamą informaciją ir atliekant kai kurias papildomas transformacijas.

Aprašant sąsajos kūrimo automatizavimo procesą, daroma prielaida, kad turimas konkrečios dalykinės srities, kuriai kuriama paieškos sistema, koncepcinis modelis, sudarytas naudojant UML kalbą. Šio proceso etapai ir jų aprašymai pateikti 3 lentelėje.

3 lentelė. Paieškos sistemos sąsajos kūrimo automatizavimo proceso etapai

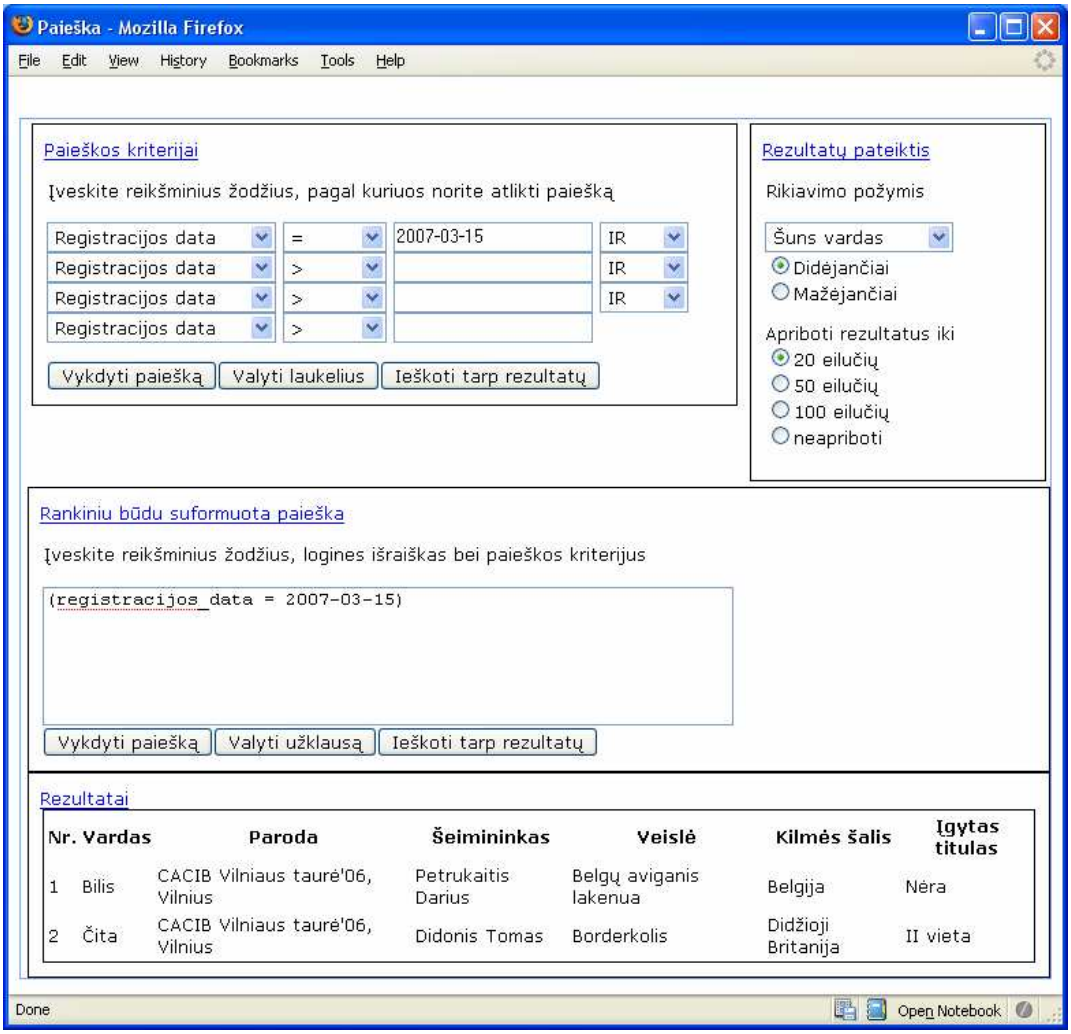
Etapai	Apibūdinimas
Sukuriamas UML klasės modelio išplėtimas, remiantis metamodeliu	UML kalba teikia galimybes sukurti profilį, išplečiantį UML galimybes pritaikant standartinius elementus naudojimui konkrečioje srityje. Taigi sukuriamas UML profilis, apibūdinantis kintamus (kiekvienos paieškos srities atveju) paieškos sistemų elementus – paieškos kriterijus, objektus, rikiavimo kriterijus, galimą kriterijų skaičių. Tam naudojami UML išplėtimo mechanizmai – stereotipai ir žymėtosios reikšmės.
Koncepcinio modelio elementams suteikia papildoma prasmė, naudojant profilio stereotipus ir žymėtąsias reikšmes	Koncepciniame modelyje pateikiamos dalykinės srities esybės ir jų atributai. Atributai, kurie viena ar kita forma pateikiami vartotojui paieškos sąsajoje (pvz., atributai, pagal kuriuos užklausos rezultatai turi būti rikiuojami), pažymimi stereotipais. Paieškos sąsajos elementams, kurie tiesiogiai nepriklauso nuo koncepcinės srities, pvz., paieškos kriterijų eilučių skaičius, nustatomos žymėtųjų reikšmių reikšmės.
Papildytas koncepcinis modelis yra apdorojamas taikant pasirinktą technologinį metodą	Iš koncepcinio modelio ir UML metamodelio generuojama sąsaja. Tai patogu atlikti išnaudojant UML projektų saugojimo XMI faile galimybę. Šis failas apibrėžta standartus objektiškai orientuotoms sistemoms kuriančiu OMG (angl. Object Management Group) organo; failas yra XML formato, todėl patogu sąsajai generuoti naudoti formatui skirtą XSLT transformacijų kalbą. Šiuo tikslu sukuriamas XSLT dokumentas, kuriame aprašomos XMI failo transformavimo į reikiamą formatą (pvz., html) taisyklės. Dalis taisyklių apibūdina nekintančias sąsajos dalis, tai yra tokias, kurias galima taikyti visoms sistemoms: loginius ir lyginimo operatorius.
Sugeneruojama paieškos sąsaja	Jeigu naudojamas XSLT dokumentas, tuomet pagal jame aprašytas taisykles sugeneruojamas sąsajos html tekstas.

Kad būtų lengviau įsivaizduoti aprašytą procesą, jo pradiniai, tarpiniai ir galutiniai rezultatai vizualiai pateikti 4 pav.



4 pav. Paieškos sąsajos kūrimo, naudojant suformuotą užklausų metamodelį bei koncepcinį dalykinės srities modelį, eiga

Remiantis metamodeliu ir aprašyta tvarka, sukurtas prototipas, leidžiantis atlikti paiešką pagal įvairius kriterijus įvairius kriterijus įvairiose sistemos objektų aibėse. Prototipo vartotojo sąsaja pateikta 5 paveiksle. Taip pat prototipas suteikia galimybę, kaip aprašyta metamodelio posistemės dalyje, išsaugoti rezultatus bei atlikti paiešką tarp jų. Tokiu būdu įgyvendinama daug geresnė paieška, kadangi vartotojas gali tikslingai susiformuoti reikiamas užklausas. Pateiktą metodiką galima taikyti kuriant įvairias sistemas.



5 pav. Prototipo paieškos sąsaja

5. Išvados

Straipsnyje pateikta labiausiai praktikoje naudojamų – reliacinių ir XML – duomenų šaltinių užklausų kalbų SQL, XQuery ir XPath lyginamoji analizė. Panaši analizė iki šiol nebuvo atlikta, nors dedama daug pastangų padidinti semantines paieškos galimybes.

Atliekant analizę, susiformavo idėja, kad užklausų kalbos ir paieškos sistemos veikia panašiai, todėl galima sudaryti modelį, kuris leistų tokių sistemų kūrimą automatizuoti. Paieškos sistemos modelį tikslinga išvesti iš dalykinės srities, kurioje bus atliekama paieška, koncepcinio modelio, papildant jį paieškos sistemos stereotipais. Kad paieška būtų efektyvesnė, paieškos sistemą reikia ne tik kurti pagal užklausų struktūrą, bet ir papildyti ją nuoseklaus vykdymo elementais.

Pagal siūlomą metodiką realizuoto prototipo realizacija parodė, kad metodika įgyvendinama ir ją galima taikyti įvairioms sistemoms, kurios atlieka paiešką struktūrizuotuose informacijos šaltiniuose. Sudarytas modelis padidina semantines paieškos galimybes, kadangi vartotojas gali panaudoti rezultatus tolesniam užklausų patikslinimui

Literatūros sąrašas

[1] ANSI/ISO/IEC International Standard (IS) Database Language SQL – Part 1: SQL/Framework. September 1999

[2] **Bonifati A., Ceri S.** Comparative Analysis of Five XML Query Languages. *ACM SIGMOD Record*, 2000, Vol. 29, No 1. p. 68-79

[3] **Chamberlin, D.** XQuery: An XML query language. *IBM Systems Journal*, 2002, Vol. 41, No. 4, p. 597-615

[4] **Eisenberg A., Kulkarni K., Melton J., Michels J.-E., Zemke F.** SQL:2003 Has Been Published. SIGMOD Record, 2004, Vol. 33, No. 1., p. 119-126

[5] **Melton J.** SQL, XQuery and SPARQL: What's Wrong With This Picture? [interaktyvus] *XML 2005 Conference Proceedings* [žiūrėta 2006-05-01]. Prieiga per internetą: http://xtech06.usefulinc.com/schedule/paper/119

[6] **Ozcan F., Chamberlin D., Kulkarni K., Michels J.-E.** Integration of SQL and XQuery. *IBM Systems Journal*, 2006, Vol. 45, No. 2, p. 245-270

[7] **Pal S., Cseri I., Seeliger O., Rys M., Schaller G., Yu W., Tomic D., Baras A., Berg B., Churin D., Kogan E**. XQuery Implementation in a Relational Database System. *Proceedings of the 31st international conference on Very large data bases, VLDB Endowment*, Trondheim, 2005, p. 1175-1186.

[8] XML Path Language [interaktyvus]. W3C, 1999 [žiūrėta 2006-02-07]. Prieiga per internetą: http://www.w3.org/TR/xpath

[9] XQuery 1.0: An XML Query Language [interaktyvus], W3C, 2006 [žiūrėta 2006-02-24]. Prieiga per internetą: http://www.w3.org/TR/xquery/

Query Metamodel for Automated Development of Search Systems

The article presents the analysis of SQL, XQuery and XPath query languages that are intended for the most widely used data sources, namely relational databases and XML documents. Based on the results of the analysis, a generic query model has been developed. The model should serve to help compose user queries without attaching to the context of a specific data source implementation technology. The applicability of the model in practice is demonstrated by creating methods for query interface development, based on this model.

2 priedas. Transformacijos

TransformacijaSasajai.xsl

```
<?xml version="1.0" encoding="windows-1257" ?>
<!-- xslt dokumentas analizuoja XMI faila ir isveda sasajos html -->
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:uml="http://schema.omg.org/spec/UML/2.0"
                xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"

xmlns:SearchSystem="http://www.magicdraw.com/schemas/SearchSystem.xmi"

xmlns:Subsystem="http://www.magicdraw.com/schemas/Subsystem.xmi">
  <xsl:output method="html"/>
  <!-- Root template -->
  <xsl:template match="/">
    <html>
      <head>
        <meta name="Author" content="Arjana Matonytė"/>
        <meta name="Keywords" content="Parodos, šunys, veislės"/>
        <meta name="Description" content="Šunų parodų paieška"/>
        <link rel="stylesheet" type="text/css" href="stilius.css"/>
        <title>Paieška</title>
      </head>
      <body>
        <div class="puslapis">
          <table class="virsus" width="100%">
            <!-- Du viršutiniai stulpeliai - Paieškos kriterijai ir
                 rezultatų pateiktis
            -->
            <tr>
              <td valign="top" width="70%">
                <div class="grupe">
                  <p>
                    <span class="grupesPav">Paieškos sritis</span>
                  </p>
                  <form name="simpleSearch" method="POST"
                        action="callQuery.xml">
                    <select name="selSearchObject">
                      <xsl:for-each
select="/xmi:XMI/SearchSystem:SearchObjectID">
                        <xsl:variable name="paieskosObjektas"
                                      select="@base_Element"/>
                        <!--<xsl:for-each
select="//uml:Model/ownedMember[@xmi:type='uml:Profile']/ownedMember[@
xmi:type='uml:Class']/ownedAttribute[@xmi:id=$paieskosObjektas]"> -->
```

```
                        <xsl:for-each
select="//uml:Model/ownedMember[@xmi:type='uml:Profile']/ownedMember[@
xmi:type='uml:Class' and ownedAttribute/@xmi:id=$paieskosObjektas]">
                          <option value="{@name}">
                            <!--<xsl:value-of select="atributoZyme"/>-
->
                            <xsl:value-of select="@name"/>
                          </option>
                        </xsl:for-each>
                      </xsl:for-each>
                    </select>
                    <br/>
                    <br/>
                    <span class="grupesPav">Paieškos kriterijai</span>
                    <p>Įveskite reikšminius žodžius, pagal kuriuos
norite
                       atlikti paiešką</p>
                    <select name="selSearchCriterion">
                      <xsl:for-each
select="/xmi:XMI/SearchSystem:SearchCriterion">
                        <xsl:variable name="paieskosKriterijus2"
                                      select="@base_Element"/>
                        <xsl:for-each
select="//uml:Model/ownedMember[@xmi:type='uml:Profile']/ownedMember[@
xmi:type='uml:Class']/ownedAttribute[@xmi:id=$paieskosKriterijus2]">
                          <option value="{@name}">
                            <xsl:value-of select="atributoZyme"/>
                          </option>
                        </xsl:for-each>
                      </xsl:for-each>
                    </select>
                    <!-- lyginimo operatoriai -->
                    <select name="selComparisonOperator">
                      <xsl:variable name="lyginimoOperatorius"

select="//ownedMember[@xmi:type='uml:Package' and
@name='QueryMetamodel']/ownedMember[@xmi:type='uml:Class' and
@name='LyginimoOperatorius']/@xmi:id"/>
                      <xsl:for-each
select="//ownedMember[@xmi:type='uml:Package']/ownedMember[generalizat
ion/@general=$lyginimoOperatorius]">
                        <option value="{@name}">
                          <xsl:value-of select="atributoZyme"/>
                        </option>
                      </xsl:for-each>
                    </select>
                    <!-- laukelis paieskos kriterijui ivesti -->
                    <input type="text"
name="txtPaieskosKriterijus"></input>
                    <xsl:apply-templates
select="xmi:XMI/SearchSystem:SearchSubsystem"/>
                    <p/>
                    <button name="btnVykdytiPaieska">Vykdyti
paiešką</button>
                    <button name="btnValytiLaukelius">Valyti
laukelius</button>
                    <button name="btnTarpRezultatu">Ieškoti tarp
rezultatų</button>
                  </form>
                </div>
              </td>
              <td valign="top">
                <div class="grupe">
                  <span class="grupesPav">Rezultatų pateiktis</span>
```

```
                <p>Rikiavimo požymis</p>
                <form action="rezults.xml" method="POST">
                  <!-- rikiavimo kriterijus -->
                  <select>
                    <xsl:for-each
select="/xmi:XMI/SearchSystem:SortCriterion">
                      <xsl:variable name="rikiavimoKriterijus2"
                                    select="@base_Element"/>
                      <xsl:for-each
select="//uml:Model/ownedMember[@xmi:type='uml:Profile']/ownedMember[@
xmi:type='uml:Class']/ownedAttribute[@xmi:id=$rikiavimoKriterijus2]">
                        <option value="{@name}">
                          <xsl:value-of select="atributoZyme"/>
                        </option>
                      </xsl:for-each>
                    </xsl:for-each>
                  </select>
                  <br/>
                  <input type="radio" checked="checked"
name="tvarka"
                         value="didejanti"/>
                  Didėjančiai
                  <br/>
                  <input type="radio" name="tvarka"
value="mazejanti"/>
                  Mažėjančiai
                </form>
                Apriboti rezultatus iki
                <form action="">
                  <input type="radio" checked="checked"
name="rezRiba"
                         value="eil20"/>
                  20 eilučių
                  <br/>
                  <input type="radio" name="rezRiba" value="eil50"/>
                  50 eilučių
                  <br/>
                  <input type="radio" name="rezRiba"
value="eil100"/>
                  100 eilučių
                  <br/>
                  <input type="radio" name="rezRiba" value="eil0"/>
                  neapriboti
                  <br/>
                </form>
              </div>
            </td>
          </tr>
        </table>
        <!-- Vidurinioji juosta - rankiniu būdu suformuota paieška -
->
        <div class="grupe">
          <span class="grupesPav">Rankiniu būdu suformuota
paieška</span>
          <p>Įveskite reikšminius žodžius, logines išraiškas bei
paieškos
             kriterijus</p>
          <textarea rows="5" cols="50"></textarea>
          <div class="mygtukuGrupe">
            <button name="btnRankVykdytiPaieska">Vykdyti
paiešką</button>
            <button name="btnRankValytiLaukelius">Valyti
užklausą</button>
            <button name="btnRankTarpRezultatu">Ieškoti tarp
```

```
rezultatų</button>
            </div>
          </div>
          <div class="grupe">
            <span class="grupesPav">Rezultatai</span>
          </div>
        </div>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="xmi:XMI/SearchSystem:SearchSubsystem">
    <xsl:call-template name="for">
      <xsl:with-param name="n" select="@CriteriaCount"/>
      <xsl:with-param name="i" select="1" />
    </xsl:call-template>
  </xsl:template>
  <xsl:template name="for">
    <xsl:param name="i" select="$i"/>
    <xsl:param name="n" select="$n" />
    <xsl:if test="$i &lt; $n">
    <xsl:call-template name="queryRow"></xsl:call-template>
      <xsl:call-template name="for">
        <xsl:with-param name="i" select="$i+1"/>
        <xsl:with-param name="n" select="$n"/>
      </xsl:call-template>
    </xsl:if>
  </xsl:template>
  <xsl:template name="queryRow">
    <!-- loginiai operatoriai -->
    <select>
      <xsl:variable name="loginisOperatorius"
                    select="//ownedMember[@xmi:type='uml:Package' and
@name='QueryMetamodel']/ownedMember[@xmi:type='uml:Class' and
@name='LoginisOperatorius']/@xmi:id"/>
      <xsl:for-each
select="//ownedMember[@xmi:type='uml:Package']/ownedMember[generalizat
ion/@general=$loginisOperatorius]">
        <option value="{@name}">
          <xsl:value-of select="atributoZyme"/>
        </option>
      </xsl:for-each>
    </select>
    <br/>
    <!-- paieskos kriterijus -->
    <select>
      <xsl:for-each select="/xmi:XMI/SearchSystem:SearchCriterion">
        <xsl:variable name="paieskosKriterijus2"
select="@base_Element"/>
        <xsl:for-each
select="//uml:Model/ownedMember[@xmi:type='uml:Profile']/ownedMember[@
xmi:type='uml:Class']/ownedAttribute[@xmi:id=$paieskosKriterijus2]">
          <option value="{@name}">
            <xsl:value-of select="atributoZyme"/>
          </option>
        </xsl:for-each>
      </xsl:for-each>
    </select>
    <!-- lyginimo operatoriai -->
    <select>
      <xsl:variable name="lyginimoOperatorius"
                    select="//ownedMember[@xmi:type='uml:Package' and
@name='QueryMetamodel']/ownedMember[@xmi:type='uml:Class' and
@name='LyginimoOperatorius']/@xmi:id"/>
      <xsl:for-each
```

```
select="//ownedMember[@xmi:type='uml:Package']/ownedMember[generalizat
ion/@general=$lyginimoOperatorius]">
        <option value="{@name}">
          <xsl:value-of select="atributoZyme"/>
        </option>
      </xsl:for-each>
    </select>
    <!-- laukelis paieskos kriterijui ivesti -->
    <input type="text" name="txtPaieskosKriterijus"></input>
  </xsl:template>
</xsl:stylesheet>
```

BendrasXmi.xsl

```
<?xml version="1.0" encoding="utf-8"?>

<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xmi='http://schema.omg.org/spec/XMI/2.1' >

  <xsl:output method="xml" indent="yes" />
  <xsl:template match="/">
    <xsl:apply-templates select="node()" />
  </xsl:template>

  <xsl:template match="//ownedMember/generalization">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()" />
    </xsl:copy>
    <xsl:variable name="name" select="../@name" />
    <atributoZyme>
      <xsl:value-of
select="document('pavadinimai.xml')/pavadinimai/elementas[@id=$name]/e
lLt/text()" />
    </atributoZyme>
  </xsl:template>

  <xsl:template
match="//ownedMember[@xmi:type='uml:Profile']/ownedMember[@xmi:type='u
ml:Class']/ownedAttribute">
    <xsl:copy>
      <xsl:apply-templates select="@*" />
      <xsl:variable name="cname" select="../@name" />
      <xsl:variable name="aname" select="@name" />
      <atributoZyme>
        <xsl:value-of
select="document('pavadinimai.xml')/pavadinimai/klase[@id=$cname]/atri
butas[@id=$aname]/atributoZyme/text()" />
      </atributoZyme>
      <xsl:apply-templates select="node()" />
    </xsl:copy>
  </xsl:template>


  <!-- Praleisti egzistuojančius elementus -->
  <xsl:template match="atributoZyme|atributoZyme/text()">
    <xsl:apply-templates select="@*|node()" />
  </xsl:template>

  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()" />
    </xsl:copy>

  </xsl:template>
```

```
</xsl:stylesheet>
```

3 priedas. Duomenų bazės esybių-ryšių diagrama