KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS
Kompiuterių katedra

Magistrinis darbas

Informacijos valdymo metodų analizė ir sprendimas informacijos paieškai naudojant ontologijas

Analysis of information control methods and solution to information search using ontologies

Magistrantas: M.Nekroševičius

Vadovas: lekt. Dr. A.Janavičiūtė

KAUNAS, 2009

Turinys

Įvadas

Šiuolaikiniam verslui yra būdingas pastovus aplinkos, kurioje dirba, adaptuojasi ir išgyvena organizacijos ir visuomenės, kitimas. Konkurencijos sąlygomis verslo likimą didele dalimi lemia kompanijos reakcijos į išorinės aplinkos pokyčius greitis ir tikslumas, o tai reikalauja vadyboje panaudoti naujas koncepcijas, technologijas ir instrumentarijų. Organizacijų veikla šiuo metu vis labiau priklauso nuo jų turimų žinių ir galimybės jas efektyviai panaudoti. Tačiau priemonės, skirtos žinių vaizdavimui, dar nėra pakankamai išvystyti ir dažnai reikalauja iš naujo ieškoti tų pačių problemų kitų sprendimų.

Žinių valdymas šiandien nagrinėjamas kaip stiprus konkurencinis pranašumas įmonėje, kuri orientuota į pastovius verslo procesų pokyčius. Bet nei informacinės technologijos, nei duomenys negali užtikrinti konkurencinį pranašumą ilgame periode. Konkurenciniai pranašumai gali būti pasiekti tik transformuojant informaciją į vertingas ir prasmingas veiklos komandas [20]. Praktiškai visos kompanijos turi didelį kiekį duomenų ir praktinio patyrimo. Bet kol kas ši informacija yra paskirstyta duomenų bazėse, dokumentų saugyklose, elektroninio pašto pranešimuose ir pan. Problema glūdi tame, kaip organizuoti prieigą prie visų šitų duomenų, suteikus jiems vartotojui patogią formą.

Tačiau, informacijos rinkinys pasaulyje padalintas per vieną ar daugiau duomenų bazių mazgų iš sistemos topologijų plataus rėžio, paaiškinimui įtraukiant autonomiją, mastelį, pasiekiamumą, įvykdymą ir apsaugą. To tikslas yra naudoti keletą nepriklausomų informacijos šaltinių kaip vieną. Tai įgalina informacijos užklausas, šaltinį ir atradimo paslaugą ir bendravimo funkcionalumą veikti sistemoje kaip vieną visumą, negu duotą jos dalį.

Tokiu būdu žinių valdymas – tai organizacijos strategija, kurios tikslas išskirti visą reikalingą jai informaciją, darbuotojų patirtį ir kvalifikaciją, kad užtikrinti vartotojų aptarnavimo kokybę ir sutrumpinti reakcijos į aplinkos pokyčius laiką. Vienas iš perspektyviausių žinių formalizacijos, kuri suteikia galimybes panaudoti sukauptas žinias kompiuteriniam apdorojimui, krypčių yra ontologijos [21]. Žinių inžinerijos kontekste ontologijos yra aibė (žodynas) atvaizduotų joje terminų, organizuotų į taksonomiją. Tokioje ontologijoje sąvokos susieja dalykinės srities esybių (klasės, santykiai, funkcijos ir kiti objektai) vardus su natūralios kalbos tekstu, aprašančiu, ką reiškia šie vardai, ir formaliomis aksiomomis, ribojančiomis terminų interpretavimą ir konkretų panaudojimą.

Šiuo metu vykdoma daug darbų ontologijų srityje, tame tarpe ir darbų, susijusių su informacijos paieškos intelektualizacija, ypač interneto aplinkoje. Bendras tokių projektų tikslas yra sukurti naujus žinių erdvės vaizdavimo būdus ir priemonės darbui su jais.

Žinių portalų, orientuotų į informacijos paiešką internete, ontologijų ypatybė yra tai, kad juose be tradicinių dalykinės srities aprašų yra ir tinklinių resursų aprašai. Tokie aprašai leidžia duomenis paskirstytose duomenų bazėse, turinčiose skirtingus informacijos vaizdavimo būdus, apjungti ir vykdyti paiešką metaduomenų lygmenyje. Tokia ontologija pradinio indeksavimo pagrindu grupuoja interneto resursų nuorodas į kategorijas ir susieja jas su sąvokomis, kurios yra jose aprašytos. Tinklo resursų savybių pagrindu ontologija atseka tarpusavio ryšius tarp informacijos šaltinių tam kad susietų juos pagal užklausos tematiką, grupe ir kitus parametrus. Žinių portalų ontologijos naudojamos ne tik informacijos lokalizacijai indeksuotuose šio portalo resursuose, bet ir užklausos formulavimo patikslinimui, atliekant užklausą visame tinkle.

Temos aktualumas ir naujumas. Šiuo metu yra daug darbų, nagrinėjančių duomenų ir žinių atvaizdavimą. Yra sudaryti tokių sistemų veikimo scenarijai. Tačiau nėra labai daug darbų, liečiančių vartotojo bendravimo priemones su tokiomis sistemomis. Pagrindinę vietą čia užima vartotojo sąsaja. Tokiose sistemose vartotojo komunikacija turi būti vykdoma naudojant sąsają, kuri padeda vartotojui formuoti užklausas, pateikti alternatyvius terminus, apibendrinimus, patikslinimus ir pan., remiantis ontologijos samprata, kuri apjungia paskirstytų duomenų bazių informaciją metaduomenų lygmenyje ir tinklinių resursų aprašus.

Šiame darbe bus atlikta vartotojo sąsajos planavimo, projektavimo principų analizė. Apžvelgta vartotojo sąsajos su metaduomenų ir tinklinių resursų aprašais kūrimo ir įdiegimo problematika. Kuriama vartotojo sąsaja informacijos gavimui žinių inžinerijos pagalba leis padidinti organizacijos darbo našumą, nes jie gali rasti reikiamą informaciją greičiau ir apsaugoti save nuo informacijos peteklišKumo, kuo dabar pasižymi daugelis informacijos paieškos sistemų.

Tyrimo objektas. Informacijos, atitinkančios vartotojo poreikius, gavimas.

Darbo tikslas. Atlikti informacijos paieškos sistemų, kuriose taikomi metaduomenų ir tinklinių resursų aprašai, veikimo principų analizę. Pateikti vartotojo užklausų transformavimo galimybes, naudojant XML schemas, paieškai skirtingų tipų duomenų bazėse.

Darbo uždaviniai. Siekiant minėto darbo tikslo suformuluoti šie pagrindiniai darbo uždaviniai:

1. Atlikti metaduomenų ir tinklinių resursų aprašymo kalbų analizę.
2. Atlikti informacijos pateikimo ir išrinkimo paskirstytose, skirtingų tipų duomenų bazėse priemonių analizę.
3. Pateikti XML schemų panaudojimo užklausoms skirtingų tipų duomenų bazėse galimybes, įvertinant dalykinės srities ontologijas.

Naudojami metodai. Literatūros analizė ir programavimas.

1. Teorinė dalis

1.1. Pagrindinės informacijos valdymo problemos

Pastaruoju metu informacijos valdymo problema yra viena iš pagrindinių sričių, kuri būtina tolimesniam informacinių technologijų vystymui. Ši problema iškyla ne tik elektroninėse bibliotekose, kurios šiuo metu sėkmingai funkcionuoja informacinių paieškos sistemų pavidalu, kurios realizuoja vieną iš informacijos valdymo technologijų (duomenų bazių, internetinių technologijų arba tekstinių sistemų), bet ir tokių sistemų kūrimui, kurios integruoja tokias technologijas. Pastaruoju metu viena iš pagrindinių krypčių informacijos valdymo technologijų srityje tapo jų integracija konkrečių informacinių sistemų konkrečiose realizacijose, o viena iš perspektyviausių sričių – duomenų bazių technologijų integracija su internetinėmis technologijomis [20]. Duomenų bazių technologijos yra labai svarbios informacinėms paieškos sistemoms, nes jos atitinkamų duomenų bazių valdymo sistemų pagalba gali užtikrinti efektyvią prieigą prie įvairių struktūrizuotų duomenų, o vartotojo prieigos duomenų bazėse sukūrimas ontologijų pagrindu yra pripažintas, kaip vienas iš aktualiausių uždavinių.

Žinios naudojamos duomenims apdoroti ir taip versti juos į informaciją. Vadinasi, informacija siaurąja prasme, būtų antrinis produktas, duomenys – pirminis, o žinios – instrumentarijus, kurio pagalba duomenys yra apdorojami ir paverčiami informacija. Žmonės įvertina informaciją, priima sprendimus, atlieka veiksmus.

Šiuo metu egzistuoja du populiariausi informacijos pateikimo būdai – tekstų kolekcijos ir reliacinės duomenų bazės. Jos skiriasi prieigos prie informacijos būdais [22]. Reliacinėse duomenų bazėse užklausa formuojama objektine duomenų bazių sistemos kalba SQL[1]. Tokioms užklausoms būdingas sudėtinga struktūrizuota paieška ir sudėtingas rezultatų pateikimo organizavimas, rastos informacijos turinio pagrindu. Ištisinių tekstų atveju, užklausa formuojama raktinių žodžių pagalba, o rezultatas yra šių žodžių atitikimas atskiruose teksto fragmentuose (įvertinant morfologiją[2]) ir rezultatų pateikimas rastų dokumentų sąrašo pavidalu. Paieškos pagreitinimu naudojamas tekstų indeksavimo leksemomis (reikšmines vieno žodžio formas, pavyzdžiui, stalas, stalo, stalui ir t.t.) metodas.

Užklausos paprastumas ir informacijos paieškos ištisiniuose tekstuose universalumas suteikia galimybę panaudoti paieškos kriterijus, kuriuos užduoda vartotojas atskiriems ištisinių tekstų rinkiniams [11]. Reliacinėse duomenų bazėse prieiga prie saugomos jose informacijos vyksta iš anksto nustatytų procedūrinių modulių, su unikalia semantika, kas leidžia suformuluoti abstrakčią

[1] Structured Query Language - „struktūrizuota užklausų kalba“
[2] gr. *morphe* - forma, *logos* – mokslas. Žodžių formos ir jų reikšmės

užklausą visose duomenų bazėse. Todėl vartotojui, nesusipažinusiam su konkrečios reliacinės duomenų bazės struktūra ir turiniu, būtina užtikrinti paieškos organizavimą, raktinių žodžių pagrindu.

Šiuolaikinės duomenų bazių sisteminiuose duomenyse yra saugomi visų lentelių ir jų stulpelių pavadinimai [12]. Šios informacijos ir raktinių žodžių rinkinio, kurį užduoda vartotojas, pagrindu pakankamai paprasta sukonstruoti SQL užklausas, kurios iš duomenų bazės išrenka lentelių eilutes (kortežus), kurių bent viename stulpelyje yra raktinis žodis. Rezultate vartotojui pateikiamas įvairaus formato lentelių rinkinys, kurių stulpelių vardai atitiks duomenų bazės stulpelių vardus. Priklausomai nuo reliacinės duomenų bazės architektūros ir realizacijos, toks atvaizdavimas turės sekančias ypatybes:

- kiekvienos rezultato lentelės atitikimas užklausai bus nedidelis, o normalizuotoje duomenų bazėje raktiniai žodžiai bus išbarstyti atskiruose stulpeliuose;
- lentelių ir jų stulpelių vardai vartotojui yra labai mažai informatyvūs, nes dažniausiai reliacinėse duomenų bazėse stulpelių vardai yra programinių objektų vardai.

Vartotojui tokia informacija yra labai mažai naudinga.

Reliacinėse duomenų bazėse atskiros lentelės neturi semantinio pilnumo. Semantiškai pilni objektai (pradiniai ir rezultato dokumentai) formuojami iš kelių lentelių kortežų, tarpusavyje sujungtų bendrais raktais. Rezultato pilnumas gali būti padidintas, jei paieška bus vykdoma ne atskirose reliacinės duomenų bazės lentelėse, bet tam tikruose dokumentų rinkiniuose. Tai lemia, kad duomenų bazės aprašas turi būti saugomas išplėstinių sąvokų, kurios būtų aiškiai suprantamos vartotojui, pavidalu. Tokių, orientuotų į vartotoją, sąvokų rinkiniai leidžia išplėsti raktinių žodžių paieškos sritį, nes vienoje schemoje ieškoma leksema gali būti reikšmėse, kitose – pavadinimuose.

Yra dar kitos problemos, susijusios su informacijos paieška reliacinėse duomenų bazėse. Viena iš jų yra susijusi su informacijos vaizdavimo abstrakcijos lygiu [13]. Pavyzdžiui, vienoje duomenų bazėje pageidaujama informacija yra saugojama lentelėje „automobilio detalės“, o kitoje – atskirose lentelėse „ratai“ ir „pavaros“. Skirtis gali ir įvairių vartotojų užduodamų raktinių žodžių rinkinio, kaip užklausos kriterijaus, abstrakcijos lygis. Kita problema yra susijusi su duomenų bazės kūrėjų ir vartotojų terminologijos skirtumais [13]. Be sinonimiškumo, terminologijos skirtumai gali pasireikšti naudojant įvairius sutrumpinimus, kurie duomenų bazėse naudojami dažniau, nei vientisuose tekstuose.

Šios problemos reliacinėse duomenų bazėse yra susijusios su informacijos saugojimo būdais jose. Normalizacijos taisyklės ir dirbtinių raktų panaudojimo praktika sukelia tai, kad aiški savybės reikšmė yra saugoma tik vieną kartą, o jos pasikartojimui yra naudojamos nuorodos. Tekstiniuose dokumentuose toks atributas yra aiškiai išreikštas, tačiau jo pasikartojimui dažnai naudojami sinonimai arba įvardžiai. Tai daro labai sudėtingą paiešką tokiuose tekstiniuose dokumentuose.

Pateiktos problemos smarkiai sumažina rezultatų informatyvumą ir atitikimą vartotojo užklausoms. Norint minimizuoti šias problemas, paieškos sistemoms reikalinga prieiga prie papildomų informacinių resursų [14].

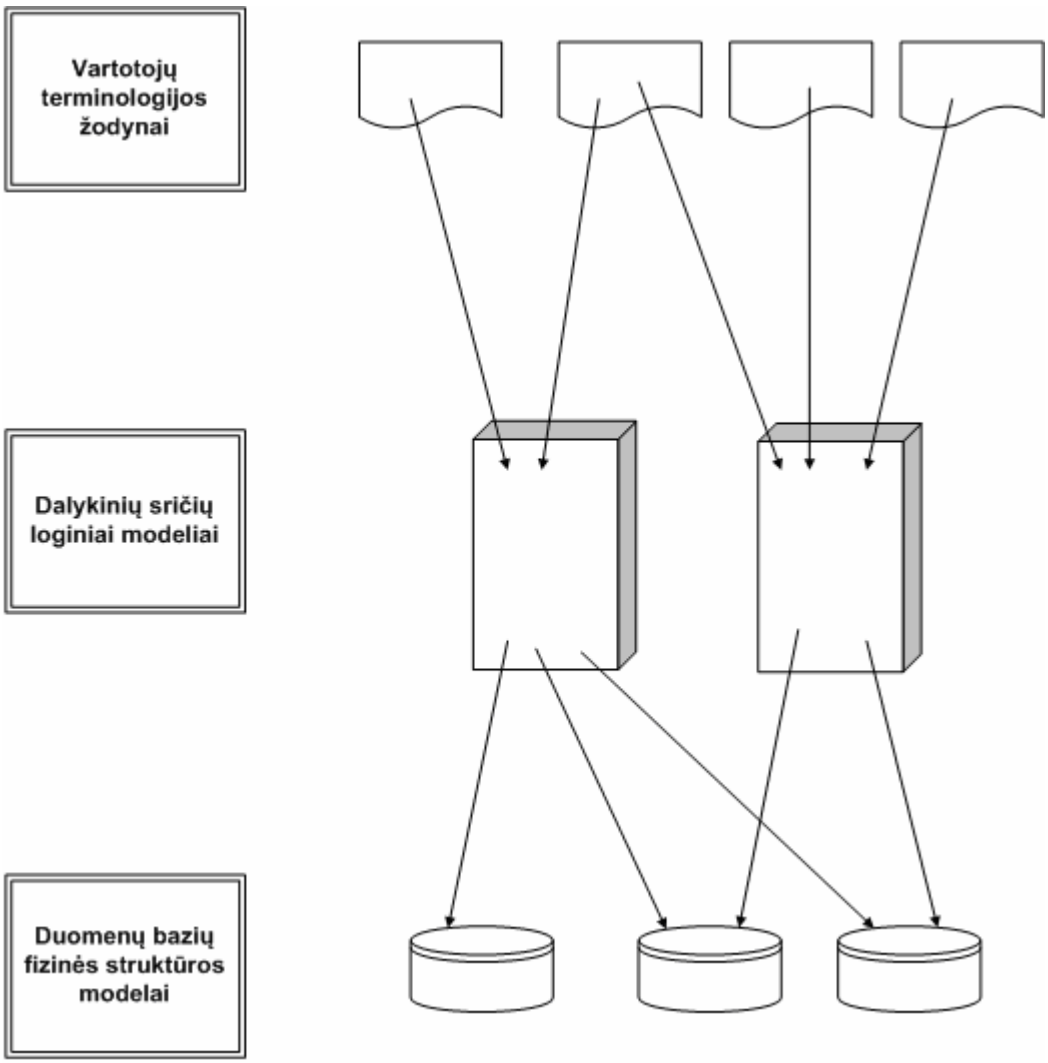
Kaip jau minėjome, efektyviai paieškai pagal raktinius žodžius reliacinėje duomenų bazėje trūksta kelių tipų informacijos. Visų pirma, tai informacija apie pačios duomenų bazės struktūrą. Duomenų bazės atributų surištumas ir artumas nustatomas kortežų surištumu pagal raktinius žodžius. Tokiu būdu, kad vartotojui būtų pateikta prasminga informacija paieškos sistema turi turėti informacija apie galimus kortežų sujungimus pagal raktus. Tokią informaciją yra lengva pasiekti betarpiškai iš duomenų bazės schemos, kadangi šiuolaikinėse duomenų bazių valdymo sistemose informacija apie raktus saugoma sisteminiuose duomenyse.

Kita informacija, tai semantiniai žymekliai [23], skirti vienodai pažymėti visus duomenų bazės atributus. Tokiam žymėjimui patogu naudoti dalykinės srities loginę schemą su ryšiais „klasė-poklasis-egzempliorius“. Toks būdas žinomas kaip Semantic Web ir šiuo metu plačiai taikomas dokumentų paieškos sistemose. Reliacinės duomenų bazės praktiškai jau yra sužymėtos atributų vardais ir tokiu būdu yra labiausiai paruoštos bendram su Semantic Web technologijų panaudojimui, nes reikia sukurti atributo vardo ir tam tikro semantinio vieneto atitikmenį.

Taip pat paieškos mechanizme turi būti leksinė-semantinė informacija apie terminologiją, kurią naudoja skirtingos vartotojų grupės, pažymėti vienus ar kitus objektus, savybes ir ryšius dalykinėje srityje (vartotojų žodynai). Tokia informacija leidžia išspręsti sinonimijos, sutrumpinimų ir abreviatūrų[3], kurias vartoja skirtingos vartotojų grupės.

Papildomos informacijos struktūra pateikta pav. 1.

[3] Sutrumpinimas, sudarytas iš sudėtinio pavadinimo žodžių pirmųjų raidžių. Pavyzdžiui, Lietuvos Respublikos Seimas – LRS.



1 pav. Papildomos informacijos, reikalingos efektyviai paieškai, struktūra

Reikia pažymėti, kad ši informacija gali būti pateikta daugelyje egzempliorių. Paieškos sistema gali būti susieta su daugeliu duomenų bazių, šios bazės gali būti susietos su daugeliu dalykinių sričių arba būti tarpdisciplininės, skirtingos vartotojų grupės gali naudoti skirtingas terminologinius žodynus, net daugiakalbius.

1.2. Ontologijos ir jų taikymo galimybės

Iškyla klausimas apie tokios įvairiapusiškos informacijos pateikimo formalizmą. Kiekvienas iš šių modelių turi nusistovėjusį vaizdavimo formalizmą. Duomenų bazės schemos tradiciškai vaizduojamos naudojant SQL DDL, loginiai modeliai – ER-diagramomis[4], semantinė informacija – tezaurusais. Akivaizdu, kad tokių įvairiapusiškų modelių naudojimas techniškai yra labai sudėtingas. Paieškos sistemoms pakanka galingo formalizmo, orientuoto į automatinį informacijos, saugomos modeliuose, apdorojimą, panaudojimo. Nagrinėjamam atvejui galima pasiūlyti ontologijų formalizmą. Ontologijų daliniais atvejais yra ir loginiai ir fiziniai duomenų modeliai, o taip pat ir žodynai [15].

Naudojant ontologijas, atspindinčias skirtingus žinių vaizdavimo aspektus, patogu naudoti ir vieningą jų aprašymo kalbą. Tokia kalba gali būti OWL kalbos, sudarytos Sematic Web Activity grupės, poaibis OWL-DL (Description Logic). Tokį pasirinkimą lemia praktiniai aspektai, nes šią kalbą palaiko daugelis šiuo metu egzistuojančių žinių aprašymo bei loginio programavimo sistemų, o taip pat šios kalbos perspektyvomis tapti tarptautiniu standartu.

Bendru atveju, informacinėse sistemose, ontologija – tai duomenų modelis, turintis dvi specifines savybes:

- Ontologijos sudaromos remiantis bendru tam tikros grupės dalykinės srities supratimu. Šis supratimas atvaizduojamas ekspertų susitarimu dėl sąvokų ir jų tarpusavio ryšių dalykinėje srityje.
- Ontologijos naudoja tokį vaizdavimo būdą, kuris gali būti apdorojamas kompiuterinėmis programomis.

Būtent šios savybės leidžia padidinti informacinių paieškos sistemų intelektinį lygį, įvedant į jas žinias apie dalykinę sritį ontologijų pavidalu [17]. Dalykinė sritis atvaizduota ontologijomis gali būti panaudota trijų pagrindinių žinių valdymo procesų valdymui: komunikacija, integracija ir loginės išvados. Šie procesai remiasi vienas kitu ir atspindi augančius formalizacijos lygius nuo paprasčiausių žodynų iki formalių loginių struktūrų, kurios leidžia atlikti sudėtingas logines išvadas. Ontologijų panaudojimas komunikacijai palengvina bendravimą, atvaizduodamos bendrai naudojamas sąvokas, kurios gali būti panaudojamos teiginių apie probleminę sritį formulavimui ir užrašymui (užklausų, atsakymų). Tokių pranešimų gavėjams, ontologija leidžia suprasti pranešimą ir išvengti dviprasmybės, pateikdama kontekstą interpretavimui. Bendru atveju, visų priimtos, gerai suprantamos kalbos panaudojimas leidžia organizacijoms geriau mokytis ir greičiau reaguoti į išorinės aplinkos pokyčius. Esant suderintam taikymui, bendrai naudojama ir formaliai pripažinta

[4] Standartine ER diagrama iš dalies galima pavaizduoti verslo taisyklių rūšis – faktus bei apribojimus (pvz.: vienas užsakymas turi tik vieną užsakovą)

kalba skatina , kad darbo grupės (pavyzdžiui, įvairios darbo su klientais grupės) gali dirbti labiau susiderintai, didindamos viena kitos supratimą [18].

Integracija reiškia įvairių tipų ir paskirstytų informacijos ir žinių resursų bendrą naudojimą. Ontologijos yra daugiau nei paprasti tiksliai apibrėžtų sąvokų žodynai: jų pagrindinis privalumas yra galimybė aprašyti ryšius tarp probleminės srities esybių. Bet kuris žodynas yra prasmės saugykla, bet jis nustato ryšius tik pagal jų santykį su kitais žodžiais. Realybėje, informacinis vienetas apsprendžiamas tuo, su kokiais kitais elementais ir kaip jis susijęs. Santykiai (ryšiai) tarp sąvokų žymiai padidina navigacijos ir paieškos dalykinėje sistemoje būdų skaičių, atlikti žinių analizę, klasifikavimą ir vizualizavimą. Tam, kad panaudoti tokią sąvokų įvairovę, skirtingi informaciniai ir žinių resursai turi būti sužymėti ontologijų pagalba. Šis procesas vadinamas semantine normalizacija. Terminas resursas čia naudojamas plačiąja prasme: duomenų bazės, dokumentai, vartotojų kompetencija, interneto resursai ir t.t. Resursų įvairovė apsunkina tarpusavio ryšių nustatymą („žinios apie tai, ką mes žinome“). Integracijos lygis yra virš komunikacijos lygio. Šiuo atveju komunikacija vyksta tarp ontologijos ir metaduomenų, bei vartotojo, kuris atlieka informacijos paiešką. Daugeliu atveju ši komunikacija vykdoma kompiuterinės sistemos pagalba, kuri leidžia vartotojams formuluoti užklausas, siūlydama alternatyvinius terminus, apibendrinimus, tikslinimus it t.t., priklausomai nuo to, kaip ši sistema supranta ontologijas.

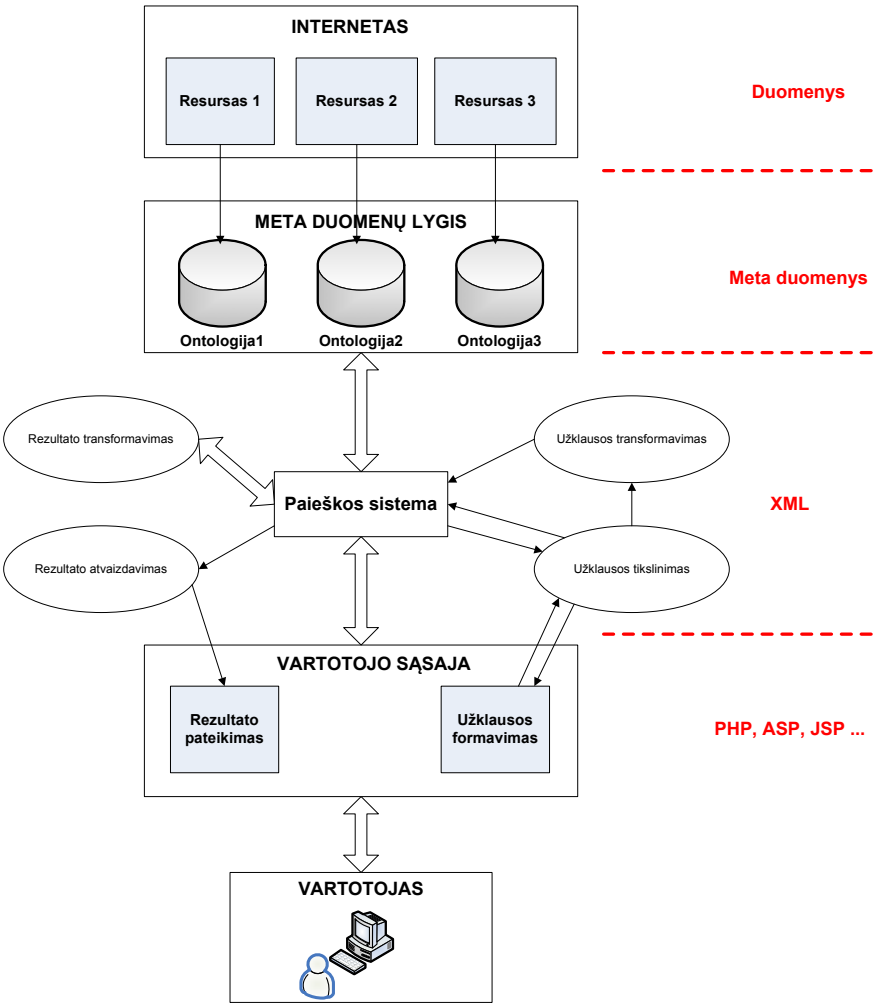
Loginės išvados formavimas yra vienas iš sunkiausių ontologijų panaudojimo variantų. Jei ontologijų panaudojimas komunikacijai ir integracijai remiasi žiniomis apie tai, kokie yra dalykinės srities objektai ir kaip jie susiję tarpusavyje, šis panaudojimo variantas reikalaija žinių kodėl vieni ar kiti objektai yra susiję tarpusavyje.

1.3. Ontologijų taikymas paieškai internete

Kadangi kreipdamasis į paieškos sistemą vartotojui reikalinga galimybė atsakymui gauti visus resursus, atitinkančius užklausos prasmę, paieška turi būti semantiškai orientuota. Tokiai paieškai organizuoti yra siūloma taikyti ontologijas, kuriose yra resursų semantikos aprašymai.

Interneto resursų semantika yra labai įvairi, todėl, informacijos paieška yra tuo paprastesnė, kuo siauresnė dalykinė sritis. Dėl to, praktikoje galima apsiriboti vienos konkrečios dalykinės srities ontologijos aprašymu [20].

Kaip jau minėjome, ontologijai sudaryti reikalinga tiksliai organizuotų konstrukcijų deklaratyvus vaizdavimas, kuriame yra teminės srities terminų žodynas, šių terminų apibrėžimų aprašymas, egzistuojantys ryšiai tarp jų, jų teoriškai įmanomi ir neįmanomi tarpusavio ryšiai. Tokiu būdu aprašytos ontologijos yra tarpininkas tarp paieškos sistemos ir duomenų, saugojamų paskirstytuose resursuose (pav. 2).



2 pav. Informacijos paieškos organizavimas ontologijų pagalba

Sąveika su ontologija numatoma sekančiuose etapuose:

- paieškos sąlygas atitinkančio dokumento formavimas;
- užklausos paieškos sistemai sudarymas ir tikslinimas;
- užklausą atitinkančių dokumentų sąrašo formavimas.

Problema yra paiešką padaryti labiau dinamišką ir patogią vartotojui. Bet kuriam užklausos, kylančios žmogaus praktinėje veikloje, tipui turi būti atrastos atitinkamos žinios informacinėje interneto erdvėje. Tuo pačiu užklausų kalba neturi būti sudėtinga. Iš esmės, vartotojo bendravimą su paieškos sistema galima padaryti paprastesnę, priartinus užklausų kalbą prie natūralios kalbos.

Tokiame paieškos organizavime, vartotojo užklausą atitinkančio dokumento formavimo etape, iš vartotojo užklausos išskiriamos prasminės struktūros: reikšminiai žodžiai ir probleminės srities terminai. Šios prasminės struktūros vėliau naudojamos paieškos vaizdinio formavimui. Užklausą atitinkančio dokumento vaizdinys yra paieškos sistemos veikimo pageidaujamo rezultato aprašymas, kuriame yra:

- terminų, kurie turi būti dokumente, rinkinys;
- dokumento charakteristikų rinkinys;
- reikalavimų paieškos sistemai, tokių kaip dokumentų skaičius ir pan., rinkinys.

Ontologijos pagalba transformuota pradinė užklausa pateikiama vartotojui patikslinimui. Po to išplėsta ir vartotojo patikslinta užklausa automatiškai modifikuojama į užklausą duomenims. Čia paieškos sistema užduoda paieškos parametrus, būdingus kiekvienai paieškos sistemai.

Tokios paieškos sistemos darbo rezultatas yra nuorodų į HTML[5]-dokumentus rinkinys. Kadangi tarp šių nuorodų gali būti resursai, neturintys nieko bendro su vartotojo užklausa, sekančiame etape vykdomas paieškos sistemos rezultatų patikrinimas paieškos užklausai. Rezultato HTML-dokumentų analizė turi remtis raktinių žodžių išskyrimu iš tokių dokumento dalių kaip antraštės, nuorodos ir dokumento turinio. Kadangi paieškos sistemos darbo rezultatas yra labai didelis nuorodų kiekis, tai dokumentų analizė turi būti vykdoma naudojant griežtas atrankos taisykles. Po atlikto dokumentų filtravimo, paieškos rezultatai pateikiami vartotojui patogia forma [14].

Tokios paieškos organizavimo schemos, panaudojant ontologijas, realizacija numato sekančius etapus:

1. Užklausos natūralia kalba specifikavimas;
2. Ontologijos sudarymas;
3. Paieškos rezultatų analizės priemonių pasirinkimas;
4. Sąsajos tarp vartotojo ir ontologijos sukūrimas; ši sąsaja turi užtikrinti užklausos formavimą natūralia kalba, užklausos patikslinimą, paieškos rezultatų paiešką;
5. Ontologijos tarpusavio ryšio su paieškos sistema modulio sudarymas; šis modulis turi transformuoti užklausą, sudarytą ontologijos, į atitinkamą paieškos sistemos užklausos formatą ir paieškos rezultatų pateikimą vartotojui.

Kaip matome, didelę svarbą tokioje schemoje užima vartotojo sąsajos, kurios tikslai yra užklausos įvedimas, užklausos transformavimas, užklausos tikslinimas ir rezultatų atvaizdavimas. Čia didelį vaidmenį vaidina užklausos tikslinimas. Toks tikslinimas gali būti atliekamas dviem būdais:

- užklausos tikslinimas pagal jos vykdymo tarpinių rezultatų analizę;
- užklausos tikslinimas pagal ontologiją, t.t. ne pagal konkrečius duomenis, bet pagal duomenų semantinį aprašą.

[5] *Hyper text Markup Language* - Hiperteksto žymėjimo kalba. Tai kompiuterinė žymėjimo kalba, naudojama pateikti turinį internete.

2. Analitinė dalis

2.1. XML dokumentai

Dėl savo nepriklausomumo nuo DBVS ir platformos tipo, XML tapo ypatingai naudingas atviroms susisiekiančioms sistemoms, kuomet duomenys apsikeičiami tarp heterogeninių duomenų šaltinių. Atsiradus XML užklausų kalboms tokioms kaip XML-QL, XQL, Quilt, ir kt., tapo įmanoma modeliuoti duomenų srautus nedetalizuojant juos realizuojančios DBVS.

XML (*angl.* Extensible Markup Language) – išplečiama žymių kalba buvo pradėta kurti apie 1996-uosius metus ir pagrindine priežastimi to tapo, kad jos pirmtakė, SGML yra pernelyg sudėtinga, kad būtų laisvai naudojama informacijos pasikeitimui Pasaulinio tinklo susietose informacinėse sistemose. Pirmos kalbos versijos standartizavimo procesas užtruko kelis metus ir buvo baigtas 1998, paskelbus W3C konsorciumo rekomendaciją XML kalbai.

XML yra medžio tipo struktūra paremta duomenų aprašymo kalba. Medžio tipo struktūra yra dažnai naudojama programinėse priemonėse ir tinka aprašyti daugumą (ne visus) duomenų tipų. Esminiai XML struktūros elementai yra Elementai, kurie gali būti tiek su turiniu (t.y. kitais elementais) (*angl. c*ontainer element), tiek be (*angl.* node element). Elementai taip gali turėti viena ar daugiau atributų kurie susiejami su tam tikru tipu bei jiems priskiriama reikšme. Elementų turinys gali būti tiek kiti elementai, tiek tam tikro tipo simboliškai išreikšta reikšmė.

XML dokumentas yra tekstinis dokumentas kurio bazinis elementas – simbolis yra Unicode standarto simbolis. Tuo XML skiriasi nuo kitų nestandartizuotų formatų, kurie daugiausia yra dvejetainiai failai, perskaitomi ir išsaugomi specialiomis programomis. Tuo tarp XML dokumentai gali būti sudaromi ir keičiami standartiniais tekstiniais redaktoriais.

Tai yra išplečiama kalba, t.y. priešingai nei HTML, kur kiekvienam simboliui yra suteikta nustatyta ir nekeičiama prasmė bei sintaksė, XML dokumento elementai gali turėti taikomajai sričiai reikalingus pavadinimus, kas atitinkamai suteikia papildomą meta-informaciją apie dokumento turinį, be to tokia struktūra lengvai perprantama žmogaus akimi.

2.1.1. Privalumai bei trūkumai

Šiame poskyryje išvardinta pagrindinės XML savybės ir jų interpretacija kaip teigiamų ir neigiamų savybių, priklausomai nuo konteksto.

XML formatas yra patogus dėl daugelio priežasčių, svarbesnės iš jų:

- formatas yra lengvai perskaitomas tiek žmogaus tiek skaičiavimo mašinoms;
- naudoja Unicode tarptautinį simbolių formatą, todėl pritaikomas praktiškai visų kalbų dokumentams;
- gali išreikšti pagrindines informatikoje naudojamas struktūras: sąrašus, įrašus, medžius;
- pats save aprašantis dokumentas, nurodantis reikšmių bei elementų prasmę;
- griežtai apibrėžtas todėl gali būti perskaitomas ir patikrinama automatizuotomis programinėmis priemonėmis;
- pagrindinė išraiška yra tekstinis failas, todėl prieinamas visose sistemose ir atsparus technologijų pasikeitimams;
- yra kilęs ir dalinai suderinamas su SGML, todėl jau yra daug jį palaikančių taikomųjų programų;

Žemiau pateikiami XML panaudojimo trūkumai (dažniausiai pasireiškiantys tam tikrose panaudojimo srityse):

- XML formato išraiška turi daug perteklinių elementų ir todėl gali būti sunkiai perskaitoma ir reikalauja daugiau vietos talpyklose. Tai iš dalies gali būti atsveriama panaudojus duomenų kompresiją, tačiau tai nėra visada įmanoma;
- rekursyvi struktūra reikalauja daug skaičiavimų jos patikrinimo ir perskaitymo metu, kas gali sudaryti didesnį apkrovimą skaičiuojamam elementui negu pats dokumento turinio apdorojimo procesas;
- kai kurie laiko sintaksę pernelyg sudėtinga ar paveldinčią pernelyg daug nereikalingų savybių iš pirmtakės SGML;
- dažniausiai nuskaitymo procese negalima nustatyti perskaitomos informacijos įrašų tipo neturint dokumentą aprašančios duomenų schemos (DTD ar XML Schemos);
- sudėtinga išreikšti nehierarchiškas duomenų struktūras;
- objektinių ar reliacinių duomenų perteikimas į XML dokumentą gali būti neefektyvus, sukeliantis pašalinių efektų, anomalijų;

2.1.2. XML dokumento tikrinimas

Dėl savo tekstinės prigimties ir medžio tipo struktūros prieš duomenis interpretuojant dažnai yra reikalinga formato patikra (angl. *validation*). XML dokumento tikrinimas gali būti įgyvendinamas dviem lygiais:

A) Ar dokumentas atitinka XML sudarymo taisykles.

Šią taisyklę tenkinantis dokumentas atitinka XML sintaksės taisykles, t.y. jis yra hierarchiškas dokumentas, kiekvienas elementas turi atidarymo ir uždarymo žymes (angl. *tag*) ir kt. Jeigu dokumentas nepatenkina šio lygio tikrinimo jis laikomas netinkamu ir nėra tikrinimas sekančio lygio (B) taisyklėmis.

B) Ar dokumentas atitinka vartotojo duomenų išraiškos (sintaksės) taisykles.

Jei XML dokumentas atitinka vartotojo nurodytas dokumento sudarymo taisykles jis laikomas tinkamu (angl. *valid*). Vartotojo taisyklių pvz.: nurodant asmens duomenų įrašą, jis turi turėti vardo bei pavardės vidinius elementus, saugančius po teksto tipo eilutę. Šio tipo patikrinimas dažnai yra atliekamas XML tikrintojais (angl. *validator*) kurie tikriną dokumentą pagal jo tipo aprašą (schemą).

XML dokumentų tipams apibrėžti žinomiausi yra šios schemų tipai:

- DTD (*Document Type Definition*). Tai anksčiausiai sukurtas duomenų aprašo formatas, originaliai skirtas SGML dokumentų tikrinimui, tačiau taip pat tinkantis ir XML, tačiau nepalaiko kai kurių XML ypatybių.
- XML schema (XML Schema). Tai DTD analogas, skirtas išskirtinai XML standarto dokumentams apibrėžti.
- RELAX NG. Ši schema turi du variantus – XML ir ne XML dokumentams apibrėžti, kompaktiškesnis variantas. Palyginus su XML Schema, siūlo lengviau realizuojamą tikrinimo būdą.

2.1.3. Dokumento ir jo konceptualaus medžio pavyzdys

Čia pateikiamas ir detalizuojamas XML dokumento pavyzdys nurodant pagrindinius jo elementus.

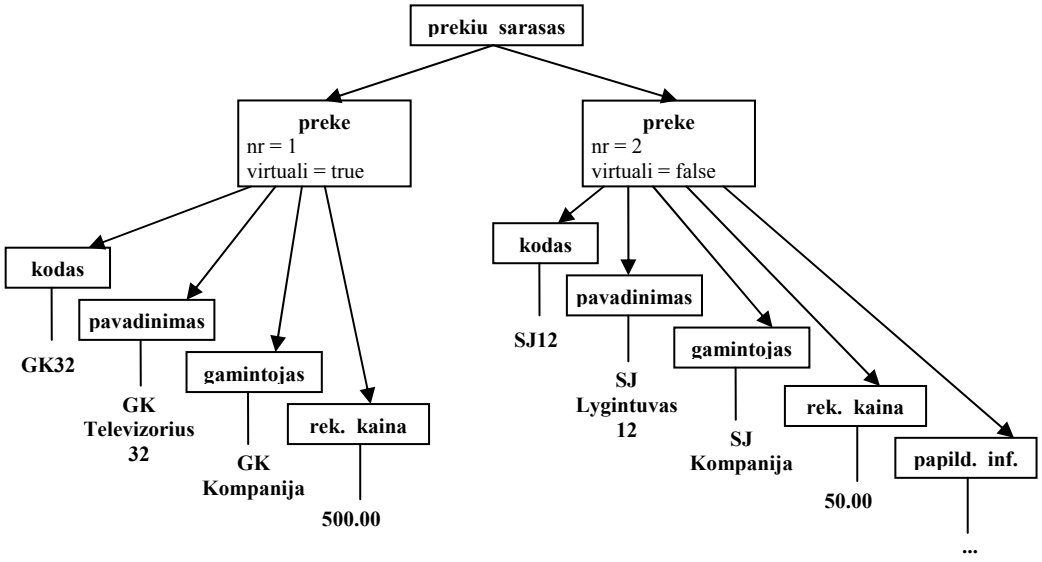
```
<?xml version="1.0" encoding="utf-8"?>
<prekiu_sarasas>
  <preke nr="1" virtuali="true">
        <kodas>GK32</kodas>
        <pavadinimas>GK Televizorius 32</pavadinimas>
        <gamintojas>GK Kompanija</gamintojas>
        <rekomenduojama_kaina>500.00</rekomenduojama_kaina>
  </preke>
  <preke nr="2" virtuali="false">
        <kodas>SJ12</kodas>
        <pavadinimas>SJ Lygintuvas 12</pavadinimas>
        <gamintojas>SJ Kompanija</gamintojas>
        <rekomenduojama_kaina>50.00</rekomenduojama_kaina>
        <papildoma_informacija>Rekomenduojama tik vidurio Lietuvos
regionui.</papildoma_informacija>
  </preke>
</prekiu_sarasas>
```

XML dokumentas visuomet prasideda įžangine sekcija `<?xml .. ?>` kurioje nurodomas dokumento tipas (XML), XML versija bei dokumento koduotė.

Dokumentas yra surūšiuotas žymėtas medis, kurio išraiška pasižymi šiomis savybėmis:

- dokumentas susideda iš mazgų, kuriuose saugomi tekstiniai duomenys, mazgai savyje gali saugoti kitus mazgus;
- XML dokumente gali būti tik vienas šakninis (pradinis) mazgas – nagrinėjamame pavyzdyje – tai `<prekiu_sarasas>`;
- mazgai turi sau priskirtą vardą bei atributus, kurie atitinkamai susideda iš vardo bei reikšmės.

Aukščiau pateikto dokumento mazgų medis gali būti pavaizduojamas grafiškai:



3 pav. XML dokumento konceptualus medis.

2.2. XML Schemos

XML Schema yra vienas iš XML dokumentus apibrėžiančių schemų tipų. Jo pirmtakas buvo DTD, tačiau ilgainiui jo galimybių neužteko. Išleistas kaip W3C rekomendacija 2001 šiuo metu jis tampa populiariausiu XML schemos formatu. Kūrimo metu stengtasi perimti geriausias DTD ir ankstyvųjų XML schemų formatų (DDML, SOX, XML-Data) savybes, ir šiuo metu XML Schema pripažinta didžiausių programinės įrangos tiekėjų kaip pagrindinė XML schema.

XML Schema, kaip formatas apibrėžia dokumento struktūra aukštesniu lygiu nei XML kalba, kuri užtikrina tik dokumento atitikima XML formatui. XML schema suteikia XML dokumento elementams apibrėžtą sintaksę bei prasmę: apibrėžtą elementų hierarchiją, atributų aibę, skaitinius, tekstinius ir kt. apribojimus. Jeigu XML dokumentas atitinka programos nurodytą XML schemą, jis galima būti priimamas ir interpretuojamas taikomosios programos. XML Schemos dokumentas tuo pačiu metu yra XML dokumentas, t.y. XML schemos dokumento formatas taip pat gali būti apibrėžiamas atitinkama XML Schema.

2.2.1. Savybės. Privalumai, trūkumai.

XML schemos turi daug savybių, kurios palengvina sprendimų kūrimo procesą, tačiau tuo pačiu metu jį ir apsunkina. Skyriuje yra nagrinėjamos šios savybės.

Esminė schemos paskirtis yra nustatyti duomenų formato susitarimą tarp duomenų šaltinio ir priėmėjo. Tokiu būdu dokumentai gali būti patikrinami prieš jų panaudojimą. Kadangi dažniausiai realūs perduodami tarp informacinių sistemų duomenys turi sudėtingą, tarpusavyje susijusią struktūrą, šios struktūros apibrėžimas taip pat reikalauja daug pastangų. Todėl XML Schema gali pasirodyti pernelyg sudėtinga savo paskirčiai, tačiau to reikalauja jos detalumas ir plati galimybių aibė.

Esminiai privalumai, gaunami naudojant XML Schemą:

- išskiriami ir tiksliai apibrėžiami informacinės sistemos duomenų formatai, paskirtis, tiek išsiunčiami tiek priimami duomenys gali būti patikrinami;
- trivialūs, žmogaus parašyti dokumentai gali būti integruojami į informacinę sistemą nes yra galimybė juos patikrinti esančius teisingo formato;
- centralizuotai saugomas duomenų apibrėžimas leidžia sukurti nepriklausomas programines priemones jų apdorojimui;

Tačiau specifinių XML Schemos savybių, jos panaudojimas pasireiškia šiais trūkumais:

- XML Schema yra sąlyginai sudėtinga, ir reikalauja daug pastangų, kad būtų panaudojama, juo labiau efektyviai;
- XML Schemos, nors ir tiksliai apibrėžiančios dokumento gramatiką ir semantiką nenurodo tikslaus dokumento laukų panaudojimo ir/ar interpretavimo;
- Jos panaudojimas reikalauja gan daug techninių resursų ir projektuotojų pastangų. XML Schemų tikrintojai (angl. *validator)* ir nagrinėtojai (angl. *parser*) privalo būti pilnai realizuoti, kad galima būtų nurodyti dokumento atitikimą schemai. Ši programinė įranga yra gana sudėtinga ir vykdoma sąlyginai lėtai.

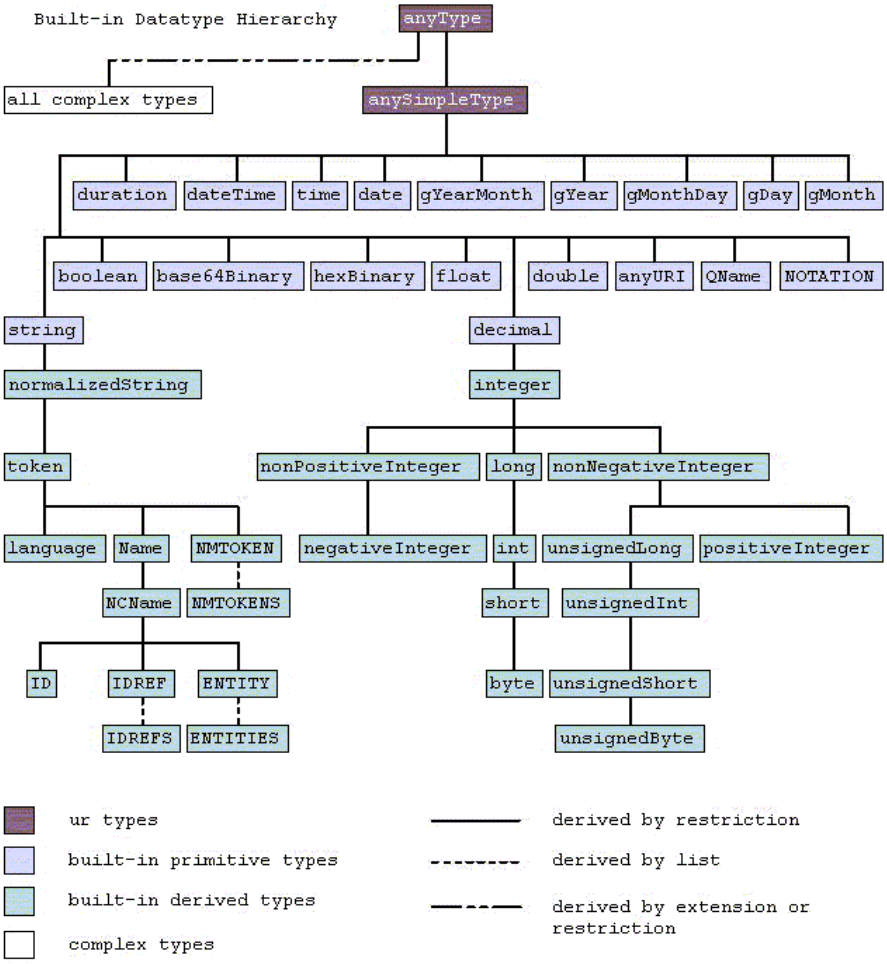
2.2.2. Struktūra

XML Schema turi specifinę struktūrą, kurią bus stengiamasi apžvelgti šiame skyriuje.

Pagrindinės schemos struktūrinės dalys yra duomenų, elementų, atributų tipų, vardų erdvių. XML Schemos gali būti panaudotos viena kitos aprašui, kas leidžia lengvai išplėsti ir sukurti naujus schemų tipus keičiantis programinei įrangai (PĮ), išvengiant suderinamumo problemų, pagerinamas PĮ moduliarizmas.

Duomenų tipai sudaro nepersidengiančią hierarchiją iš jau sudarytų, iš anksto nustatytų XML Schemos ir vartotojo aprašytų duomenų tipų ir skirstomi į sudėtingus (angl. *complex*) bei paprastuosius (angl. *simple*):

- paprastieji tipai yra aprašomi `simpleType` elementu ir sudaromi patikslinant ar išplečiant jau egzistuojantį paprastąjį tipą. XML Schema apibrėžia pagrindinius duomenų tipus kaip sveikasis skaičius (angl. *integer*), teksto eilutė (angl. *string*) ir kt.;
- sudėtingieji tipai, priešingai nei paprastieji gali turėti vidinius elementus bei priskirtus atributus. Sudėtingieji apibrėžiami naudojant XML schemos elementą `complexType`, ir sukuriami išplečiant jau egzistuojantį sudėtingąjį tipą;
- duomenų tipai gali būti neįvardinti, jeigu naudojami tik vieno XML schemos elemento apibrėžimui, kas leidžia sutrumpinti ir supaprastinti XML schemos aprašą;



4 pav. XML Schemos apibrėžtų duomenų tipų hierarchija

Elementai yra apibrėžiami nurodant jų turinio duomenų tipą, kuriame atitinkamai nurodyta kokius atributus elementas gali turėti, kokie galimi vidiniai jo elementai, ar galimas mišrus turinys (angl. *mixed content*), t.y. sumaišytas simbolinis-tekstinis bei elementų turinys – analogas HTML `<body>` elementas. Elementas taip pat gali neturėti vidinio turinio, tik atributus; – tuomet jis laikomas tuščiuoju elementu, analogas – HTML `<br>` elementas.

Atributai apibrėžia XML elemento leidžiamus priskirtu atributus. Atributams gali būti suteikiami tik paprastieji tipai, kadangi atributai negali turėti viduje XML elementų. Kaip ir elementai atributai gali būti apribojami pagal jų būtinybę elemento apraše, sintaksę ir pan. Atributams, kaip ir elementams gali būti priskirti įvairūs apribojimai, tokie kaip pasikartojimo (angl. *occurence*).

Vardų erdvės (angl. *namespace*) yra XML Schemos elementas apribojantis konkretaus XML Schemos dalies matomumą tik tam tikroje vardų erdvėje. Tai leidžia modularizuoti XML Schemų aprašus, turint omenyje kad XML Schemos gali būti panaudojamos viena kitos aprašui sukurti.

XML Schemoje yra nustatyti dokumentacijos elementai `<annotation>` ir `<documentation>` kurie leidžia dokumentuoti paprasta kalba schemos struktūrą, todėl atkrenta būtinybė saugoti atskirus konkrečios XML schemos aprašo dokumentus.

2.3. XML duomenų bazės, užklausų kalbos

Nagrinėjant XML duomenų integraciją natūraliai yra reikalinga apžvelgti dabartines priemones XML duomenų analizei.

XML duomenų bazės (angl. Native XML Database) yra naujas konceptas, kuris gali būti apgaulingas, nes tokios duomenų bazės nesaugo XML natūralioje formoje (tekstu), jos netgi gali nebūti nepriklausoma PĮ. Todėl pateikiamas toks jų apibrėžimas:

- XML DB apibrėžia loginį XML dokumento modelį, kuris gali būti DOM, XML Schema ir kt.;
- XML dokumentas tokioje DB yra fundamentalus informacijos vienetas, taip kaip lentelės eilutė reliacinėje duomenų bazėje;
- nebūtina kad tokia DB turėtų kažkokį konkretų fizinį duomenų saugojimo modelį, todėl gali būti sukurta ant egzistuojančios reliacinės, hierarchinės ar objektinės duomenų bazės, arba naudoti savo unikalią duomenų saugojimo sistemą.

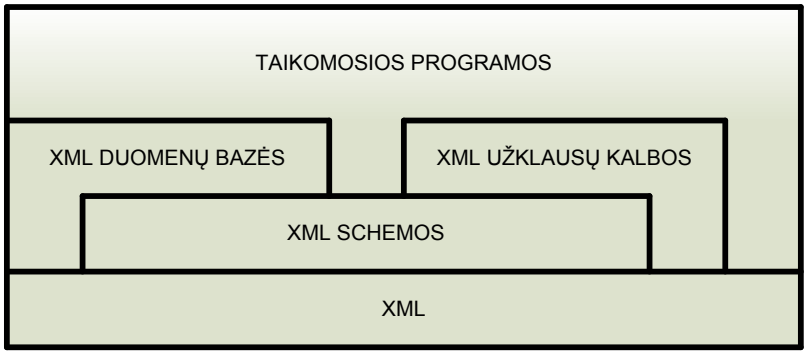
Kadangi XML duomenys tam tikrais aspektais stipriai skiriasi nuo reliacinių ar objektiškai orientuotų duomenų, tokių duomenų apdorojimui negalime naudoti įprastas SQL (Structured Query Language) ar OQL (Object Query Language) kalbas. Į pagalbą modeliuojant šių duomenų apdorojimą galima pasitelkti XML užklausų kalbas. Pastaruoju metu atsiradę gan daug kalbų, realizuojančių užklausas iš XML dokumentų:

1) XML-QL. Ši užklausų kalba turi SELECT/WHERE konstrukcijas, primenančias SQL.
2) XQuery. Lanksti užklausų kalba, galinti tvarkyti ne tik struktūrizuotus duomenis (kaip XML), bet ir reliacines bei objektines duomenų bazes.
3) XQL. Dar viena kalba, orientuota tik į XML, remiasi XPath ir XSLT specifikacijomis.
4) Quilt. Užklausų kalba, perimanti konstrukcijas iš SQL ir OQL, pritaikyta vien XML dokumentams.

XML dokumentų schemos yra nepriklausomos nuo platformos bei DBVS, todėl suteikia galimybę modeliuoti informacinę sistemą jos dar fiziškai nerealizavus, o tik nustačius duomenų mainus, kuriais galima operuoti aukščiau paminėtomis kalbomis.

2.4. Apibendrinimas

Apžvelgus aukščiau aprašytas technologijas galima nurodyti pagrindinių XML technologijų infrastruktūrą, kuria galima remtis kuriant naujas taikomąsias sistemas.



5 pav. Pagrindinių XML technologijų infrastruktūros schema.

2.5. XML ir paieška paskirstytose duomenų bazėse

XML naudojama dokumentų, turinčių struktūrizuotą informaciją, aprašymui. Tokia informacija turi turėti ir turinį (aprašymus, paveikslelius ir kt.), taip pat vieną ar kelis požymius, rodančius, kokį vaidmenį šis turinys atlieka. Pvz. tas pats tekstas pavadinime turi kitą reikšmę nei tekstas dokumento pabaigoje, reiškia, kad kažkoks skirtumas turi būti ne tik informacijos turinyje. Galima pastebėti, kad beveik visi dokumentai turi būtent tokią struktūrą.

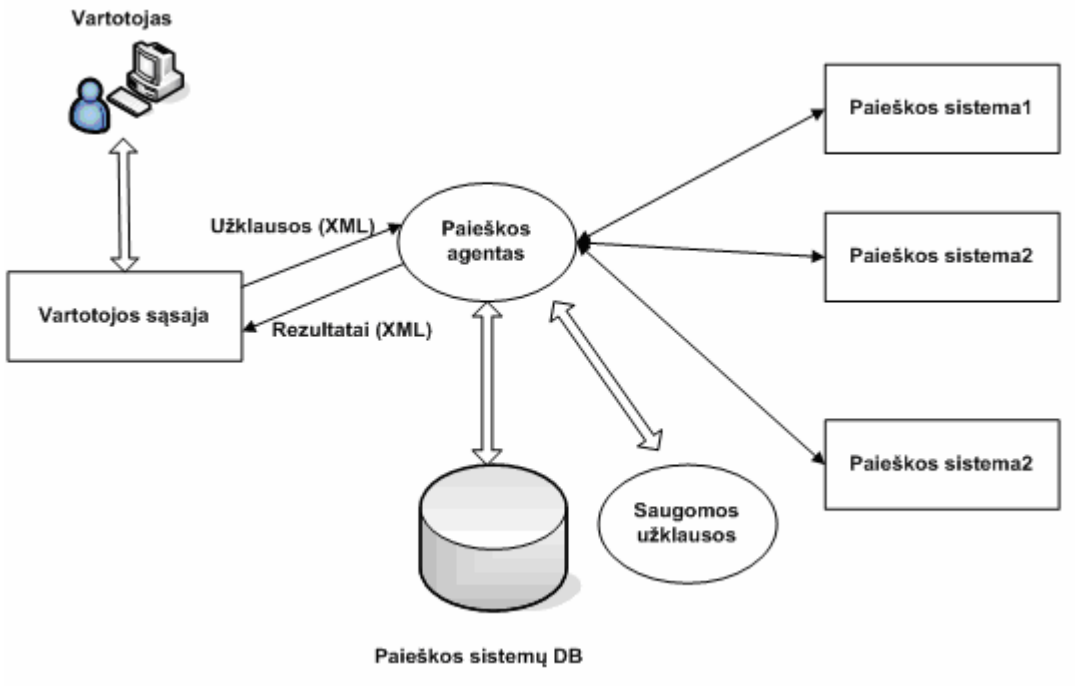
XML atitinka tokius reikalavimus:

- XML turi būti tiesiogiai panaudojama Internete;
- skirtingos aplinkos turi palaikyti XML;
- XML turi būti lengvai panaudojamas programavimui;
- pasirinktinų ar kintančių XML savybių turėtų būti kuo mažiau, idealiausia, kad jų nebūtų visai;
- XML turėtų būti lengvai įskaitomas žmogui (priešingai nei binarinės informacijos laikmenos);
- XML ruošiniai ar projektai turėtu būti nesunkiai sukuriami ir paruošiami;
- XML aprašymas turėtų būti formalus ir nesukelti dviprasmybių.

Šių reikalavimų užtikrinimas garantuoja, kad XML yra tinkama priemonė naudoti kaip duomenų aprašymo priemonę.

Dauguma interneto paieškos sistemų pagal tam tikrus kriterijus automatiškai surenka informacijos vienetus (html puslapius, tekstinius dokumentus ir kt.) į DB, juos indeksuoja ir užtikrina paiešką pagal raktinius žodžius. Paprastai šios paieškos sistemos turi skirtingas paieškos

užklausų taisykles. Ieškodamas informacijos internete, vartotojas turi išmanyti jas visas. „Šokinėti“ nuo vienos paieškos sistemos prie kitos taip pat nėra patogu. Tam tikslui vartotojui būtų gerai turėti vieną paieškos sistemą (engine), kuri užtikrintų unifikuotą vartotojo sąsają, transliuotų paieškos užklausas iš bendro užklausų formato į skirtingus užklausų formatus skirtingose informacijos paieškos sistemose. Paieškos skirtingose sistemose rezultatai galėtų būti grąžinami vartotojui standartizuotu formatu. Apibendrinta tokio metapaieškos agento schema pateikta pav. 6.



6 pav. Paieškos agentas

Kiekviena paieškos sistema gali naudotis įvairiomis duomenų bazių valdymo sistemomis, turėti skirtingą architektūrą, struktūrą, perdavimo protokolus. Net pats duomenų bazės aprašymas gali būti skirtingas.

Šiuo metu nėra technologijos, leidžiančios automatiškai konvertuoti duomenų bazės struktūrą, atitinkančią sprendžiamus uždavinius, į XML formatą. Todėl vienas iš pagrindinių uždavinių buvo XML formato, aprašančio duomenų bazės struktūrą sudarymas. Toks formatas turi būti nepriklausomas nuo informacijos (duomenų bazės turinio) ir jos paskirties.

Kad gauti struktūrą, visų pirma reikia apspręsti, kokios reliacinės duomenų bazės esybės bus aprašomos XML formatu. Tai:

- Lentelės;
- Ryšiai;
- Raktai[6];
- Laukai.

[6] minimalus atributų rinkinys, vienareikšmiškai apibrėžiantis atskirą įrašą

Tada nustatome taisykles, kurių pagalba bus aprašomos esybės. Kaip ir reliacinėse duomenų bazėse, pagrindinis struktūrinis vienetas yra *lentelė*. Jos žymėjimui naudojame elementą <table>. Lentelių aibė žymima elementu <tables>. Lentelėje pagrindinis vienetas yra *laukas*, kurį žymėsime elementu <field>. Lentelės laukų visuma žymima < fields>. Būtina nurodyti lentelės vardą - <name>. Yokiu būdu turime bendrą struktūros karkasą:

```
<root>
        <tables>
                <table>
                        <name> lentelės_vardas</name>
                        <fields>
                                <field>lauko_vardas </field>
                                <field>lauko_vardas </field>
                                <field>lauko_vardas </field>
                        </fields>
                </table>
                <table> … </table>
                <table> … </table>
                        …
        </tables>
</root>
```

Kitų atributų aprašymui naudosime papildomas XML žymes. Panagrinėkime konkretų pavyzdį:

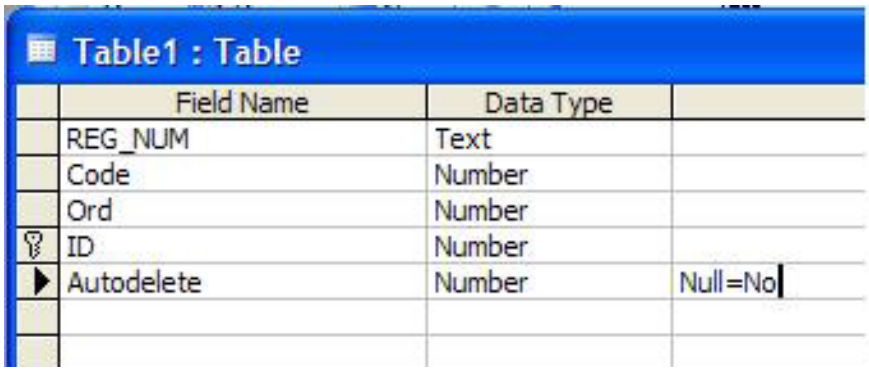


Table1 : Table

	Field Name	Data Type	
	REG_NUM	Text	
	Code	Number	
	Ord	Number	
	ID	Number	
▶	Autodelete	Number	Null=No

7 pav. Lentelė TABLE duomenų bazėje

Tada lentelė TABLE XML formate galima aprašyti:

```
<table>
        <name>TABLE</name>
                <fields>
                        <field key=«on» type=«int»>ID</field>
                        <field rootkey=«on» hookTable=«MAIN» hookField=«Reg_Num»>
REG_NUM</field>
                        <field type=«int»>CODE</field>
                        <field type=«int»>AutoDelete</field>
                        <field type=«int»>Ord</field>
                </fields>
</table>
```

Duomenų tipai laukams nenurodami, išskyrus sveikaskaitinius type=«int». Sveikaskaitiniams laukams reikšmės nebus imamos į kabutes «».

Pirminis raktas nurodomas atributu key=«on».

Ryšiai tarp lentelių aprašomi išorinių raktų pagalba. Kaip ir duomenų bazėje, gali būti keli išoriniai raktai. Jie žymimi raktiniu žodžiu rootkey=«on». Be to, reikia nurodyti lentelės pirminiu raktu ir lauką, pagal kurį vykdomas ryšys hookTable=«MAIN» hookField=«Reg_Num». Jei lentelė yra pirma hierarchijoje, tai šios lentelės ryšys pačia su savimi.

Tokiu būdu XML formate aprašyti bet kokios duomenų bazės struktūrą. Ši struktūra yra dalykinės srities ontologijos dalis ir yra pagrindas formuoti užklausą XML formate.

Išskleistos XML užklausos pavyzdys gali būti:

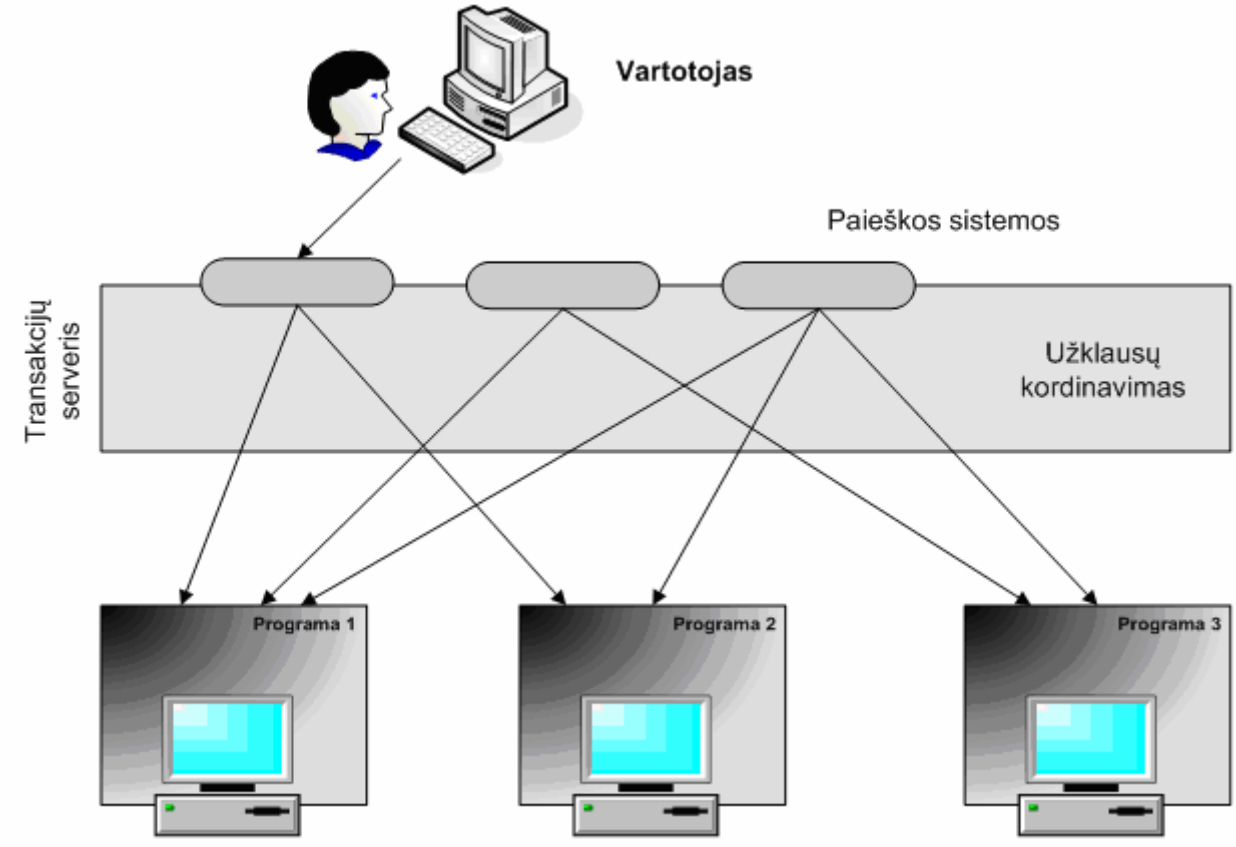
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE QUERY SYSTEM "query.dtd">
<QUERY>
<PROPERTYNAME>title</PROPERTYNAME>
<CONSTRAINTS> <CONSTRAINTS>
<CONTAINS propertyName="title" value="Ontology"/>
</CONSTRAINTS>
<AND/>
<CONSTRAINTS>
<PARENTHESISOPEN/>
<CONSTRAINTS> <CONSTRAINTS>
<NEGATIONOPEN/>
<CONSTRAINTS>
<EQUALS propertyName="author" value="Mark Walberg"/>
</CONSTRAINTS>
<NEGATIONCLOSE/>
</CONSTRAINTS>
<OR/>
<CONSTRAINTS>
<GREATER propertyName="pages " value="500"/>
</CONSTRAINTS> </CONSTRAINTS>
<PARENTHESISCLOSE/>
</CONSTRAINTS> </CONSTRAINTS>
</QUERY>
```

Tokiu būdu, kita svarbi dalis yra užklausos transformavimo modulis.

2.6. Transakcijų serveriai

2.6.1. Transakcijų serverio sąvoka

Transakcijų serveriai arba transakcijų valdymo programos (angl. transaction processing monitors) yra tarpinė programinė įranga (angl. middleware) veikianti tarp klientų ir serverių. Labiausiai paplitę yra šie transakcijų serveriai: IBM CICS, Encina, Microsoft Transaction Server, BEA Tuxedo. Transakcijų serverių funkcionalumas neapsiriboja vien tik paprastu tarpininkavimu tarp kliento ir serverio. Transakcijų serveris leidžia kurti sąsajas su daugeliu paskirstytų programų kartu palaikant transakcines savybes (8 pav.). Jis išplečia transakcijų valdymo galimybes už duomenų bazės ribų ir kartu teikia priemones ir įrankius kurti taikomąsias programas, kuriose būtų užtikrintas sklandus transakcijų vykdymas.



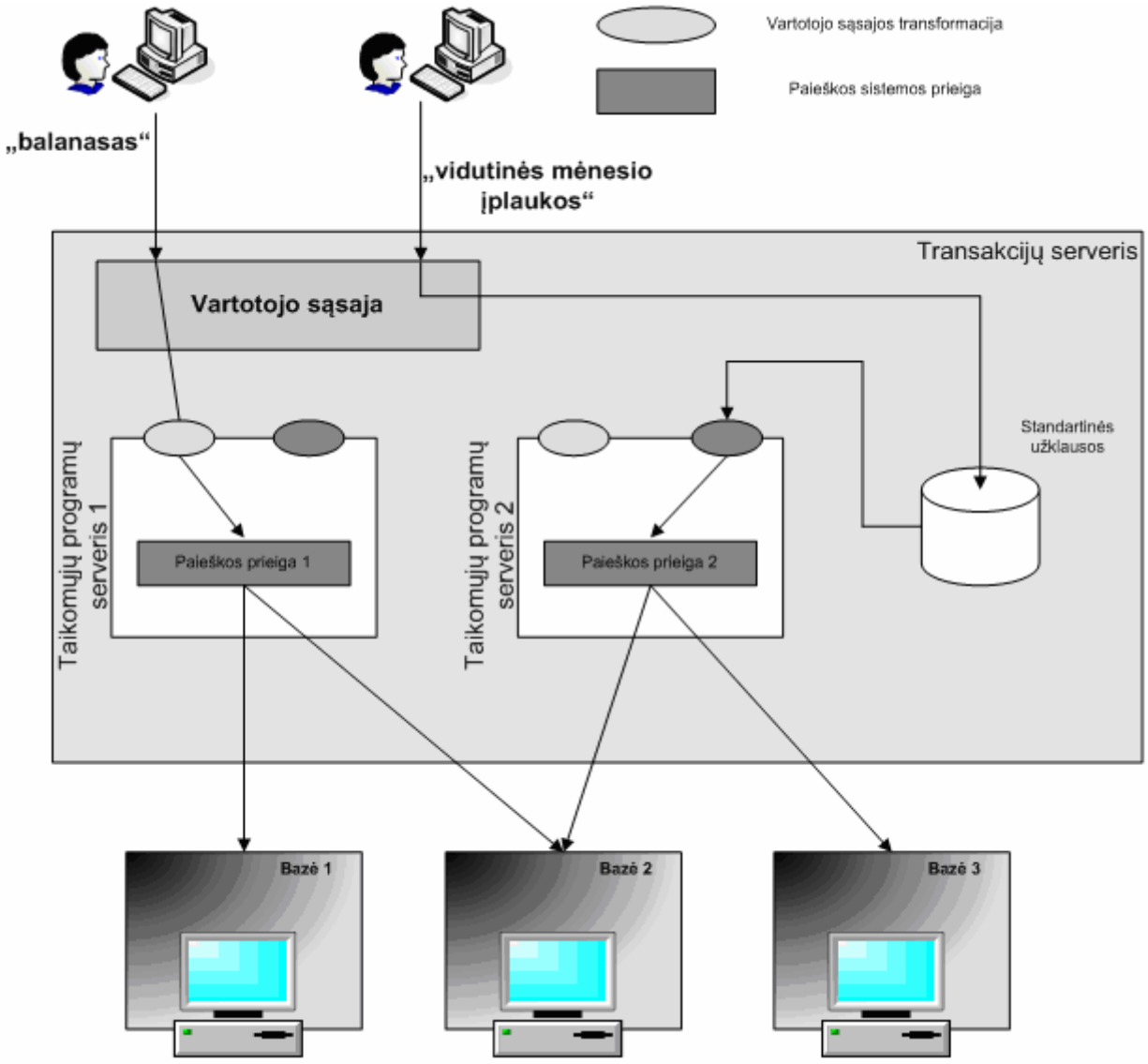
8 pav. Transakcijų serveris, jungiantis klientus su paskirstytomis programomis

Transakcijų serverių atsiradimą sąlygojo tai, kad operacinės sistemos nebuvo skirtos transakcijų apdorojimui. Pažymėtina, kad bendras transakcijų serverių funkcionalumas nėra tiksliai apibrėžtas ir dažnai priklauso nuo konkrečios sistemos. Taip yra todėl, kad transakcijų serveriai buvo kuriami, kaip antro lygmens operacinės sistemos ant esamų operacinių sistemų. Bendru atveju

transakcijų serveris – tai integravimo įrankis, kuris leidžia projektuotojams apjungti skirtingus sistemos komponentus, panaudojant įvairias pagalbines priemones. Panaudojant šias priemones integravimas tampa daug paprastesnis, nes didžiąją dalį reikalaujamo funkcionalumo palaiko pats transakcijų serveris.

Transakcijų serveris sprendžia tokias problemas, kaip daugelio vartotojų prieiga prie nevienalyčių paskirstytų duomenų šaltinių, tiksliai apibrėžtų sąsajų teikimas bei transakcinių savybių užtikrinimas. Transakcijų serveriai taip pat atlieka resursų rakinimo, resursų prieigos paskirstymo (angl. scheduling), įvykių žurnalo vedimo (angl. logging) bei atstatymo funkcijas.

Bendrinė transakcijų serverio architektūra pateikta 9 pav. Klientai per bendrą sąsają jungiasi prie transakcijų serverio, kuris koordinuoja daugelio taikomųjų programų serverių veiksmus.



9 pav. Apibendrinta transakcijų serverio architektūra

Pagrindinės transakcijų serverio teikiamos paslaugos yra šios:

- Transakcinis nutolusių procedūrų iškvietimas (angl. remote procedure call).
- Transakcijų valdymas: atliekamas dviejų fazių įrašymas ir užtikrinama atstatymo galimybė.
- Įrašų žurnalo (angl. log) valdymas: fiksuojami visi transakcijų metu atlikti pakeitimai, tam, kad nesėkmės atveju sistemą būtų galima atstatyti.
- Rakinimo valdymas: prieigos prie bendrų duomenų reguliavimas.

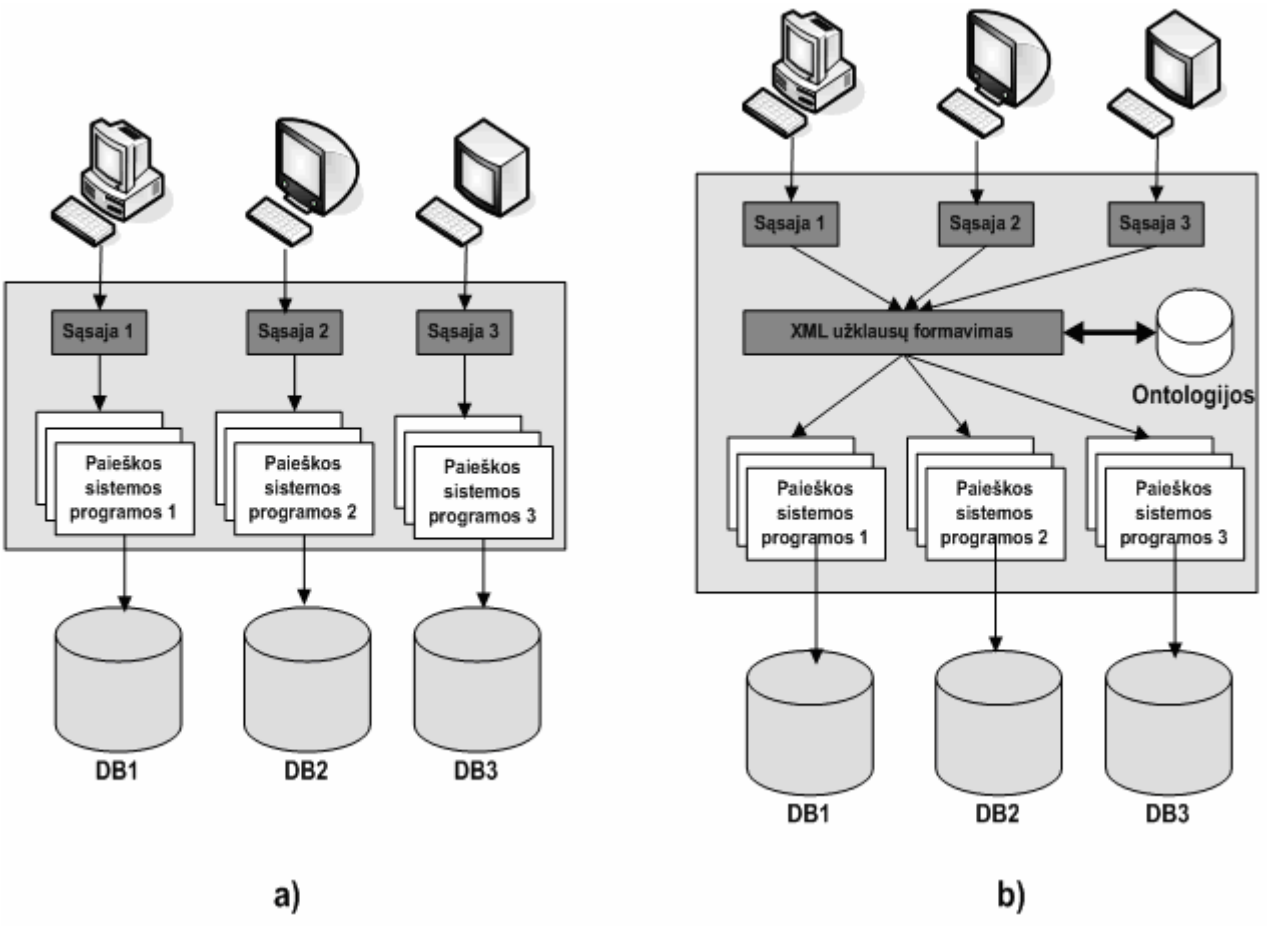
Papildomos transakcijų serverių teikiamos paslaugos:

- Serverių valdymas ir administravimas: serverių paleidimas, sustabdymas, apkrovimo paskirstymas.
- Autentifikacija ir autorizacija: tikrinama ar vartotojas gali iškviesti paslaugą, iš tam tikro terminalo, tam tikru laiku ir su tam tikru parametrų rinkiniu.
- Transakcijų eilės: asinchroninei sąveikai tarp komponentų.

2.6.2. Transakcijų serverių struktūra

Tam kad užtikrintų efektyvų transakcijų vykdymą transakcijų serveriai ne tik gali perimti dalį operacinės sistemos funkkcijų, bet atlikti užklausos transformavimą į bendrinę užklausą, kuri remiasi ontologijos pagrindu ir nepriklauso nuo duomenų bazės struktūros. 10 pav. pateikiama klasikinė (a) ir siūloma (b) transakcijų serverio architektūros.

Padalinamos struktūros atveju funkcionalumas padalinamas tarp daugelio nepriklausomų paskirstytų procesų.



10 pav. Klasikinė (a), ir siūloma (b) transakcijų serverių struktūros

Esant siūlomai struktūrai išvengiama priklausomybė nuo atskirų paieškos sistemų pateikiamų reikalavimų užklausų struktūrai.

3. Reikalavimų programinei įrangai analizė

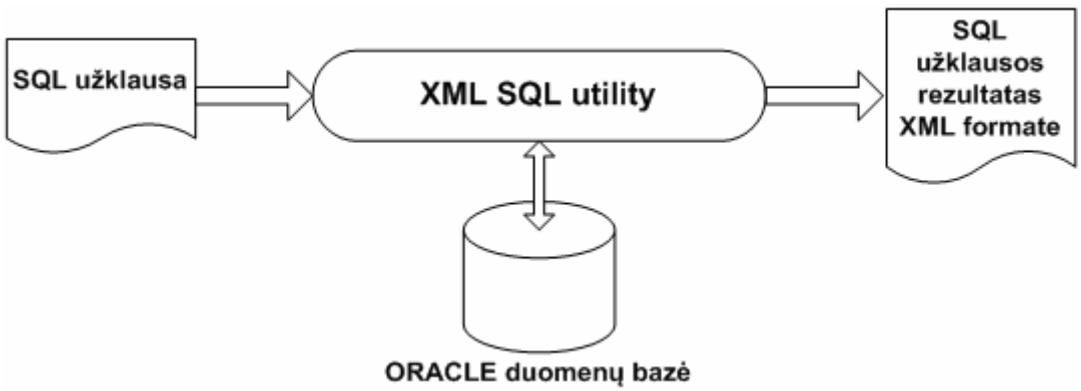
3.1. XML technologijos duomenų bazių valdymo sistemose

Norint atlikti skirtingų tipų duomenų šaltinių integraciją, visų pirma reikia turėti visiems integruojamiems šaltiniams bendrą schemą, kuri leistų atlikti integraciją. Šiame darbe nusprendėme pasirinkti XML kalbą ir šios kalbos struktūrą apibrėžiančias XML schemas. Panagrinėsime XML kalbos ir schemų privalumus bei trūkumus, aptartsime jų sandarą bei panaudojimą.

3.1.1. Oracle duomenų bazė ir XML panaudojimo galimybės

Oracle[7] turi XML failų nagrinėjimo priemones (angl. *parsers*) Java, C, C++ ir PL/SQL kalboms, todėl programuojant šiomis kalbomis XML panaudojimas yra lengvesnis.

Naudojant XML SQL Utility galima vykdyti SQL užklausas ir gauti rezultatus XML dokumentą iš suformuotų užklausų rezultatų (11 pav.).

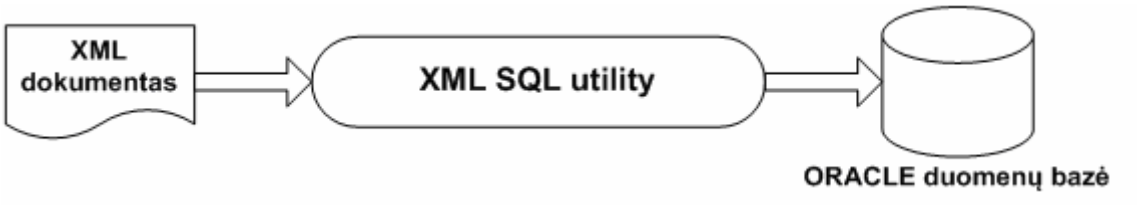


11 pav. XML dokumento formavimas naudojant XML SQL Utility

Gauto XML dokumento struktūra atitinka Oracle DB struktūrą, kuri buvo nurodyta formuojant SQL užklausą. Duomenų bazės stulpelių pavadinimai atitinka XML elementų grupių pavadinimus, o reikšmės yra priskiriamos vidiniams elementams.

Taip pat naudojantis XML SQL Utility galima ir surašyti XML duomenis į Oracle duomenų bazę (12 pav).

[7] www.oracle.com



12 pav. XML dokumento duomenų surašymas į Oracle DB

Saugant duomenis iš XML dokumento į Oracle duomenų bazę katu išsaugoma ir dokumento struktūra. XML dokumento elementų pavadinimai atitinka duomenų bazės lentelių ir lentelių stulpelių pavadinimus, o reikšmės yra įrašomos į atitinkamus lentelės laukus.

3.1.2. MS SQL Server duomenų bazė ir XML panaudojimo galimybės

MS SQL Server[8] duomenų bazės duomenis į XML galima perduoti trimis būdais:

- Baigiant SELECT sakinį FOR XML;
- XPath queries priemonių pagalba, kurios komentuoja XML schemas;
- OpenXML funkcija procedūrose.

OpenXML funkcija dažniausiai naudojama atvaizduoti XML dokumento duomenims į sąryšinės bazės lenteles.

MS SQL Server yra galimos tokios XML savybės:

- HTTP pasiekiamumas (rašant SQL sakinį naršyklės adresų eilutėje);
- Formuoti XML galima SELECT sakinio pagalba (FOR XML);
- XML views (nurodoma kaip sąryšiniai duomenys turi būti atvaizduoti XML);
- XPath queries (nurodoma kaip duomenys turi būti paimti iš DB);
- OpenXML (nurodo, kaip duomenys turi būti atvaizduoti XML ir suteikia galimybę XML failo duomenis atvaizduoti duomenų bazėjes lentelėse);
- OLE DB ir ADO pasiekiamumas (naudojama sąsajai su sąryšinėmis DB formuojant XML iš DB ir surašant XML duomenis į DB).

[8] MS SQL Server

3.1.3. Oracle ir MS SQL Server teikiamų XML savybių palyginimas

1 lentelė. Oracle ir MS SQL Server teikiamos XML savybės

Savybės	SQL Server 2000	Oracle
Deklaratyvus žymėjimas laisvai susietų verslo sistemų (ši savybė suteikia dvikrypti žymėjimą tarp XML ir sąryšinių duomenų vaizdų). SQL Server 2000 turi integruotas XML savybes, kurios leidžia priėjimą prie sistemos su minimaliu programavimu. Oracle neturi integruotų XML specifinių savybių.	XML Views XPath support Transact-SQL FOR XML išplėtimai Transact-SQL OPENXML išplėtimai	Neturi specifinių savybių. Reikia rašyti papildomas programas.
Vietiniai XML išplėtimai skirti SQL ir leidžiantys vartotojui vykdyti aplikaciją greitai ir be papildomo programavimo. SQL Server 2000 teikia išplėtimus SQL kalbai, kurie gali būti panaudoti gražinti XML duomenis iš standartinių SQL užklausų. Oracle teikia serverio pagalbinius įrankius, kuriuos vykdant neišplečiama SQL ir reikalaujama papildomo programavimo gražinant XML duomenis.	Transact-SQL FOR XML išplėtimai Transact-SQL OPENXML išplėtimai	Neturi specifinių savybių. Reikia rašyti papildomas programas naudojant XSQL Utility.
Įvairūs XML prieigos metodai, kurie palaiko lankstumą, leidžiantį programuotojams pasiekti XML duomenis per tinklo prisijungimus. SQL Server 2000 ir Oracle palaiko įvairius metodus leidžiančius pasiekti XML duomenis, esančius duomenų bazėse.	HTTP prieiga OLE DB/ADO prieiga JDBC	HTTP prieiga OLE DB/ADO prieiga JDBC
XML šablonai SQL Server ir Oracle palaiko	URL/HTTP prieiga XML šablonai	URL/HTTP prieiga XSQL šablonai

išsaugotus XML užklausų šablonus serveryje su laisvai pasirenkamais parametrais.		
Saugi Web aplikacijų prieiga prie XML duomenų SQL Server 2000 palaiko saugumą, kuris gali būti valdomas individualios lentelės lygyje. Oracle palaiko apribotą saugumą tik duomenų bazės vartotojo, bet ne individualios lentelės lygio.	Duomenų bazės vartotojo lygmens saugumas Duomenų bazės objekto lygmens saugumas	Duomenų bazės vartotojo lygmens saugumas
Galimybės detalesnei paieškai sudėtingiems XML dokumentams. SQL Server 2000 ir Oracle teikia duomenų bazės palaikomą sudėtingų tekstinių dokumentų paiešką apimančią ir XML.	Pilna teksto paieška	Tarpinė teksto paieška su segmentų savybėmis.
XML failų nagrinėjimo priemonės (angl. *parsers*) (skirti programuotojams, kurie siekia didesnės galios ir lankstumo, kai kuriamos XML aplikacijos).	MSXML COM priėjimo parserį palaiko daugelis kalbų	Oracle parserius palaiko daugelis kalbų

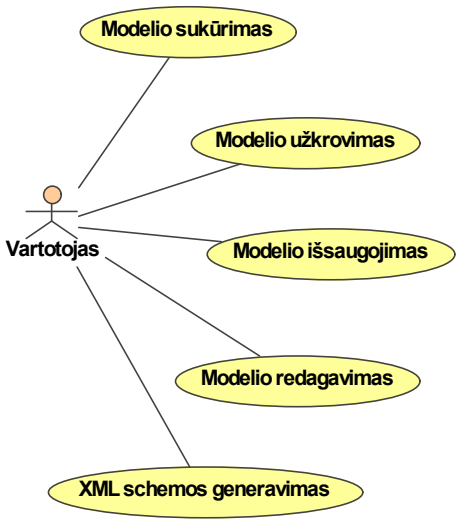
Microsoft SQL Server 2000 siūlo XML savybes daugiausia skirtas Web ir duomenų bazių programuotojams. Web programuotojams nereikia mokintis duomenų bazių programavimo subtilybių norint pasinaudoti SQL Server 2000 XML privalumais, nes SQL Server teikia standartines XML konstrukcijas. Taip pat, duomenų bazių programuotojams nereikia išmanyti XML programavimo subtilybių, nes jie gali naudotis FOR XML ar OpenXML siekdami manipuliuoti XML duomenimis.

Oracle labiau skirtas naudoti patyrusiems programuotojams. Norint pasiekti sprendžiamų uždavinių rezultatą tenka naudotis keliomis priemonėmis arba kurti specifinius įrankius.

3.2. Reikalavimų specifikacija

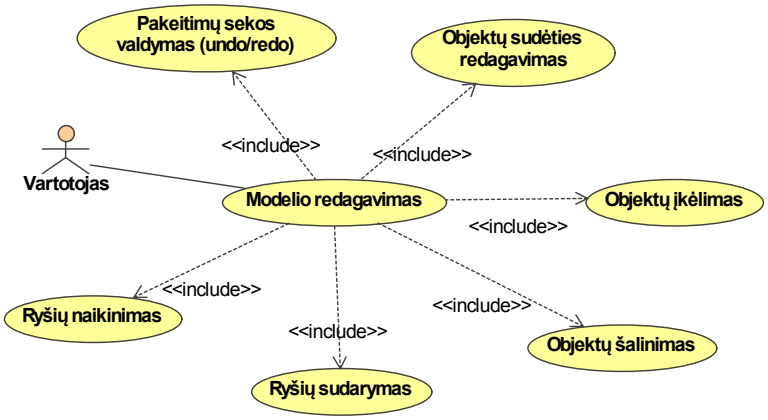
3.1.1. Panaudojimo atvejai

Duomenų šaltinių integracija apima atskirų duomenų šaltinių koncepcijų ir supratimo sujungimą į bendrą vaizdinį, kuris neparodo atskirų sistemų detalių. Izoliuojant vartotoją nuo duomenų šaltinių ir jų supratimo sujungimo sudėtingumo, pasiekiama tai, kad sistemos iš vartotojo perspektyvos tampa suderinamos, nes jos gali pasiekti duomenis visuose duomenų šaltiniuose, neatsižvelgiant į tai, kaip šitą užduotį atlikti. Vartotojų sąsaja apibendrinama šiais panaudos programos atvejais. Taikomoji programa turi tik vieną vartotojo tipą.

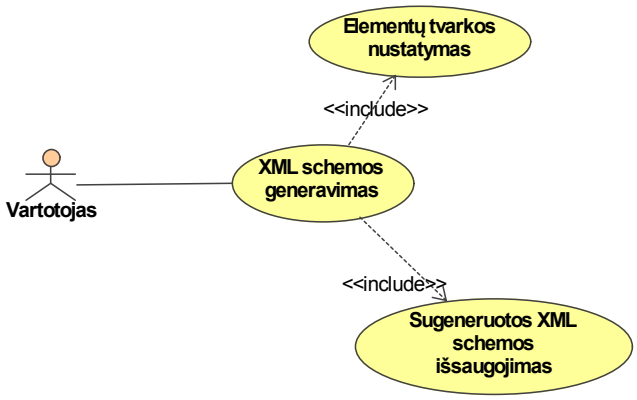


13 pav. Pagrindiniai panaudos atvejai.

Modelio redagavimo bei XML schemos generavimo panaudos atvejai yra detalizuojami toliau.



9 pav. Modelio redagavimo vidiniai panaudos atvejai



14 pav. XML schemos generavimo vidiniai panaudos atvejai

3.1.2. Panaudojimo atvejų specifikacijos

Šiame skyriuje pateikiamos panaudojimo atvejų specifikacijos.

2 lentelė. Modelio sukūrimo panaudos atvejo specifikacija

Prieš sąlyga	<Nėra>
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka modelio sukūrimo veiksmą arba programa yra paleidžiama.	1.1. Jei šiuo metu programoje jau yra redaguojamas modelis, užklausiama ar galima jį naikinti.
Po sąlyga	Sistema sunaikino senąjį modelį ir sudarė naują.

3 lentelė. Modelio užkrovimo panaudos atvejo specifikacija

Prieš sąlyga	<Nėra>
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka modelio užkrovimo veiksmą.	1.1. Jei šiuo metu programoje jau yra redaguojamas modelis, užklausiama ar galima jį naikinti.
2. Vartotojas patvirtina, kad dabartinis modelis gali būti naikinamas.	2.1. Sistema naikina dabartinį modelį. 2.2. Išveda failo pasirinkimo dialogą vartotojui.
3. Vartotojas pasirenka norimą užkrauti failą failų sistemoje.	3.1. Sistema patikrina ar failas gali būti užkrautas 3.2. Sistema užkrauna bei apdoroja modelio failą sukurdama ir atvaizduodama modelį vartotojui.
Po sąlyga	Sistemoje senas modelis ištrinamas ir vaizduojamas vartotojo pasirinktas užkrautas modelis.

4 lentelė. Modelio išsaugojimo panaudos atvejo specifikacija

Prieš sąlyga	Sistemoje yra sudarytas netuščias modelis.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka modelio išsaugojimo veiksmą.	1.1. Sistema vartotojui parodo failo pasirinkimo dialogą.

2. Vartotojas nurodo failą (failo pavadinimą) kuriame turėtų būti išsaugotas modelis.	2.1. Jei toks failas jau egzistuoja, sistema paklausia dialogu ar vartotojas nori užrašyti ant seno failo turinio.
3. Vartotojas patvirtina išsaugojimą nurodytu pavadinimu.	3.1. Sistema išsaugo modelį į nurodytą išorinį failą.
Po sąlyga	Sistema išsaugoja dabartinį modelį į išorinį failą.

5 lentelė. Modelio redagavimo panaudos atvejo specifikacija

Prieš sąlyga	Sistemoje yra užkrautas modelis.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Sistema pateikia vartotojui grafinius modelio redagavimo įrankius.	1.1. Vartotojas atlieka modelio pakeitimus.
Po sąlyga	Atlikti modelio pakeitimai.

6 lentelė. Objektų įkėlimo panaudos atvejo specifikacija

Prieš sąlyga	Sistemoje sudarytas modelis.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka objekto įkėlimą.	1.1. Sistema pateikia dialogą objekto savybėms suvesti (pavadinimui, ir pan.)
2. Vartotojas suveda objekto savybes į objekto savybių redagavimo langą.	2.1. Sistema sukuria objektą su vartotojo nurodytomis.
Po sąlyga	Į modelį įkeltas naujas objektas.

7 lentelė. Objektų šalinimo panaudos atvejo specifikacija

Prieš sąlyga	Pažymėtas kokio nors objektas.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka objekto pašalinimo.	1.1. Sistema užklausia vartotojo dialogu ar tikrai norima naikinti objektą kartu su jo ryšiais su kitais.
2. Vartotojas patvirtina objekto naikinimo veiksmą.	2.1. Šalinami objekto ryšiai su kitais objektais. 2.2. Objektas naikinamas modelyje.
Po sąlyga	Iš modelio pašalintas objektas ir jo ryšiai su kitais objektais.

8 lentelė. Objektų sudėties redagavimo panaudos atvejo specifikacija

Prieš sąlyga	Pasirinktas koks nors modelio objektas.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka objekto turinio redagavimo veiksmą.	1.1. Sistema parodo objekto sudėtį (atributus, savybes).
2. Vartotojas papildo/šalina objekto atributus, keičia objekto savybes (pavadinimą ir pan.)	
3. Vartotojas pasirenka pakeitimų atlikimo veiksmą.	3.1. Sistema išsaugo objekto pakeitimus modelyje pagal vartotojo suvestus pakeitimus.
Po sąlyga	Pakeista modelio objekto sudėtis.

9 lentelė. Ryšių sudarymo panaudos atvejo specifikacija

Prieš sąlyga	Sistemoje yra netuščias modelis.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka ryšio įkėlimo veiksmą.	
2. Vartotojas nurodo objektus, tarp kurių yra sukuriamas ryšys.	2.1. Sistema patikrina ar ryšys gali būti sukurtas (ar toks jau egzistuoja, ar galimas). 2.2. Jei ryšys gali būti sudarytas, jis įkeliamas į modelį. 2.3. Jei ryšys negalėjo būti sukurtas, apie tai pranešama vartotojui.
Po sąlyga	Į modelį įkeltas ryšys tarp dviejų vartotojo pasirinktų objektų arba vartotojui nurodoma, kad ryšys tarp jo nurodytų objektų negalimas.

10 lentelė. Ryšių naikinimo panaudos atvejo specifikacija

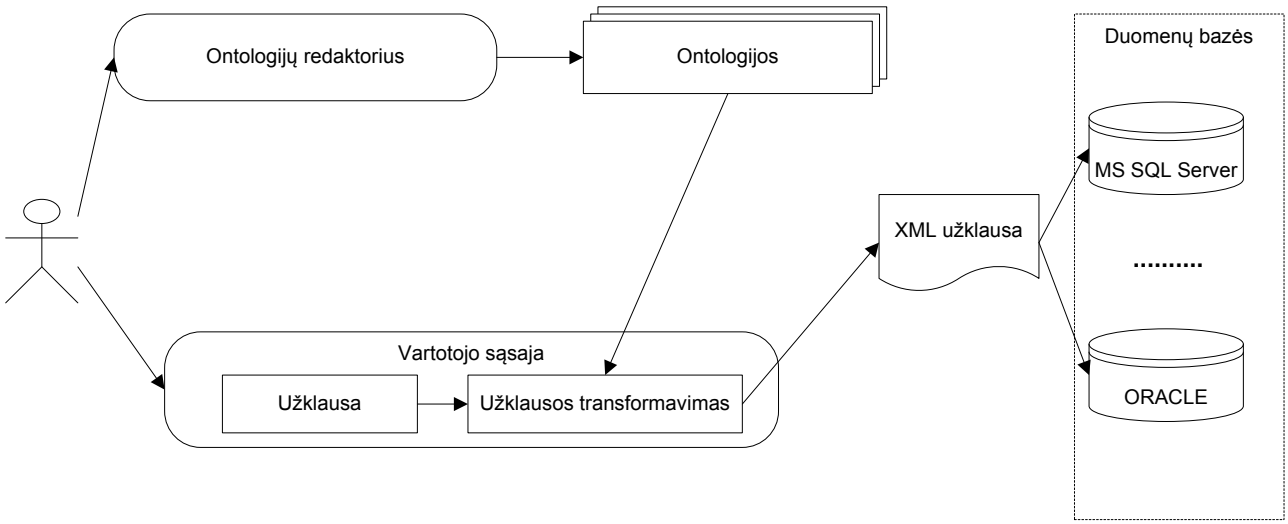
Prieš sąlyga	Sistemoje yra pasirinktas ryšys tarp objektų.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka ryšio naikinimo veiksmą.	1.1. Sistema dialogu paprašo patvirtinti ryšio tarp objektų naikinimą.
2. Vartotojas pasirenka ryšio naikinimo veiksmo patvirtinimą.	2.1. Sistema šalina vartotojo nurodytą ryšį tarp dviejų objektų iš programoje užkrauto modelio.
Po sąlyga	Ryšys panaikintas iš modelio.

11 lentelė. XML schemos generavimo panaudos atvejo specifikacija

Prieš sąlyga	Sistemoje yra užkrautas netuščias modelis.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka XML schemos generavimo veiksmą.	1. Sistema generuoja schemą ir ją išsaugo vartotojo nurodytame faile.
Po sąlyga	Sistema sugeneravo ir išsaugojo vartotojo pasirinktą modelį kaip XML schemą į išorinį failą.

3.2. Siekiamos sistemos apibrėžimas

Vartotojo sąsaja ir ontologijų redaktorius, kaip taikomoji programa, yra izoliuota, nepriklausoma nuo išorinių sąlygų, sistema. Duomenys apsikeičiami XML formato failais.



15 pav. Sistemos kontekstas

Vartotojas nepriklausomai nuo užklausos gali sudaryti ontologijų modelius, kuriuos ontologijų redaktorius išsaugo kaip XML schemas. Vartotojo užklausa užklausų transformatorius, pagal ontologijų modelius, transformuoja į XML užklausą, kuri nepriklauso nuo DB struktūros.

3.3. Vartotojų analizė

3.3.1. Vartotojų aibė, tipai ir savybės

Sistemos vartotojais numatomi dviejų tipų vartotojai. Orientuojantis į tai, kad reikalingas ontologijų modelių kūrimas vienas vartotojų tipas turėtų būti probleminės srities žinovas. Kitas varotojo tipas turėtų būti paprastas vartotojas, tačiau susipažinęs su XML pagrindais, kad iškilus reikalui galėtų koreguoti suformuotą užklausą.

Vartotojai-žinovai naudosis ontologijų redaktoriumi sudaryti XML schemas iš duomenų modelių, kurie nustatyti sistemos analizės fazėje, todėl jiems reikalinga, kad nebūtų didelio atotrūkio tarp pradinio duomenų modelio bei schemos sąvokų bei sudarymo principo. Transformuoti užklausai ontologijos pagalba taip pat naudojamas XML schemų generatoriai.

3.3.2. Vartotojų tikslai ir problemos

Skirtingose duomenų bazėse saugomų duomenų kiekis pastoviai auga. Dauguma šių duomenų bazių yra pasiekiamos per Internetą arba vidinį tinklą. Tačiau netgi patį Internetą galima laikyti didele paskirstyta heterogenine duomenų baze, kuria dažniausiai naudojasi paprasti vartotojai. Atliekant vieną duomenų transakciją galima gauti, siųsti ir apdoroti įvairiems duomenų šaltiniams skirtus duomenis. Taip pat vienoje transakcijoje apdoroti duomenys gali būti skirtingose duomenų bazėse, kurios yra skirtingose operacinėse sistemose. Tokiu būdu, pagrindinis vartotojo tikslas yra gauti jo norimą informaciją be praradimų, kuriuos gali sukelti skirtingų struktūrų ir tipų duomenų bazės, esančios skirtingose platformose.

3.4. Vartotojo sąsajos realizavimo platformos bei architektūros parinkimas

Siekiant parinkti optimalią programos architektūrą, buvo nagrinėjami keli šiame skyriuje nurodyti variantai. Pagrindiniai kriterijai parinkimui yra šie:

- Naudojamos programavimo kalbos patogumas.
- Platformos bibliotekų apimtis, XML schemų apdorojimo priemonių galimybės, prieinamumas bei dokumentacijos kokybė.
- Numanoma realizavimo sparta, platformos taikomųjų programų kūrimo priemonių įvairovė ir kokybė.
- Numatomos programos architektūros paprastumas, redaktoriaus funkcijų realizavimo būdų įvairovė, išplečiamumas.

3.4.1. MVC architektūra GEF priemonėmis, Eclipse architektūra

Eclipse platforma, paremta Java programavimo kalba, pateikia platų bibliotekų rinkinį ir yra lengvai išplečiama įskiepių (angl. *plugin*) pagalba. Vienas iš įskiepių – GEF [6] (Grafinio redagavimo platforma – angl. Graphical Editing Framework) yra MVC[9] architektūros grafinio redagavimo klasių rinkinys, kuris gali būti išplečiamas sukurti plataus profilio grafinius redaktorius. Eclipse architektūra turi išplečiamą XML schemų bibliotekų rinkinį XSD (XML Schema Infoset Model) [7], kuris būtų naudojamas XML schemų apdorojimui.
Ontologijų redaktorius veiktų kaip įskiepis Eclipse platformoje. Jis būtų modeliuojama MagicDraw įrankiu, programavimas būtų atliekamas toje pačioje Eclipse Java IDE aplinkoje.

[9] MVC (*Model-View-Controller*) karkasas, projektavimo šablonai

3.4.2. Architektūros pasirinkimas

Įvertinant šiuos architektūros modelius, buvo pasirinktas paskutinis, kadangi Eclipse GEF priemonės suteikia daug paruoštų klasių, kurios sutrumpintų modeliavimo ir realizavimo laiką naudojant MVC modelį. Be to ši platformas turi geras XML Schemos modeliavimo priemones.

4. Vartotojo sąsajos projektas

4.1. Vartotojo sąsajos architektūra

Žodyno struktūra yra modeliuojama remiantis aukščiausio lygio ontologinėmis kategorijomis, kurias pasiūlė J.Sowa[5].

4.1.1. Darbo aplinka (Workbench)

Eclipse vartotojo darbo aplinka remiasi keliomis esminėmis sąvokomis.

Projektai Eclipse aplinkoje yra resursų rinkinys realizuojantis vieną programinį sprendimą. Projektai gali būti tipizuoti, t.y. skirti tam tikro tipo (pvz. Java) sprendimams realizuoti.

Resursai yra duomenų vienetas (dažniausiai failas) platformoje, kurio apdorojimui/redagavimui gali būti priskirtas konkretus užkrautas įskiepis. Resursai gali būti tiek vietiniai tiek nutolę (per ftp, http protokolą ir pan.).

Žymės (angl. marker) yra platformos mechanizmas apipavidalinti resursus. Žymės gali būti naudojamos nurodyti kompiliavimo klaidas, darbų sąrašą, paieškos rezultatus ir kita. Žymės gali būti išplečiamos įskiepių, pritaikant tam tikram resursų tipui.

Pakeitimų istorija platformoje yra papildoma apsaugos nuo vartotojo klaidų priemonė leidžianti realizuoti vykdymo ir gražinimo (angl. undo/redo) mechanizmą visiems vartotojo atliekamiems veiksmams.

4.1.2. Vartotojo sąsajos priemonės

Eclipse platformos vartotojo sąsajos realizacija remiasi darbo aplinka, ir gali būti išplečiama įskiepių. Realizacija remiasi dviem pagrindiniais įrankiais:

SWT, vartotojo grafinių komponentų (angl. widget) biblioteka, abstrahuojanti realios vartotojo OS aplinkos komponentus Java objektais. Šia biblioteka realizuotas visas Eclipse vartotojo interfeisas. SWT sujungia skirtingų OS grafinės aplinkos komponentus į vieną interfeisą,

vienu metu pagerinant sistemos perkeliamumą bei sudarant natūralios konkrečiai OS taikomosios programos vaizdą.

JFace yra programinių priemonių biblioteka, pagreitinant vartotojo aplinkos programavimo užduotis. JFace objektais abstrahuoja modernios vartotojo aplinkos sudarymo elementus: veiksmus (angl. action), MVC elementus: rodytojus (angl. viewers) ir valdiklius.

4.1.3. Kitos priemonės

Bazinėje platformoje yra sukurta vartotojo pagalbos priemonių infrastruktūra. Paremta HTML failais, dokumentai gali būti indeksuojami, pasiekiami centralizuotai ir pagal kontekstą.

Taip pat yra realizuotos pagrindinės komandinio darbo priemonės, kurios gali būti išplečiamos priklausomai nuo naudojamos infrastruktūros. Kartu su bazine platforma yra gaunamas CVS komandinio darbo įskiepis, išplečiantis platformos komandinio darbo priemones.

4.1.4. XML užklausos generavimo procesas

Kiekvienos duomenų bazės struktūra paremta probleminės srities objektų-savybių (OS) modeliu. Tai aiškiausias modelis, nusakantis probleminės srities objektus, jų savybes bei ryšius tarp objektų. Todėl jis gali būti naudojamas ontologijų sudarymui XML schemų pagrindu. Todėl reikia apibrėžti tokio modelio transformavimas į XML schemas taisykles [10].

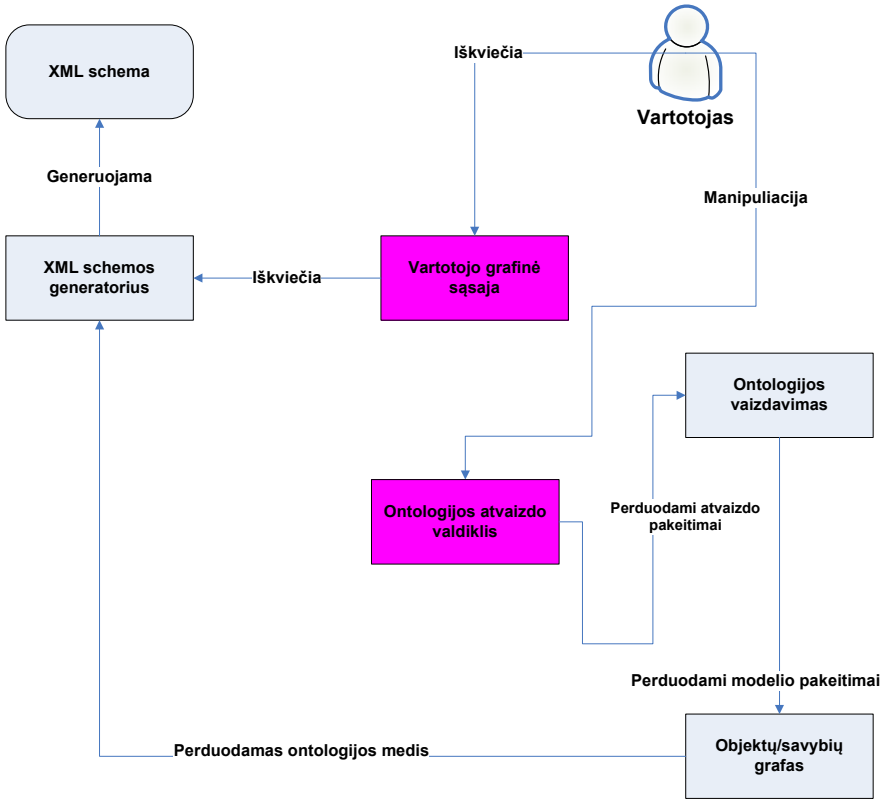
- OS objektas O transformuojamas į XML kompleksinio tipo elementą: OS(O) → XML(kompleksinio tipo elementas);
  - Objekto O atributas A, priklausomai nuo to ar turi identifikuojantį vaidmenį objekte O, yra transformuojami:
    - Į paprasto tipo elementą: OS(O.A) → XML(paprasto tipo elementas)
    - Į objekto O elemento XML atributą: OS(O.A) → XML(O kompleksinio elemento atributas);
- Objekto O vidinis sąryšis (dar vadinamas sudėtiniu atributu) yra transformuojamas skirtingai priklausomai nuo to ar turi identifikuojantį vaidmenį:
  - Jei vidinis sąryšis R neturi identifikacinio vaidmens:
    - Ryšys R transformuojamas į XML kompleksinį elementą: OS(O.R) -> XML (kompleksinis elementas);
    - Ryšio atributai R.A į vidinio ryšio elemento subelementus: OS(O.R.A) -> XML (R kompleksinio elemento subelementas);

- Jei vidinis R turi identifikacinį vaidmenį:
  - Ryšys R transformuojamas į atributų grupę: OS(O.R) -> XML (O kompleksinio elemento atributų grupė);
  - Ryšio atributai R.A į atributų grupės atributus: OS(O.R.A) -> XML (O kompleksinio elemento atributų grupės atributas);
- Funkcinė priklausomybė OS apibrėžime gali būti interpretuojama kaip atskiras OP objektas, kuriame vaizduojamos egzempliorių aibės objektų projekcija sudaro identifikacinį, o atvaizduojamosios aibės objektai – neidentifikuojantį vidinį ryšius. Taip specifikuotas OS objektas gali būti transformuojamas į XML schema pagal aukščiau aprašytas taisyklės.
- Objekto metodai neturi atitikmens XML schemoje, kadangi joje galima aprašyti tik duomenų struktūrą o ne elgseną, todėl jie gali būti į ją perkelti tik komentarų/aprašo pavidalu.
- OP modelyje vienas objektas gali būti kito objekto savybė, todėl reikalingas XML Schemos pagrindu sudarytas agregacijos ir nuorodų mechanizmas. XML Schema yra hierarchiškas dokumentas todėl natūraliausia yra objektus kaip kitų objektų savybes agreguoti tėvinių objektų XML atitikmenų elementuose.

XML schemos sudarymui yra reikalingas OS grafo medžio padengimas, nurodantis aprašomo XML medžio struktūrą. Medis padengiamas vartotojui nurodžius esybes, atributus bei ryšius kurie įeina į medžio hierarchiją. Naudodamiesi OS modelio sudedamųjų klasių diagrama analizuojame medį nuo šaknies, ir kiekvieną vis tolimesnę nuo šaknies modelio elementą transformuojame į atitinkamą XML schemos elementą.

4.2. Projekto realizacijos komponentų detalizacija

Pateikiama loginė redaktoriaus architektūra, pavaizduojant failų mainus, sąsają su vartotoju ir vidinius duomenų srautus (16 pav).

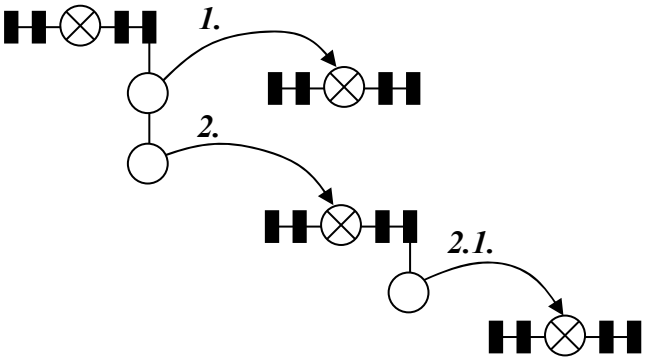


16 pav. Vartotojo sąsajos vieta informacijos paieškos mechanizme

Vartotojas manipuliuoja grafiniais elementais, kurių poveikio metu yra generuojamos užklausos (angl. *request*), kurie apdoroti to grafinio elemento valdiklio virsta komandomis (angl. *command*) keičiančiomis realius objekto/savybių modelio schemą realizuojančius objektus.

4.3. XML schemos generavimo iš objektų/savybių poschemės procesas

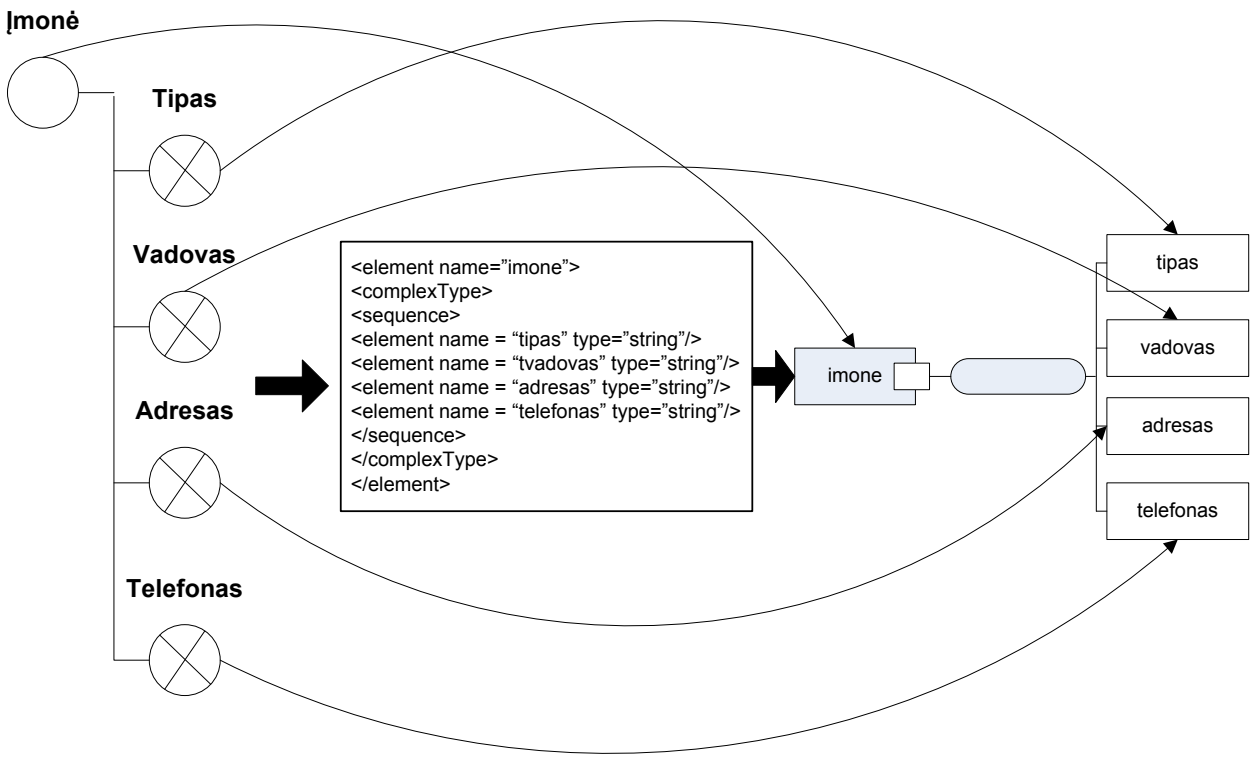
Kaip pateikta 17 pav., XML schemos sudarymui yra reikalingas OS grafo medžio padengimas, nurodantis aprašomo XML medžio struktūrą. Medis padengiamas vartotojui nurodžius esybes, atributus bei ryšius kurie įeina į medžio hierarchiją [9]. Naudodamiesi OS modelio sudedamųjų klasių diagrama analizuojame medį nuo šaknies, ir kiekvieną vis tolimesnę nuo šaknies modelio elementą transformuojame į atitinkamą XML schemos elementą.



17 pav. Objektų-savybių modelio grafo padengimas medžio tipo struktūra

Turint objektų/savybių modelio padengimą yra generuojamas jį atitinkančios XML schemos dokumentas, naudojantis taisyklėmis, apibrėžtomis 4.1.4 skyriuje.

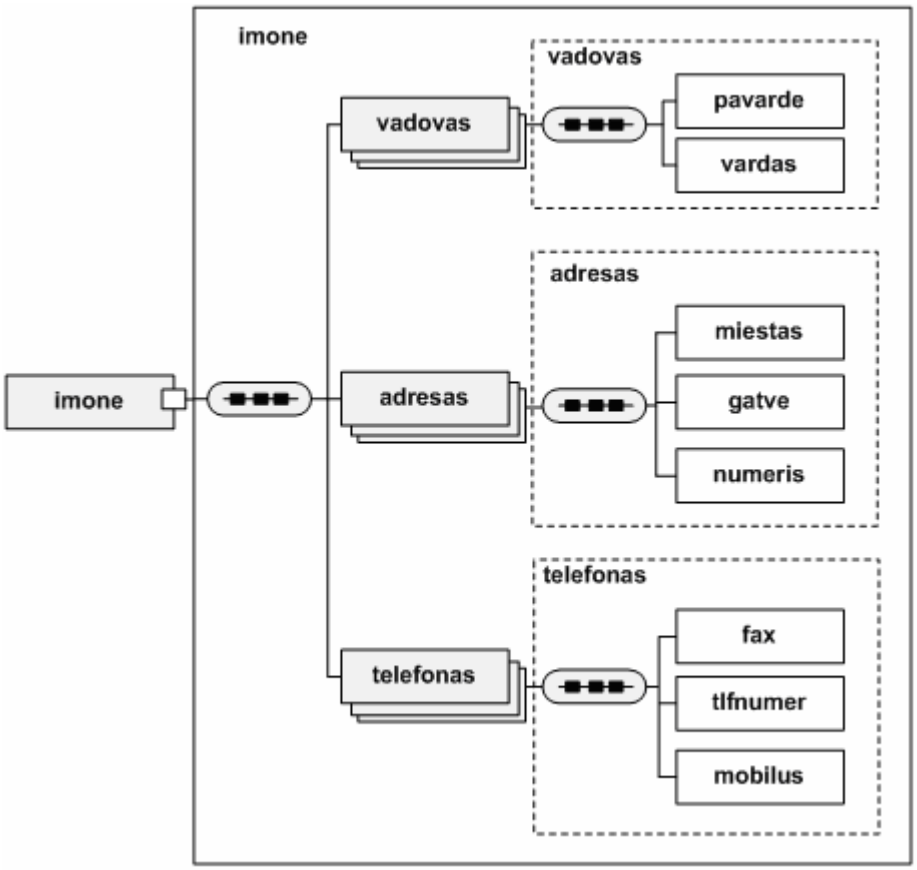
XML schemos, sugeneruotos pagal pirmą taisyklę pavyzdys pateiktas 18 pav.



18 pav. Objektų/savybių atributų transformavimas į XML elementus

4.4. XML schemos generavimas

Objektų/savybių modelio didžiausias trūkumas yra tai, kad su juo nėra suderintos programinės įrangos. Dėl to norint atlikti integraciją reikėjo panaudoti plačiai paplitusį formatą, kuris būtų suprantamas ir suderinamas su įvairia programine įranga. Šiam tikslui buvo pasirinktas XML schemų formatas ir remiantis 4.1.4 skyriuje pateikiamomis taisyklėmis buvo sugeneruota XML schema, kuri atitinka objektų/savybių modelio struktūrą, ryšius tarp objektų ir puikiai tinka duomenų integracijai. Šios XML schemos vaizdas pateikiamas 19 pav., o jos tekstinis aprašas yra pateikiamas priedų skyriuje.

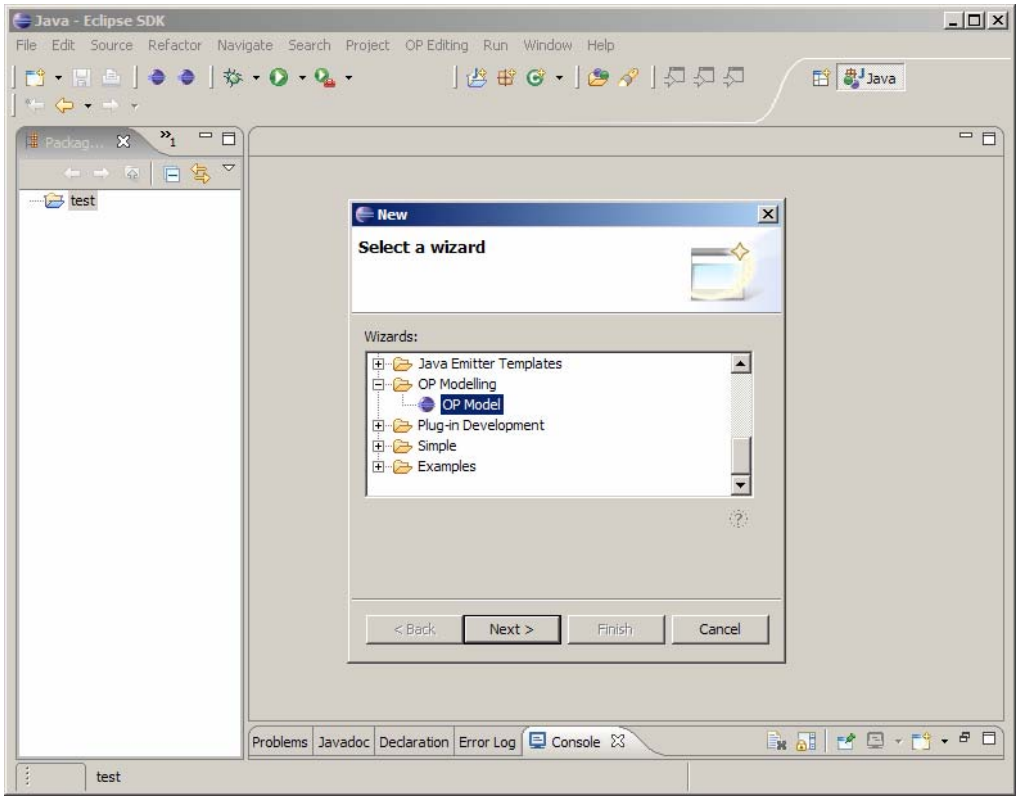


19 pav. Eksperimento XML schemos dokumento grafinis vaizdas

5. Vartotojo dokumentacija

5.1. Ontologijos redagavimas

Įdiegus ontologijų taikymą, remiantis objekto-savybių modeliu, sudaroma galimybė sudaryti ir redaguoti objektų/savybių modelio (opmodel) failus. Paleidus Eclipse yra sukuriamas naujas arba panaudojamas jau egzistuojantis projektas ir sukuriamas naujas objektų/savybių modelio failas. Taip pat galima užkrauti jau egzistuojantį modelį, pasinaudojus Eclipse *File⇒Load* funkcija.

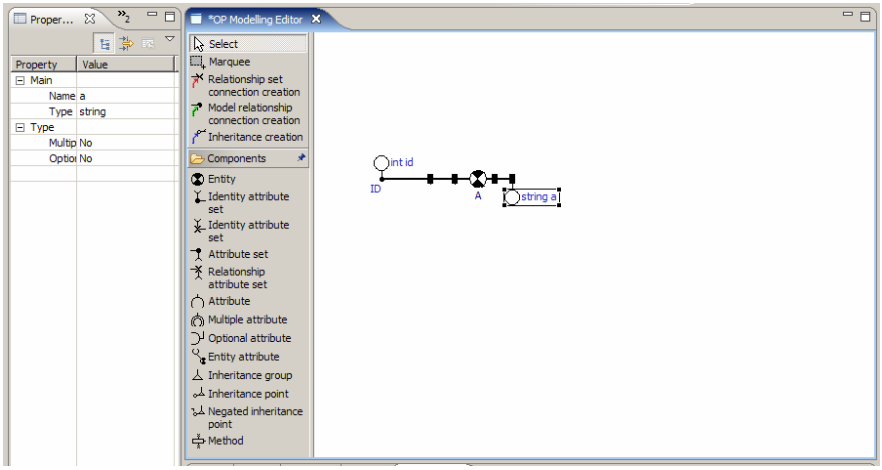


20 pav. Ontologijos objekto/savybių modelio failo sukūrimas.

Užkrovus ar sukūrus naują failą vartotojo redagavimo lauke atsiranda OP modelio darinių paletė, kurios elementus galima „užtempti“ ant darbo lauko į tinkamą vietą. Kursoriaus pavidalu vartotojui yra nurodoma ar veiksmas (ne tik įkėlimo) galimas.

Ryšys tarp tinkamos rūšies darinių yra sudaromas pasirinkus jo ikoną paletėje bei nuspaudus du kartus po vieną kartą ant atitinkamų elementų. Jei ryšys nėra galimas sudaryti dėl apribojimų, jis nesudaromas, o vartotojui indikuojama kitokiu kursoriumi.

Modelio realizacijoje yra numatyti įvairūs apribojimai, todėl, pvz. negalima įkelti kelių atributų (*angl.* Attribute) su vienodais vardais į tą pačią esybę (*angl.* Entity).



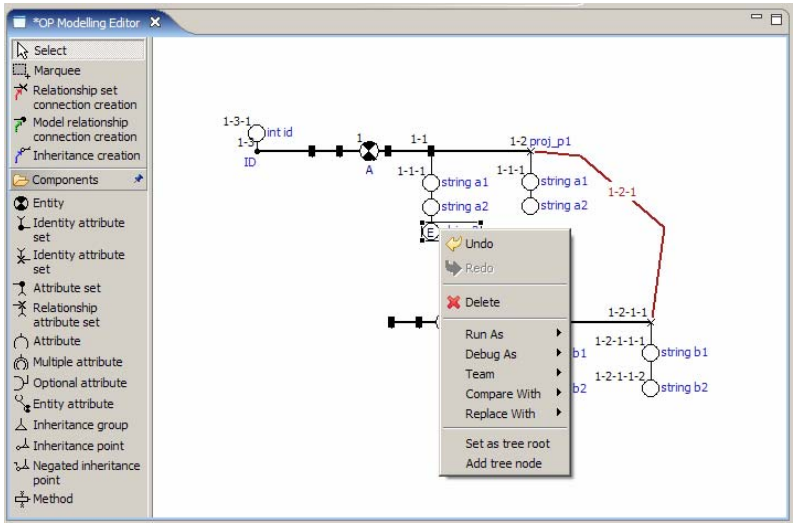
21 pav. Objektų/savybių modelio ekranas

Modelio redagavimas vykdomas įvairiais būdais. OP darinio, kuris gali būti įvardintas, pavadinimą galimą keisti vietoje, pasinaudojus F2 klavišu, arba du kart spustelėjus ant pavadinimo. Taip pat galima naudoti savybių (*angl.* Properties) langą, įjungiamą pasinaudojant meniu *Window⇒ Show⇒ View⇒ Other⇒ Basic⇒ Properties*. *Properties* lange taip pat galima keisti kitas OP darinio savybes, apart pavadinimo, priklausomai nuo jo tipo: atributo reikšmės tipą, geometrinę poziciją lange ir kt. Taip pat galimas darbas klaviatūra, naudojant kitus funkcinius klavišus, kaip: elemento trynimas su *Delete*, navigacija tarp elementų su *Up*, *Down*, *Left*, *Right* klavišais, kontekstinio meniu iškvietimas *Menu* klavišu.

Taip pat realizuotas veiksmų sugražinimas ir pakartotinis vykdymas (*angl.* undo/redo). Veiksmą galimą gražinti bei vėl įvykdyti klavišų kombinacijom *Ctrl-Z* bei *Ctrl-Y* atitinkamai.
Redaguojamas modelis gali būti išsaugotas pasinaudojus *File⇒ Save/Save As* meniu punktais.

5.2. OP modelio padengimas medžiu, subgrafo iškėlimas

Redaktoriuje realizuota galimybė OP modelio grafą padengti medžiu, kuris naudojamas apibrėžiant OP modelio subgrafą kurio kopiją norima gauti arba transformuoti į XML Schemos formatą. Šis funkcionalumas pasiekiamas naudojant OP darinio kontekstinio meniu punktus „*Set as tree root*“, „*Add tree node*“, „*Remove tree node*“ punktus.



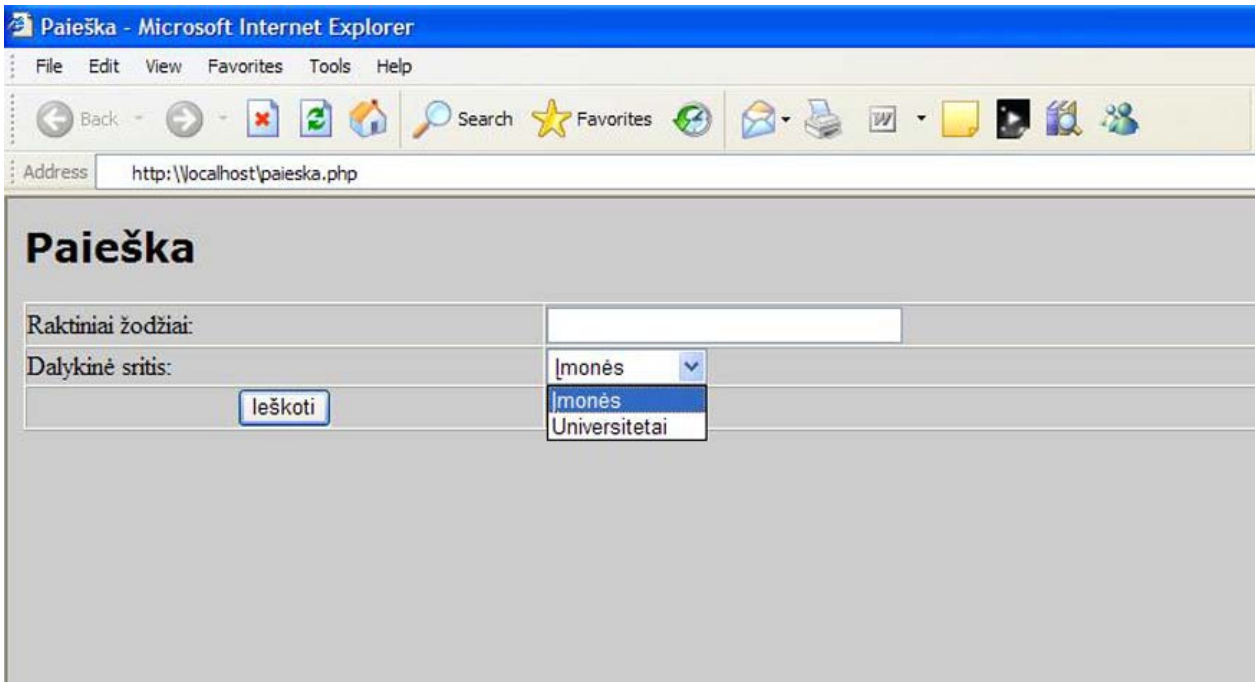
22 pav. Objektų/savybių grafo padengimas medžiu.

Sudarius objektų/savybių padengimą, galimas subgrafo generavimas pasirinkus meniu *OP Editing⇒ Extract Graph* meniu punktą. Vartotojui yra suteikiamas langas nurodyti kopijos išsaugojimo failą, redaktorius sugeneruoja pažymėto grafo kopiją, į ją įtraukdamas ir būtinus elementus (pvz. jei medyje padengtas tik modelio objekto atributas, tai kopijoje privaloma įtraukti ir OP objektą, bei atributų rinkinį kuriam padengtas atributas priklauso.

Kaip matome, kad siunčiant užklausas visai nereikalinga užklausų sistemose naudojamų duomenų bazių struktūra. Užklausoms persiųsti naudojami XML duomenys, ir siunčiami tik tie duomenys, kurių struktūra aprašyta XML schemoje.

6. Eksperimentinė dalis

Vartotojas pateikia savo užklausą raktiniais žodžiais, pasirinkdamas dalykinę sritį (ontologiją), kurioje jis nori atlikti paiešką (23 pav.).



23 pav. Vartotojo sąsaja, atliekant paiešką, naudojant ontologijas

Žemiau pateiktas kodas parodo Java panaudojimą XML parser'yje (XML parser for Java v. 2), nagrinėjant XML dokumentą.

```
import  org.w3c.dom.*;
import java.util.*;
import java.io.*;
import java.net.*;
import oracle.xml.parser. v2.* ;

public class XSLTransform
{
        public static void main (String args(1) throws Exception
        {
                DOMParser parser;
                XMLDocument xml, xsldoc, out;
                URL xslURL;
                URL xmlURL;
                try
                {
                        if (args.length !-   2)
                        {
                                //   Perduodami XSL ir XML failų vardai
                                System.err.println("Usage: javą XSLTransform xslfile xmlfile");
                                System.exit (1) ;
                        }
                        // Nagrinė jami XSL ir XML dokumentai
                        parser = new DOMParser ();
                        parser.getPreserveWhitespace(true);

                        xsXURL = createURHargs [0] ) ;
                        parser.parse(xslURL);
                        xsldoc = parser.getDooument();
```

```
                        xmlURL = createURL(args[1]);
                        parser.parse(xmlURL);
                        xml = parser.getDocument();

                        // Įdiegiama stilių lentelė
                        XSLStylesheet xsl = new XSLStylesheet (xsldoc, xalURL);
                        XSLProcessor processor = new XSLProcessor();

                        // Pateikiami klaidų, jei jos yra pranešimai
                        processor.showWarnings(true);
                        processor.setErrorStream(System.err);

                        // Apdorojamas XSL
                        DocumentFragment result = processor.processXSL(xsl, xml);

                        // Kuriamas rezultatinis dokumentas
                        out = new XMLDocument();
                        Element root = out.createElement("root");
                        out.appendChild(root);
                        root.appendChild(result);

                        // Rezultato spausdinimas
                        out.print(System.out);
                }
                catch (Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

Dalykinės srities ontologija

```
<!ELEMENT imone (vadovas+, adresas, telefonai)>
<!ELEMENT vadovas (vadovas+, pavarde, vardas,)>
<!ATTLIST pavarde type CDATA>
<!ATTLIST vardas type CDATA>
<!ELEMENT adresas (adresas+, gatve, miestas, indeksas)>
<!ATTLIST gatve type CDATA>
<!ATTLIST miestas type CDATA>
<!ATTLIST indeksas type CDATA>
<!ELEMENT telefonai (darbo?, namu)>
<!ATTLIST darbo type CDATA>
<!ATTLIST namu type CDATA>
```

Naudojant šią ontologiją galima užklausą

„Imoniu, kuriu indeksas 3001 miestas ir gatve“

Panaudojant XML SQL įrankį galima perduoti sekančią užklausą:

```
SELECT miestas, gatve FROM imone WHERE indeksas='3001'
```

Naudojant priede 5 pateiktą XML schemą, ši užklausa sugeneruoja sekančią XML schemą:

```
<?xml version="1.0"?>
<ROWSET>
        <ROW id="1">
                <MIESTAS>Kaunas</MIESTAS>
                <GATVE>Taikos pr.</GATVE>
        </ROW>
</ROWSET>
```

Naudojant XML SQL įrankį, galime ne tik suformuoti XML dokumentą, bet ir jį struktūrizuoti ir atvaizduoti mums patinkančia forma (pvz.: nurodyti maksimalų grąžinamų eilučių skaičių arba kokį XLS failą panaudoti. Sekantis Java pavyzdys siunčia užklausą į DB ir formuoja XML failą.

```
import java.sql.*;
import java.math.*;
import oracle.xml.sql.query.*;
import oracle.jdbc.*;
import oracle.jdbc.driver.*;
```

```
public class xmlquerydb {
        public static void main(String args[]) throws SQLException
        {
        String tabName = "emp"; String user = "scott/tiger";

        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        // Inicijuojama  JDBC jungtis
        Connection conn = DriverManager.getConnection("jdbc:oracle:oci8:"+user+"O");

        // Inicijuojama OracleXMLQuery užklausa

        OracleXMLQuery qry = new OracleXMLQuery(conn,"select miestas, gatve from "+tabName );

        // Generuojamo XML dokumento struktūra

        qry.setMaxRows(2);  // max grąžinamų eilučių skaičius
        qry.3etRowsetTag("ROOTDOC"); // Šakninio dokumento žymė
        qry.setRowTag("DBROW"); // elučių atskyrimo žymė
        qry.setStyleSheet("emp.xsl"); // stilių lentelė

        // Rezultatas pateikiamas simboliniame formate

        String xmistring = qry.getXMLString();

        // Spausdinti XML dokumentą

        System.out.printlnf" OUTPUT IS:\n"+xmlstring);
        }
}
```

Pateikto kodo rezultatas yra XML failas (kaip ir buvo nurodyta pirmos dvi eilutes is lentelės *imone*).

```
<?xml version="1.0"?>
<ROOTDOC>
<DBROW id="l">
        <MIESTAS>Kaunas</MIESTAS>
        <GATVE>Taikos pr.</GATVE>
</DBROW>
<DBROW id="2">
        <MIESTAS>Kaunas</MIESTAS>
        <GATVE>Pramones pr.</GATVE>
</DBROW>
</ROOTDOC>
```

Transakcinis serveris naudoja XSQL servlet'us. Jie leidžia vykdyti SQL užklausas, bei pateikti rezultatus XML formatu. Šis naudoja Java Servlet'us. Įėjimo duomenys yra XML failas, turintis informaciją apie SQL užkausą. Jis naudoja XML Java parser'į bei XML SQL įrankį atliekant visas operacijas.

XSQL servelt'ą galima paleisti bet kuriame WEB serveryje, kuris palaiko Java servlet'us. Sekantis pavyzdys paaiškina, duomenų srautus nuo kliento iki servlet'o ir vėl atgal iki kliento:

1. Vartotojas naršyklėje surenka reikiamą URL adresą. Adresas nurodo reikiamą XSQL failą (*.xsql) ir, jei reikia, parametrus. Taip pat XSQL servlet'ą vartotojas gali iškviesti ir komandinės eilutės pagalba.
2. Servlet'as perduoda XSQL failą XML Java parser'iui, kuris apdoroja gautą užklausą ir sukuria API XML turinio priėjimui.
3. Puslapių procesoriaus komponentas naudoja API XML parametrų bei SQL sakinių perdavimui XML SQL įrankiui.
4. XML SQL įrankis siunčia SQL užklausas į Qracle 8i DB kuri gražina norimus rezultatus.
5. XML SQL įrankis grąžina užklausos rezultatus XSLT procesoriui XML formatu.

6. Jei reikia, užklausos rezultatai yra transformuojami XSLT procesoriaus naudojant specialų XSL stilių failą. Duomenys gali būti transformuoti į bet kokį reikiamą XML, HTML ar bet kokį kitą formatą.
7. XSLT procesorius perduoda suformuotą dokumentą klientui į naršyklę.

Pateikiame SQL servlet'o pavyzdį, naudojamą šiame pavyzdyje:

XSQL faile aprašyta užklausa, išrenkanti vadovus iš *imones* lentelės

```
<?xml version="1.0"?>
<?xml-stylesheet type ="text/xsl" href="rowcol.xsl"?>
<query connection="demo" find="%" sort="PAVARDE" null-indicator="yes" >
SELECT *  FROM imones ORDER BY {<Ssort}
</query»
```

XSQL faile taip pat nurodyta, kad grąžinami rezultatai turi būti apdoroti naudojant XSL stilių failą rowcol.xsl.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.orq/1999/XSL/Transform"> <xsl:tempiate match="/"> <html>
<body class= "page">
<xsl:apply-templates/>
</body>
</html>
</xsl:template>
<xsl:tempiate match="ROWSET">
        <center>
        <table border="0" cellpadding="4">
        <xsl:choose>
                <xsl:when test = "ROW":>
                <!—pateikiami: row[l] stulpeliai +-->
                        <xsl:for-each select="ROW[1]">
                                <tr>
                                        <xsl:for-each select="*">
                                                <th align="left">
                                                        <xsl:attribute name="class">
                                                        </xsl:attribute>
                                                        <xsl:value-of select="name(.)"/>
                                                </th>

                                        </xsl:for-each>
                                </tr>
                        </xsl:for-each>
                <xsl:apply-templates/>
        </xsl:when>
                <xsl:otherwise>
        <tr>
        <td>No Matches</tdx>
        </tr>
        </xsl:otherwise>
        </xsl:choose>
        </table>
        </center>
</xsl:template>
<!—stulpeliai ir eilutes +-->
<xsl:tempiate match="ROW">
        <tr>
                <xsl:attribute name = "class" >
```

```
                </xsl:attribute>
                <xsl:for-each select="*">
                        <td>
                                <xsl:attribute name="class">
                                </xsl:attribute>
                                <xsl:apply-templates select='.'/>
                        </td>
                </xal:for-each>
        </tr>
</xsl:tempiate>
</xsl:stylesheet>
```

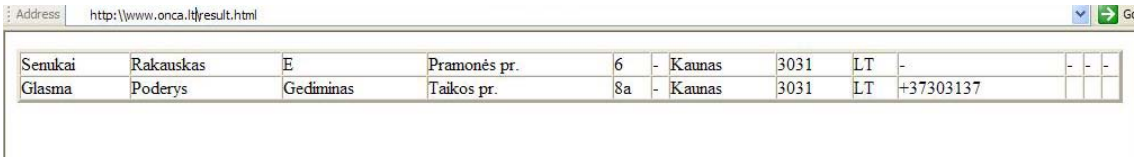
XSQL faile aprašyta užklausa, išrenkant informaciją iš imones lentelės.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="rowcol.xsl"?>
        <query connection="demo"
                find="%"
                sort="Pavarde"
                nuli-indicator="yes" >
                SELECT *  FROM imone WHERE indeksa = '0031' ORDER BY (@Sort)
        </query>
```

XSQL faile taip pat nurodyta, kad grąžinami rezultatai turi būti apdoroti, naudojant XSL stilių lentelę *rowcol.xsl*:

```
<xsl: stylesheet xmlns : xsl="http ://www.w3.org/1999/XSL/Transform" >
        <xsl:tempiate match="/">
                <html>
                        <body class="page" >
                                <xsl:apply-templates/>
                        </body>
                </html>
        </xsl:template>
```

XSQL sevletas grąžino sekančius rezultatus HTML lentelės pavidalu



Senukai	Rakauskas	E	Pramonės pr.	6	-	Kaunas	3031	LT	-	-	-
Glasma	Poderys	Gediminas	Taikos pr.	8a	-	Kaunas	3031	LT	+37303137		

Išvados

Atlikus literatūros analizę, galime pasakyti, kad žinių portalų, orientuotų į informacijos paiešką internete, ontologijų ypatybė yra tai, kad juose be tradicinių dalykinės srities aprašų yra ir tinklinių resursų aprašai. Tokie aprašai leidžia duomenis paskirstytose duomenų bazėse, turinčiose skirtingus informacijos vaizdavimo būdus, apjungti ir vykdyti paiešką metaduomenų lygmenyje.

XML schema, jos struktūra ir sandara yra labai artima reliacinių duomenų bazių struktūrai. Todėl norint keistis duomenimis, reikia sukurti XML schemą, kuri leistų perduoti duomenis į bet kurią sistemą.

Tam kad sistema galėtų keistis informacija, turi būti naudojama kalba, kuri tą informaciją aprašytų. Bendraudami žmonės naudoja žodžius ir jų paaiškinimus, kurie leidžia keistis informacija. Bendravimo metu informacija gaunama iš žodžių reikšmių ir jų pateikimo struktūros. Kadangi kompiuteryje nėra žodžių ir simbolių susiejimo su semantika mechanizmo, reikia žodyno, kuris kompiuteriui leistų nustatyti semantiškai lygias išraiškas. Šį vaidmenį atlieka ontologijos, kurios remiasi objektų/savybių semantiniu modeliu.

Šiame darbe atliktas ontologijų kūrimo mechanizmas. Ontologija naudojama transformuojant vartotojo užklausas į XML schemas. Sudaryta tokio transformavimo algoritminė schema.

Atsižvelgiant į objektų/savybių modelio darinių savybes buvo sukurtos tokio modelio komponentų transformavimo taisyklės į XML schemų komponentus, ir praktiškai pritaikyta atliekant eksperimentinę dalį. Transformacijos buvo atskirai taikomos skirtingiems objektų/savybių pasireiškimo atvejams, buvo sukurtos visas objekto/savybių modelio transformacijas atvaizduojančios XML schemos bei jų grafinis vaizdas

Literatūra

1. W3C, *Extensible Markup Language (XML) 1.0 (Third Edition)*. Interneto prieiga: http://www.w3.org/TR/2004/REC-xml-20040204/, 2004.
2. ISO/IEC, *JTC1/SC34 (Standardization in the field of document structures, languages and related facilities for the description and processing of compound and hypermedia documents)*. Interneto prieiga: http://y12web2.y12.doe.gov/sgml/sc34/sc34oldhome.htm, 2005.
3. W3C, *HTML 4.0 Specification*. Interneto prieiga: http://www.w3.org/TR/WD-html40/, 1998.
4. W3C, *XML Schema Specifications and Development*. Interneto prieiga: http://www.w3.org/XML/Schema#dev, 2004.
5. PAPA, John. *XML Features in SQL Server 2000*. Interneto prieiga: http://msdn.microsoft.com/library/default.asp?url=/msdnmag/issues/0800/sql2000/toc.asp.
6. Eclipse, Graphical Editing Framework (GEF). Interneto prieiga: http://www.eclipse.org/gef/ , 2005.
7. Eclipse, XML Schema Infoset Model (XSD). Interneto prieiga: http://www.eclipse.org/xsd/ , 2005.
8. E. van der Vlist. XML Schema, 2002, 1st edition, 13-16 psl.
9. Ambrazevičius A., Jasiukevičius A., Paradauskas B. Komunikacinių kilpų informacijos išsaugojimo kriterijus. Informacijos mokslai. 2003, Nr. 24, 100-101psl.
10. Nemuraitė L., Paradauskas B., Salelionis L.. Extended Communicative Action Loop for Integration Of New Functional Requirements. Informacinės technologijos ir valdymas, 2002 Nr. 2 (23) 20-23 psl.
11. Sowa F.J. Top-level ontological categories. International Journal of Human-
12. Computer Studies, 43:669–685, 1995.
13. Kirk T., Levy A., Sagiv Y. and Srivastava D.. The Information Manifold. In AAAI Spring Symposium on Information Gathering, 1995.
14. Genesereth M., Keller A. and Duschka O. Infomaster: An information integrationsystem. SIGMOD Record, 26(2):539–542, May 1997.
15. Li C., Yerneni R., Vassalos V., Garcia-Molina H., Papakonstantinou Y., Ullman J. and Valiveti M. Capability based mediation in TSIMMIS. In Proceedings of the ACM SIGMOD Conference on Management of Data, pages 564–566, June 1998.
16. Uniform Code Council Inc. SIL - Standard Interchange Language. Technical report, January 1999.

17. McIlraith A.S., Son C.T., and Zeng H. Semantic Web Services. [interaktyvus], 2007, [žiūrėta 2008 11 20]. Prieiga per internetą: http://ksl.stanford.edu/people/sam/ieee01.pdf.
18. Lawrence R., Barker K.. Automatic Integration of Relational Database Schemas
19. Sturm J. Developing XML Solutions, 2000
20. Doan A., Halevy Y.A. Semantic Integration Research in the Database Community. [interaktyvus], 200, [žiūrėta 2008 11 20]. Prieiga per internetą: http://pages.cs.wisc.edu/~anhai/papers/si-survey-db-community.pdf
21. Magkanaraki A. Ontology Storage and Querying. [interaktyvus], 2002, [žiūrėta 2008 11 20]. Prieiga per internetą: http://xml.coverpages.org/MagkanarakiOnt.pdf.
22. Stack M. Full Text Search of Web Archive Collections. [interaktyvus], 2005, [žiūrėta 2008 11 20]. Prieiga per internetą: http://iwaw.europarchive.org/05/stack3.pdf.
23. Sakamoto Y, Ishikawa T., Ishikava M. Concept and Structure of Semantic Markers. [interaktyvus], 2003, [žiūrėta 2008 11 20]. Prieiga per internetą: http://www.aclweb.org/anthology-new/C/C86/C86-1003.pdf.

Priedai

1 priedas. Sutrumpinimų sąrašas

DB – duomenų bazė.

DBVS – duomenų bazių valdymo sistema.

HTTP - hiperteksto perdavimo protokolas.

UML – unifikuota modeliavimo kalba.

IS – informacinė sistema.

PĮ – programinė įranga.

JAVA – Programavimo technologija.

IE – Interneto naršykle Internet Explorer.

OS – operacinės sistema.

PĮ – programinė įranga.

2 priedas. Iliustracijų sąrašas

3 priedas. Lentelių sąrašas

4 priedas. Straipsnis

Informacijos valdymo metodų analizė ir sprendimas paieškos sistemose

Marijonas Nekroševičius
KTU, Informatikos fakultetas,Kompiuterių katedra
Darbo vadovas: **lekt. Dr**. ***Audronė Janavičiūtė***

Įvadas

Skirtingose duomenų bazėse saugomų duomenų kiekis pastoviai auga. Dauguma šių duomenų bazių yra pasiekiamos per Internetą arba vidinį tinklą. Tačiau netgi patį Internetą galima laikyti didele paskirstyta heterogenine duomenų baze, kuria dažniausiai naudojasi paprasti vartotojai. Atliekant vieną duomenų transakciją galima gauti, siųsti ir apdoroti įvairiems duomenų šaltiniams skirtus duomenis. Taip pat vienoje transakcijoje apdorojai duomenys gali būti skirtingose duomenų bazėse, kurios yra skirtingose operacinėse sistemose.

Mūsų tikslas yra apibrėžti tam tikrus standartus, kuriais remiantis vyksta duomenų perdavimas tarp skirtingų duomenų šaltinių, į ką atsižvelgiant yra kuriamos duomenų priėmimo ir eksportavimo schemos. Norėčiau pateikti keletą su šia tema susijusių faktų:
Paskirstytas duomenų bazes siūlo daugelis DB kūrėjų, tačiau dažniausiai jos būna suderinamos tik su jos gamintojo programine įranga, nors vienos transakcijos metu turi būti keičiamasi duomenimis tarp kelių skirtingų duomenų šaltinių.

Sunkumai su kuriais susiduriama vystantis hibridinėms duomenų bazėms

Informacinėms technologijoms vystantis atsirandantys nauji technologiniai reikalavimai dažniausiai įtakoja papildomų duomenų struktūrų poreikį. Norint įvertinti, ar jau esantys informacijos resursai yra pakankami patenkinti naujus funkcinius reikalavimus, reikia atsižvelgti į kiekvieną iš tų reikalavimų. Naujos informacinės technologijos skatina kurti naujas duomenų bazes, tačiau palikimo duomenų bazėse esantys duomenys verčia naudoti naujas sistemas sujungiant jas su palikimo sistemomis. Palikimo sistemos dar naudingos ir tuo, kad jose gali būti likę naudingų taikomųjų programų, ir jų nereiktų keisti kompiuterizuotos informacinės sistemos perprojektavimo procese.[4]

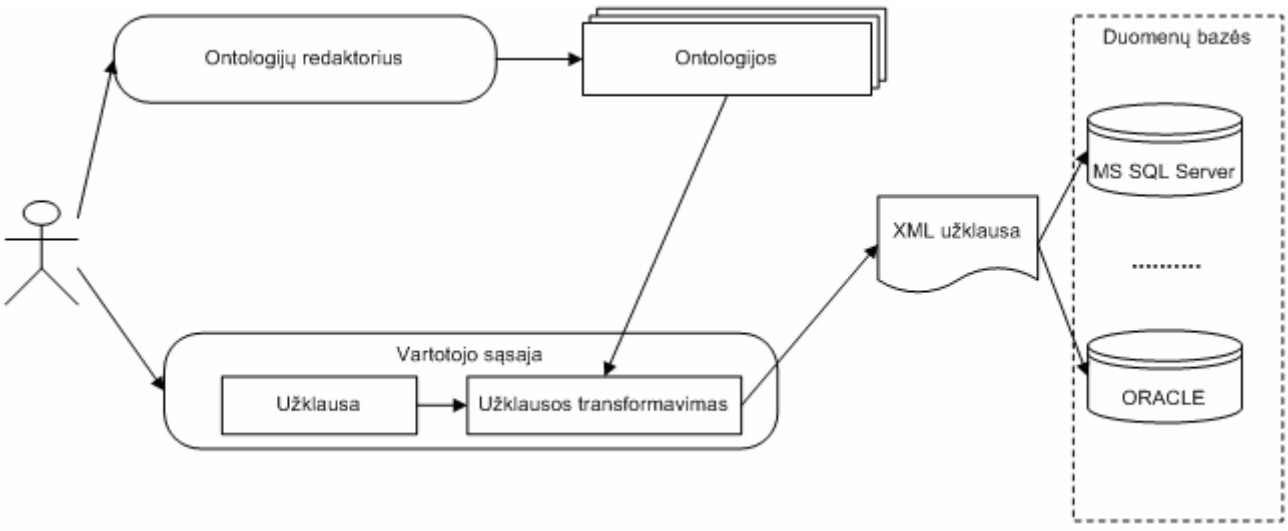
Kelių duomenų bazių valdymo sistemų arba vienos DBVS sistemos kelių versijų, kurios palaiko vienodus arba skirtingus duomenų modelius, naudojimas, privertė nustatyti nehomogeninių duomenų bazių kūrimo metodus. Nehomogeninių komponentų integracija tapo pagrindiniu informacinių sistemų perprojektavimo uždaviniu. Informacinių sistemų projektavime nauji arba kopijuoti duomenų bazės komponentai gali būti nustatyti arba sukurti remiantis

naujais funkciniais reikalavimais. Nehomogeninių duomenų bazių su reliaciniais ir objektiškai orientuotais komponentais gali būti vadinamas hibridinių duomenų bazių kūrimu.

Galima būtų išskirti tokius hibridinių duomenų bazių kūrimo etapus:

1. Transakcijų vaizdavimas atsižvelgiant į naujus funkcinius reikalavimus.
2. Duomenų objektų ir jų atributų nustatymas ir normalizuotos duomenų poschemės sukūrimas.
3. Šios poschemės integracija į bendrą duomenų schemą.
    - Duomenų schemą ir naują poschemę siejančių sąryšių nustatymas
    - Struktūrinių priklausomybių nustatymas
    - Priklausomybių suderinamumo ir objektinių tipų konfliktų analizė ir šalinimas
    - Objektų blokinio nustatymas ir globalios schemos papildymas naujais duomenų objektais
    - Bendro blokinių vaizdinio kūrimas

Jei nauja sistema įdiegta kartu su objektine duomenų baze o palikimo sistema yra paremta reliacinės duomenų bazės sistema, arba diegiamos sistemos yra skirtingų tipų, kaip Oracle ir MS SQL Server, tuomet gali reikti transformacijų.



1 pav. Oracle ir MS SQL Server duomenų bazių integracija panaudojant XML transformacijas.

Kaip matome 1 pav. norint keistis duomenis tarp heterogeninių duomenų bazių reikia sukurti apibendrintą struktūrą, kuri apimtų visų sistemų duomenis, kurie gali būti tarpusavyje visiškai nesuderinami. Tokiu atveju kiekvienai sistemai reiktų kurti programinius įrankius arba paketus, kurie palaikytų prisijungimą prie kitų sistemos duomenų bazių. Tai yra tikrai nepatogu ir sudėtinga. Dėl to duomenims tarp nehomogeninių duomenų bazių perduoti patogiausia būtų naudoti tokį

duomenų tipą, kuris labiausiai atitiktų sistemos duomenų bazių duomenis ir tiesiogiai nepriklausytų nuo atitinkamos sistemos ar duomenų bazės. Šiuo atveju patogiausia būtų naudoti XML.

XML programavimo kalba

- XML skirta keistis duomenimis
- XML skirta keistis finansine informacija, t.y. yra naudojama daugumoje B2B taikomųjų programų.
- XML gali būti naudojama duomenims paskirstyti (share)
- XML gali būti skirta duomenims saugoti
- XML gali padaryti duomenis labiau prieinamus

Remiantis aukščiau išvardintais XML požymiais galima daryti išvadą, jog XML kalba labiausiai tinka keistis duomenimis tarp heterogeninių duomenų bazių. Tačiau šis procesas nėra toks paprastas, nes kaip matome 1 pav. reikia atsižvelgti į įvairių sistemų funkcinius reikalavimus. Reikia sukurti bendrą šabloną, kuriuo remiantis būtų įmanoma atlikti keitimąsi duomenimis tarp įvairių sistemų. Naudojant XML, šiuo šablonu patogiausia naudoti XML schemas [2].

XML schemų paskirtis

XML schema nusako XML dokumento struktūrą. XML schemos skirtos:

- apibrėžia dokumente naudojamus elementus
- apibrėžia dokumente naudojamus atributus
- apibrėžia kurie elementai yra elementai – vaikai
- apibrėžia elementų – vaikų tvarką
- apibrėžia elementų – vaikų kiekį
- apibrėžia ar elementas yra tuščias ar gali saugoti tekstinę informaciją
- apibrėžia elementų duomenų tipus ir atributus
- apibrėžia standartines ir fiksuotas elementų ir atributų reikšmes
- apibrėžia ryšius tarp elementų [1], [5]

Tačiau dėl nevisiško XML schemų ir reliacinių bazių struktūros atitikimo iškyla konvertavimo problemos:

- **Reikia atskirti schemų palyginimą nuo schemų konvertavimo.**

Yra labai svarbu atskirti šiame skyriuje nagrinėjamą schemų konvertavimo uždavinį nuo labai į jį panašaus schemų palyginimo uždavinio. Kai yra duotos pradinė schema s1 ir galutinė schema t1, schemų palyginimo uždavinys suranda priklausomybes, siejančias s1 elementus su t1 elementais. Schemos konvertavimo atveju yra duota tik pradinė schema s1 ir reikia rasti jai identišką galutinę schemą t1. Schemų palyginimo problema dažniausiai egzistuoja tarp tam pačiam duomenų modeliui priklausančių schemų, tuo tarpu schemų konvertavimo problema taikoma skirtingiems duomenų modeliams priklausančioms schemoms.

- **Tarp XML ir nereliacinių duomenų modelių**

Yra sukurta nemažai konvertavimo technikų tarp XML ir nereliacinių duomenų modelių, tačiau dėl skirtumų tarp ER ir XML modelių, jas priklausomai nuo konvertuojamų objektų reikia nuolat redaguoti. Dažniausiai šios technikos buvo taip kuriamos specialiems atvejams ir taikomosioms programoms, kad juos pritaikyti XML ir reliacinių modelių schemoms konvertuoti yra nenaudinga.

- **Iš XML į reliacinius duomenų modelius**

Pastaruoju metu XML ir reliacinių duomenų modelių schemoms konvertuoti buvo pasiūlyta net keletas algoritmų. Vienas iš jų yra STORED[3]. Jis remiasi duomenų išgavimo būdu ir sukuria DTD, kuri suteikia daugiau informacijos nei buvo turėta anksčiau ir panaudojant tą DTD yra atliekamas konvertavimas iš XML į reliacinį formatą.

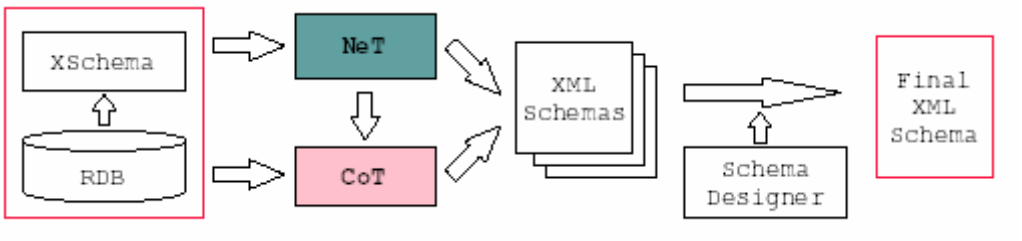
- **Iš reliacinių duomenų modelių į XML**

Yra sukurtas keletas tokio konvertavimo įrankių: XML Extender, XML-DBMS, SilkRoute, ir XPERANTO [3]. Visiems paminėtiems įrankiams reikia, kad vartotojas pats nurodytų reliacinės schemos priskyrimą XML schemai (mapping).

Toliau paminėsime 3 algoritmus, kurie ne tik atlieka konvertavimą, bet taip pat konvertuoja ir originalios sistemos semantiką, t.y. konvertuoja ne tik pačias lenteles bet ir ryšius tarp tų lentelių.

1. **CPI** (Constraints-preserving Inlining Algorithm): identifikuoja įvairius semantinius ryšius originalioje XML schemoje ir išsaugo juos perrašydamas į gaunamą reliacinę schemą.
2. **NeT** (Nesting-based Translation Algorithm): taikydamas lizdinį operatorių, iš reliacinės schemos taip sukuria lizdinę struktūrą, kad XML schema tampa hierarchinė.
3. **CoT** (Constraints-based Translation Algorithm): nors NeT taikydamas lizdinį metodą ir nustato paslėptas duomenų charakteristikas, jis pritaikymas tik vienai lentelei. Todėl negalima nustatyti

bendro DB vaizdinio, kur yra sujungta daug lentelių. Tam kad išspręstų šią problemą, CoT apdoroja priklausomybes transliacijos metu ir sujungia daug tarpusavyje sujungtų lentelių į vientisą ir hierarchinę tėvo-vaiko struktūros XML schemą. [3]



2 pav. Reliacinių schemų konvertavimo į XML schemas algoritmų pritaikymas

Išvados

Norint integruoti nehomogeninių sistemų duomenų bazes visų pirma reikia sukurti tas sistemas sujungiantį bendrą objektų blokinį. Tam patogiausia naudoti XML schemas, kurios gaunamos konvertuojant nehomogeninių sistemų reliacines schemas (šiuo atveju yra Oracle – XML ir MS SQL Server – XML transformacijos) . XML schemos yra labiausiai vartotinos dėl savo nepriklausomumo nuo nehomogeninės sistemos, taip pat nuo naudojamos operacinės sistemos. Yra naudojami šie schemų konvertavimo algoritmai: CPI, NeT, CoT. Visi šie algoritmai konvertavimo eigoje išsaugo ne tik reliacinių schemų fizinę, bet ir semantinę struktūras.

Literatūra:

[1] Jake Sturm Developing XML Solutions, 2000
[2] Graeme Malcolm Programming Microsoft SQL Server 2000 with XML
[3] D.Lee, M.Mani, W.W. Chu Schema Conversion Methods between XML and Relational Models, 2003
[4] Informacinės technologijos ir valdymas Nr.2(23), 2002, p. 23-24
[5] Interneto prieiga: http://www.w3schools.com/schema/schema_intro.asp

Heterogeneous databases integration using XML formats

The main problem in heterogeneous database integration is data incompatibility in different databases. XML is perfect solution in data exchange between different databases as it is independent from OS, applications or hardware. To implement XML in data exchange XML must be created corresponding to the databases. To create the exact mapping of these databases XML schema must be created including all semantics and tables from databases. XML schemas are being created using one of 3 algorithms: CPI, NeT, and CoT.

5 priedas. Eksperimento XML schemos dokumento aprašymas XML programavimo kalba

```
<?xml version="1.0" encoding="windows-1257" ?>
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
  <xs:element name="imone" type="imone" />
  </xs:element>
  <xs:element name="vadovas" type="vadovas" maxOccurs="unbounded" sql:field="vadovas" />
  <xs:element name="adresas" type="adresas" maxOccurs="unbounded" sql:field="adresas" />
  <xs:element name="telefonas" type="telefonas" maxOccurs="unbounded" sql:field="telefonas" />
- <xs:complexType name="vadovas">
- <xs:sequence>
  <xs:element name="vardas" type="xs:string" />
  <xs:element name="pavarde" type="xs:integer" />
  </xs:sequence>
  </xs:complexType>
- <xs:complexType name="adresas">
- <xs:sequence>
  <xs:element name="gatve" type="xs:string" />
  <xs:element name="namo_nr" type="xs:integer" />
  <xs:element name="buto_nr" type="xs:integer" />
  <xs:element name="miestas" type="xs:string" />
  <xs:element name="indeksas" type="xs:integer" />
  <xs:element name="valstybe" type="xs:string" />
  </xs:sequence>
  </xs:complexType>
- <xs:complexType name="telefonas">
- <xs:sequence>
  <xs:element name="namu" type="telnr" maxOccurs="unbounded" />
  <xs:element name="darbo" type="telnr" maxOccurs="unbounded" />
  <xs:element name="mobilus" type="telnr" maxOccurs="unbounded" />
  </xs:sequence>
  </xs:complexType>
- <xs:simpleType name="telnr">
- <xs:restriction base="xs:string">
  <xs:length value="8" />
  </xs:simpleType>
  </xs:sequence>
  </xs:schema>
```