Sissejuhatus semantilisse veebi, 4: Reaalne süsteemi loomine ontoloogiate abil

21 years ago- 22.04.2005 By AM

(Arvutimaailm 4/05)

Autor: **Mario Peterson**

Seekord räägin ontoloogiate kasutamisest ka reaalse süsteemi loomisel. Ühtlasi on see viimane ja kokkuvõtlik artikkel semnatilise veebi kohta.

Loodava süsteemi kirjeldus

Probleemi püstitus

Vaatleme situatsiooni, mille kontekstis me süsteemi loome. Oletame, et:

• meil on palju riistvara müüvaid firmasid (Riistvara kaupmeees1 ..n);

• igal kaupmehel on koduleht;

• iga firma avaldab HTML-kujul oma toodete hinnad;

• kõik kaupmehed müüvad samaväärseid kaupu, kuid erineva hinnaga;

• kaupmeeste arv (n) on suur.

Antud olukorras on väga raske saada ülevaadet müüdavatest kaupadest. See protsess on väga aeganõudev. Samuti ei ole võimalik luua sellise mudeli puhul tarkvaralist agenti, mis hinnakirjad automaatselt läbi käiks ja mingi kokkuvõtte teeks. Seega on paratamatu, et kaupmehed peavad oma hinnakirju avaldama ka mingil muul kujul, kui HTML-is.

Defineerime nõuded, millele loodav süsteem peaks vastama.

• Platvormi sõltumatus: pakutav lahendus ei tohiks sõltuda kaupmehe ega agendi poolt kasutatavast operatsioonisüsteemist ega rakendustest.

• Universaalsus: kaupmehe poolt pakutav hinnakiri peab olema arusaadav kõikidele agentidele.

• Laiendatavus: hinnakirja struktuur peab olema vabalt muudetav ilma, et häiritaks vana struktuuriga arvestavate süsteemide tööd.

Loodav süsteem

Kuna eeldasime, et igal kaupmehel on olemas koduleheküIg ja mingil kujul hinnakiri, siis tuleks esimese sammuna viia see hinnakiri RDF/ XML kujule. Minimaalselt peaks seal iga toote kohta sisalduma omadused: nimetus, hind, URL (aadress, kus on toote pikem kirjeldus).

Kui on täidetud eelpool kirjeldatud eeldused, saab iga kasutaja luua ise või kasutada mõnda tarkvaraagenti, mis neid andmeid lugeda oskab.

Meie loome oma näites veebilehe, mis käitub tarkvaraagendina - pärib infot erinevate kaupmeeste veebilehtedelt ja näitab seda vastavalt kasutaja päringutele.

Agent pärib infot kaupmeeste käest RDF-kujul. Kasutaja saab veebiliidese kaudu teha päringuid agendile. Agent teostab päringu vastavalt kasutaja poolt sisestatud päringule. Agent saab kasutada päringu tegemisel kõiki eelnevalt defineeritud kaupmeeste hinnakirju.

Ontoloogia kirjeldamine

Kuna valisime ontoloogia kirjeldamise keeleks RDF-i, siis tuleks ka tööriistade valimisel lähtuda sellest, et need toetaksid RDF-i. Maailmas on sadu ontoloogia loomise programme, kuid enamus neist on kommertsiaalsed ja suhteliselt spetsiifilised. Väheste programmide kohta on olemas korralik näidetega varustatud kasutajajuhend ja ühinenud kasutajaskond, kelle käest probleemide korral abi küsida.

Protege

Ontoloogia kirjeldamiseks kasutame visuaalset tööriista nimega Protege. Seda kasutatakse peamiselt uurimis- ja tööstusprojektides.

Protege abil on võimalik konstrueerida ontoloogiaid, luua kohandatud vorme andmete sisestamiseks ning sisestada andmeid.

Samuti saab Protege abil luua ontoloogia, sisestada eksemplare ja kontrollida ontoloogia õigsust. Väljundina on võimalik saada ühe võimalusena XML /RDF kujul fail. Lisandprogrammina on võimalik lisada ka tõestaja ja päringumootor. Tõestajana kasutasime programmi RACER.

Arutleja RACER

RACER (Renamed ABox and Concept Expression Reasoner) on teadmiste taasesitamise süsteem, mis teostab optimeeritud tõestusi väljendusrikka kirjeldusloogika jaoks. RACER pakub ka vahendeid algebraliseks arutluseks, kaasarvatud konkreetsed meetmed,käsitlemaks:

• min/max piiranguid reaalarvude hulgal;

• lineaar-polünomiaalseid võrrandeid reaalaarvude hulgal või järjestussuhteid põhiarvude hulgal;

• mittelineaarseid polünomiaalseid võrrandeid kompleksarvude hulgal;

• stringide võrduseid ja lahknevusi.

RACER toetab üldiste terminoloogiliste aksioomide kirjeldamist. TBox võib sisaldada üldiste kontseptsioonide kaasarvamist. RACER suudab käsitleda kontseptsioonide mitmeseid ja isegi tsüklilisi definitsioone.

Antud lahenduses kasutasime RACER-it Protege lisandprogrammina. Protege suhtleb RACER-i serveriga kasutades HTTP protokolli ja sõnumid on kodeeritud kasutades XML -i.

SiRPAC

Lihtsamat ontoloogiat on võimalik ka luua suvalise tekstiredaktoriga. SiRPAC on RDF-i valideerimise teenus, mis baseerub RDF-i parseril Another RDF Parser (ARP). Hetkel kasutatakse versiooni 2-alpha-1. ARF loodi Jeremy Carrolli poolt Hewlett Packardi laborites.

Valideerimine tähendab antud kontekstis süntaksi kontrolli ja viidete kontrollimist. Tulemusena on võimalik näha RDF-dokumendis kirjeldatud tripletid (subjekt, predikaat, objekt) tabelina. Samuti on võimalik lasta genereerida etteantud ontoloogia graafiline esitus, mis annab hea ülevaate. Sisendina on võimalik anda ette URL, millelt süsteem saab RDF-dokumendi lugeda või kopeerida RDF-i sisu veebilehel toodud sisendkasti.

RDF-i genereerimine kaupmehe poolt

Eelpool kirjeldatud ontoloogia kirjeldamise vahendid sobivad eelkõige ontoloogiate väljatöötamiseks ja valideerimiseks. Igapäevaselt kaupmehe poolt genereeritava RDF-kujul hinnakirja puhul on juba struktuur paigas ja hinnakirja genereerimiseks on vaja programmijuppi, mis viib kaupmehe hinnakirja etteantud

ontoloogia nõuetele vastavasse formaati.

RDF-kujul hinnakirja genereerimiseks on kaks moodust:

• failina: kaupmehe server muudab failisüsteemis olevat RDF faili peale iga muudatust hinnakirjas;

• teenusena: kaupmehe server genereerib iga pöördumise korral hinnakirja (faili failisüsteemi tegelikult ei looda).

Kõige optimaalsem on genereerida peale iga muudatust hinnakirjas fail, sest suure päringute arvu korral on selline lahendus vähem ressursinõudlik. Mahukama hinnakirja puhul on faili lahendus kindlasti ka kiirem.

Päringute tegemine

RDQL

Päringute tegemiseks kasutame RDQL-i (Query Language for RDF). Selle abil saab teha RDF dokumentidele SQL-i laadseid päringuid. Päringu käske on oluliselt vähem kui SQL-i standard ette näeb, kuid enimkasutatavad SQL käsud on olemas ka RDQL-is:

• SELECT - listing päritavatest muutujatest;

• FROM - nimekiri RDF-dokumentidest;

• WHERE - tingimused;

• AND - filtreerivad väljendid;

• USING - prefiksi deklaratsioon.

Näidispäring süsteemist

Allpool olevast näites näeme, kuidas reaalses süsteemis päringut esitatakse. Päringus küsitakse hinnakiri riistvara kaupmehe käest, kelle hinnakiri asub Beestingu koduleheküljel ja on kõigile vabalt kättesaadav.

Päringu vastuseks saadakse kõikide toodete andmed (nimetus, hind ja URI, millelt on võimalik antud toote kohta saada lähemat infot), millel on hind.

Realisatsioon

Lahendus on programmeeritud PHP-s (PHP: Hypertext Preprocessor) ja kasutajaliideseid HTML-i ja CSS-i kasutades. Kuna RDQL ei ole PHP-s standardfunktsioonide hulgas, kasutasime PHP laiendust PHP XML Classes. Kuigi lahendus kasutab serverina Linuxit, ei ole siin kasutatud platvormispetsiifilisi komponente.

Lahenduse analüüs

Võrdleme loodud lahendust olemasolevate analoogidega ja analüüsime antud kontseptsiooni rakendamise eri aspekte.

Hinnavaatlus.ee

Loodud lahendus on sarnane riistvara hindade võrdlemiseks loodud veebiportaaliga hinnavaatlus.ee. Hinnavaatlus.ee on täiesti toimiv, sealt saab hea ülevaate Eestis müüdavate arvutustehnika komponentide hindadest.

Iga süsteemiga liitunud firma uuendab oma hinnakirja faili teatud aja tagant ja tagab selle kättesaadavuse veebi kaudu.

Firmadele on antud ette kindel formaat (xls - Microsoft Exceli arvutustabel) ja väljad: toote nimetus, hind, saadavus (laos/tellitav) ja pikem kirjeldus toote kohta. Hinnavaatluse otsingumootor uuendab süsteemis olevat infot teatud ajaintervallide tagant, laadides kaupmeeste kodulehtedelt eelpool kirjeldatud hinnakirja enda süsteemi.

Selline lahendus on lõppkasutaja seisukohast väga hea. Kuigi seal pole kõik kaupmehed esindatud, annab portaal täiesti adekvaatse pildi, kuna on olemas kõik olulisemad firmad.

Semantilise veebi seisukohast on aga selline andmete jagamine vale, kuna:

• faili formaat on platvormispetsiifiline;

• antud fail ei sisalda mingeid andmeid andmete kohta (metaandmeid);

• süsteem pole laiendatav (kui tekib juurde mingi uus väli);

• kui tekib Hinnavaatlus.ee kõrvale uus hinnavõrdlusportaal, siis peaksid kõik kaupmehed looma mingis muus formaadis hinnakirja. See muudaks hinnakirjade jagamise kaupmehe jaoks keeruliseks. Kaupmees peaks koostama mitu paralleelset hinnakirja.

Loodud lahendus

Valminud lahendus on pigem tõestus, et valitud vahenditega on võimalik reaalselt toimivat süsteemi luua, kui teostada võimalikult hea lahendus toodete hindade jälgimiseks. Praktikas oleks sellisel kontseptsioonil kaks võimalikku rakendust: veebiportaal ja riistvaraagent.

Riistvaraagent

Tegemist oleks programmiga, mis asuks kasutaja arvutis ja töötaks sarnaselt programmiga Feedreader. Kuna käesolev versioon on teostatud programmeerimiskeeles PHP siis ei ole see sobiv kasutaja arvutis käivitamiseks. Seega tuleks teha programm näiteks keeles Java (et tagada sõltumatus platvormist).

Veebiportaal

Riistvaraagendi baasil on võimalik arendada välja veebiportaal, kus inimesed saavad ülevaate riistvara hindadest. Tehnilise poole pealt tuleks sel juhul teha mõningaid muudatusi.

Kui algselt loetakse RDF-dokumente reaalajas, st kui kasutaja esitab päringu, laetakse serverisse kõikide valitud kaupmeeste hinnakirjad, siis portaali korral tuleks luua hinnakirjadest lokaalsed koopiad.

Kõige optimaalsem ja lihtsam oleks kasutada mõnda relatsioonilist andmebaasimootorit nagu näiteks MySQL. Sel juhul laetakse hinnakirjad mingi süsteemse protseduuri poolt veebiserverisse teatud ajaintervalli tagant.

Seejärel loetakse andmed andmebaasi. Kasutaja poolt tehtavad päringud suunataks sel juhul samuti andmebaasi. Selline lihtsustus on aga mõttekas, vaid juhul kui hinnakirjade struktuur on väga lihtne. Keerukamate struktuuride puhul ei ole võimalik relatsioonilisi andmebaase kasutada, vaid tuleb teatud ajaintervalli tagant laadida veebiserverisse ja teostada päringuid lokaalses serveris asuvatest failidest. See vähendab võrguliiklust ja annab olulise ajalise võidu just suurte andmemahtude korral.

Andmete vahetus kaupmehe ja agendi vahel

Suurte andmemahtude (kui hinnakirjas on palju tooteid) puhul võib probleemiks saada andmete edastus. XML andmete puhul on väga palju ballasti - andmete ja metaandmete suhe on tihtipeale metaandmete kasuks. Üheks võimaluseks on jagada hinnakiri tükkideks. Näiteks tootegruppide või tootjate kaupa.

Teine optimeerimise võimalus oleks kasutada mingit muud protokolli kui HTTP. Hetkel pole ühtegi sellist levinud protokolli kasutusel, kuid teoreetiliselt on võimalik kasutada efektiivsemaid protokolle.

Antud juhul on meil mõlemal poolel olemas fail, mille erinevus on väike, nii saaksime võrrelda vana ja uut faili, ning laadida alla vaid erinevused. Halvimal juhul jääb andmete hulk samaks faili suurusega, kuid parimal juhul (kui hinnakirjas pole muudatusi toimunud) pole vaja midagi alla laadida. Siinjuures on oluline, et keskmise juhu maht jääb alla hinnakirja suuruse, sest on ebatõenäoline, et muutub kogu hinnakiri.

Kolmas võimalus andmete liikumist hõlbustada, oleks kasutada andmete pakkimist. Kaupmehepoolne server pakib alati hinnakirja faili kokku ja agendid peavad siis sama faili lahti pakkima, et andmeid lugeda. Pakkimist kasutades võidame transpordile kuluva aja pealt ligi kümme korda - just niipalju on võimalik tekstifaili keskmiselt kokku pakkida.

Miks kasutada ontoloogiat?

Loodud mudel on semantilise veebi kontseptsiooni kohaselt loodud ja vastab nõuetele. Andmete esitamiseks kasutatakse standarditele vastavaid esitusmeetodeid (XML /RDF), mis muudab andmed (hinnakirjad) kõigile arusaadavaks. Olenemata kasutatavast operatsioonisüsteemist ja programmerimiskeelest on võimalik neid andmeid kasutada ja töödelda. Samuti on selline süsteem avatud laiendustele, häirimata seejuures neid andmeid kasutavate süsteemide tööd.

Sama probleemi oleks saanud ka lahendada, kasutades traditsioonilisi meetodeid. Tõepoolest, sellisel juhul oleks kogu protsess palju lihtsam, kuid integratsioon erinevate süsteemide vahel oleks palju keerukam. Väljapakutud lahenduse eelis on just universaalsus, platvormist sõltumatus ja laiendatavus. Ontoloogiat kasutades saab mistahes tarkvaraagent pöörduda kaupmeeste poole ja saada sealt tagasi standarditele vastavat masinloetavat infot.

Semantilise veebi visioon ei ole küll veel täielikult realiseerunud, kuid paljud selle „ehitusblokid” on juba olemas. Kokkuvõtvalt võiks öelda, et tulevik sõltub suuresti sellest, kui palju avaldatakse infot, mis on kirjeldatud ontoloogia abil.

Olen kindel, et see teema muutub üha aktuaalsemaks ja inimesed hakkavad visalt eesmärgi poole pürgima ehk siis semantiliste võrgurakenduste loomise suunas.

- Lahendused