Tiedon suodattaminen XML-dokumenteista: XTrie

Reeta Kuuskoski

Helsinki 4. lokakuuta 2002

Rakenteiset dokumentit -seminaarin esitelmä

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

1 Johdanto

Kuten useaan otteeseen on mainittu, Internetin tietomäärä on valtava. Informaation määrän huomattavan kasvun myötä ovat valitsevaan tiedonjakeluun (*selective data dissemination*) perustuvat sovellukset alkaneet kiinnostaa kuluttajia. Tästä tarpeesta on syntynyt nk. *publish-subscribe* -arkkitehtuurin mukaisia järjestelmiä, joissa tiedon tuottaja, kustantaja, tuottaa dataa suurelle joukolle tilaajia, jotka ovat etukäteen määritelleet, minkälaisista asioista ovat kiinnostuneita.

Aiemmin tämäntyyppiset järjestelmät on toteutettu yksinkertaisesti sanaesiintymien tai yksinkertaisten attribuuttien arvojen vertailun perusteella. XML:n yleistyminen on mahdollistanut sisällön lisäksi myös tekstin rakenteen huomioon ottamisen tiedon valinnassa. Etenkin W3C:n standardiksi ehdottamaa XPath-kieltä, jonka avulla XML-dokumentista voidaan paikallistaa yksityiskohtaisestikin määriteltyjä osia, on alettu käyttää apuna suodatuksessa. Mm. XFilter, jonka tässä esitelmässä käsiteltävä XTrie haastaa, perustuu XPath-kielen polkulausekkeiden (*XPath Expressions, XPE*) käyttöön.

Tehtävänä suodatuksessa on siis abstraktimmin muotoiltuna nk. XPath-lausekkeen hakuongelma (*XPE Retrieval Problem*): “ on etsittävä suuresta XPE-joukosta P osajoukko, joka sopii (*match*) XML-syötedokumenttiin D”. Tehtävä ei ole aivan triviaali: ongelmana on XPathilla määriteltyjen suodatinten monimutkaisuus. Tiettyyn suodattimeen/tilaukseen sopivan (*match*) dokumentin tunnistaminen tehokkaasti ei ole helppoa. Etenkin, kun kyseessä on laaja järjestelmä, jossa käsiteltäviä dokumentteja on lukuisia ja käyttäjäkohtaisia suodatinmäärittelyjäkin paljon.

Haun nopeuttamiseksi on luotava XPE-määrittelyille (eli nk. tilauksille) sopiva hakemistorakenne. Koska polkulausekkeet voivat edustaa monimutkaisia puurakenteita, ei ongelma ole triviaali. Yksinkertaiset ratkaisuyritelmät (esim. vain esiintyvien elementtien nimiin perustuva indeksi) olisivat myös tehottomia. Tähän tarpeeseen pyrkii vastaamaan indeksointirakenne XTrie, joka sopii isoillekin järjestelmille.

Tässä esityksessä esitellään XTrien rakenne ja sen toiminta yleisehköllä tasolla. Seuraavassa luvussa esitellään muutamia taustakäsitteitä, jotka helpottavat asian ymmärtämistä. Sen jälkeen luvussa 3 tutustutaan XTrien kannalta oleellisiin käsitteisiin. Luvussa 4 perehdytään järjestelmän tietorakenteisiin ja luvussa 5 käydään yleisesti läpi sen toimintaperiaatetta. Luku 6 sisältää XTrien evaluointituloksia ja sen vertailun XFilter-järjestelmään.

2 Taustaa

Esityksen seuraamisen helpottamiseksi on hyvä muistuttaa mieliin muutamia oleellisia asioita, jotka esitellään seuraavassa lyhyesti.

XML-dokumentti. XML-dokumentin rakenne on hierarkinen. Siinä on juurielementti, jonka alla olevat elementit esiintyvät sisäkkäisesti, limittäisyys ei ole sallittua. Kullakin elementillä on nimi. Elementit rajataan alkumerkin ja loppumerkin (*tag*) avulla ja niihin voi sisältyä dataa tai muita elementtejä. XPath käsittelee XML-dokumenttia puuna, jossa elementtien nimet ovat solmuja. Esimerkkipuu näkyy kuvassa 2 b, jossa solmut b1, b3 jne. kuvaavat b-elementtejä. Solmuihin liittyvät numerot tarkoittavat solmujen käsittelyjärjestystä jäsennyksen aikana.

XPath-polkulauseke, XPE. XPathin polkulausekkeet ovat XML-dokumenttipuuhun sovitettavia rakenteisia hahmoja. Yksinkertaisimmillaan XPE on yksipolkuinen hahmo, joka on joko absoluuttinen, puun juuresta mitattu polku tai suhteellinen, tietystä solmusta lähtevä reitti. Polkulauseke koostuu askeleista, jotka määrittävät solmun (elementin) nimen ja hierarkkiset suhteet solmujen välillä.

Suhteet määritetään "/"- ja "//"-operaattoreilla joista ensimmäinen tarkoittaa, että solmut ovat toistensa äiti ja lapsi, jälkimmäinen tarkoittaa solmujen välistä esivanhemmuus - jälkeläinen -suhdetta. "/"-operaattori voi tarkoittaa vaihtoehtoisesti myös puun juurisolmua. XPath-lausekkeissa voidaan käyttää myös *-merkkiä merkitsemään mitä hyvänsä elementin nimeä. Esim. polkulauseke

$$/a/b//c$$

valitsee kaikki c-elementit, jotka ovat sellaisten b-elementtien jälkeläisiä, joiden äitisolmuna on dokumentin juurisolmu a.

Valittua solmujoukkoa tarkemmin määrittelemään käytetään predikaatteja, jotka esitetään "[]"-merkkien välissä. Predikaattien määrittelyllä voidaan rajoittaa valittujen elementtien attribuutteja tai dataa. Esimerkiksi polkulauseke

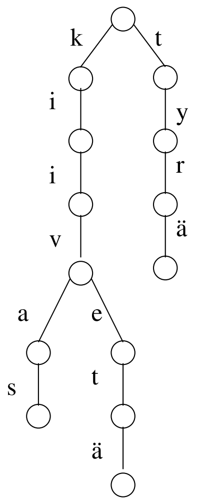
$$/a[b[@x \geq 100]/c]/*/d$$

määrittelee puuhahmon, joka alkaa juurielementistä a. Tällä on kaksi lapsihaaraa, b/c ja $*/d$, joista elementillä b on attribuutti x, jonka arvo on suurempi tai yhtä suuri kuin 100.

Trie-puu. Trie on puurakenne, jota käytetään usein indeksien toteuttamisessa. Siinä samanalkuiset hakemistoyksiköt talletetaan puuhun samalle polulle, joka haarautuu, kun yksiköt eroavat toisistaan. Puun solmusta toiseen kulkevat kaaret vastaavat yksittäisiä merkkejä. Esimerkiksi sanojen 'kiivas', 'kiivetä' ja 'tyrä' muodostama trie näkyy kuvassa 1.

3 Käsitteitä

XTrie-järjestelmä perustuu suodatinta ilmaisevien polkulausekkeiden muodostamaan XPE-puuhun ja näiden polkulausekkeiden hajottamiseen pienempiin osiin,

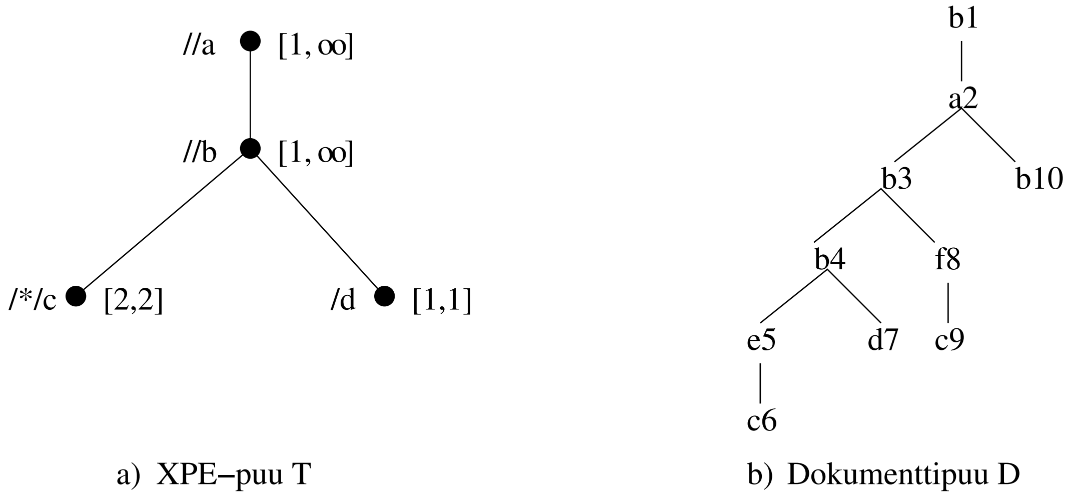


Kuva 1: Trie-puun rakentuminen.

osamerkkijonoihin. Seuraavaksi esitellään tähän liittyviä käsitteitä.

XPE-puu. XPE-puulla tarkoitetaan XPathin polkulausekkeella määritettyä puuhahmoa. Siinä solmun etikettinä (*label*) on elementin nimi, jota edeltää joko / tai // sekä mahdollinen yhden tai useamman */ -merkkijonon sarja. Prefiksit kuvaavat solmujen välisiä suhteita ja ne vastaavat polkulausekkeessa määriteltyjä ehtoja. Kuvassa 2 näkyy XPE-puuesimerkki.

Osamerkkijono. Määritelmän mukaan polkulausekkeen p elementtinimisarja $s = t_1.t_2.\cdots.t_n$ on p:n osamerkkijono (*substring*), jos s on sama, kuin solmujen elementtinimien konkatenointi polulla <$v_1, v_2, \cdots, v_n$ p:n XPE-puussa siten, että kukin v_i on solmun v_{i+1} äitisolmu ja kunkin solmun v_i nimilapulla on prefiksinään ainoastaan "/". Tällä tarkoitetaan siis, että osamerkkijonon elementtien on seurattava toisiaan lausekepolulla suorassa äiti-lapsi -suhteessa. XPE-puun polkua, joka muodostaa osamerkkijonon s, merkitään $Path(s)$. Esimerkkinä kuvan 3 a XPE-puu kuvaa lauseketta $p = /a/b[c/d//e][g//e/f]// * / * /e/f$. Kuvaan on merkitty osamerkkijonot katkoviivoin.

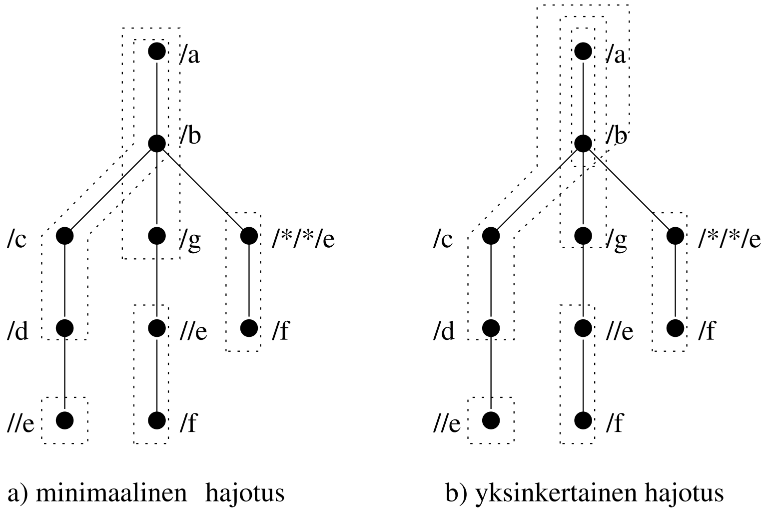


Kuva 2: XPE-puu lausekkeelle $p = //a//b[*/c]/d$ ja XML-dokumenttipuu, johon se sopii.

Osamerkkijonot toimivat XTrie-järjestelmän hakemistoyksikkönä. Koska tällainen elementtien sarja on harvinaisempi, se sopii pienemmällä todennäköisyydellä syötedokumenttiin, jolloin hakemistokyselyjen määrä laskee. Lisäksi pidempään osamerkkijonoon liittyviä, indeksoituja polkulausekkeita on vähemmän, jolloin kunkin indeksikyselyn hinta on luultavasti halvempi.

Osamerkkijonoihin hajottaminen. Hajottamisella (*decomposition*) tarkoitetaan XPE-lausekkeiden jakamista osamerkkijonoihin. XPE:n p osamerkkijonojen sarja $S =< s_1, s_2, \cdots, s_n >$ on p:n osamerkkijonoihin hajottaminen, jos kukin $s_i \in S$ on p:n osamerkkijono ja kaikki p:n XPE-puun solmut sisältyvät jonkun s_i:n polulle. **Minimaalinen** p:n hajottaminen on kyseessä silloin, kun kukin osamerkkijono s_i on maksimaalisen pituinen: p:n XPE-puussa ei ole olemassa pitempää s_i:n sisältävää osamerkkijonoa. Lisäksi on määritelty **yksinkertainen** osamerkkijonoihin hajottaminen, jotta järjestelmä toimisi oikein myös silloin, kun vaaditaan elementtien olevan oikeassa järjestyksessä polkulausekkeeseen nähden. Tällöin minimaaliseen hajotukseen lisätään osamerkkijonot kullekin haarautumasolmulle.

Kuvassa 3 esitetään XPE-puun erilaiset hajotukset.



Kuva 3: XPath-lausekkeesta $p = /a/b[c/d//e][g//e/f]// * / * /e/f$ muodostuvan puun P hajotukset sekä osamerkkijonopuu.

Suhteellinen taso. Kullekin XPE-puun solmulle t voidaan määrittää sen suhteellinen taso suhteessa äitisolmuunsa. Se määrittyy polkulausekkeen perusteella seuraavasti:

- vähintään k, merkitään $relLevel(t) = [k, \infty]$, jos solmun t nimikettä edeltää "$//$", joita seuraa $(k-1)$ "*"-merkkiä,
- tasan k, merkitään $relLevel()t = [k, k]$, jos nimikettä edeltää "$/$", jota seuraa $k-1$ "*"-merkkiä.

Kuvan 2 XPE-puuhun on merkitty kunkin solmun oikealle puolelle sen suhteellinen taso.

Sopiminen (*matching*). Olkoon $t_1, t_2, ..., t_m$ XPE-puun T solmujoukko ja D XML-dokumenttipuu. Sanotaan, että solmu t_i sopii solmuun d_i, jos solmun t_i elementtinimi on sama kuin solmussa d_i. Lisäksi kutakin solmun t_i lapsisolmua t_j kohden tulee olla olemassa vastaava d_i:n lapsi d_j dokumenttipuussa D siten, että $level(d_j) - level(d_i) \in relLevel(t_j)$. Tällä tarkoitetaan siis sitä, että elementtien nimien ja niiden suhteiden ja etäisyyden on oltava samat dokumentissa ja polkulausekkeessa.

Osamerkkijonojen nimityksiä. Äitiosamerkkijonolla voidaan tarkoittaa kahta asiaa: äitiosamerkkijono on joko osamerkkijono, joka on tarkasteltavan osamerkkijonon prefiksi tai sitten osamerkkijono, jonka viimeinen solmu on lapsimerkkijonon ensimmäisen solmun äiti. Vastaavaan tyyliin voidaan määritellä vanhempi- ja jälkeläisosamerkkijono. Lehtiosamerkkijonolla tarkoitetaan osamerkkijonoa, jolla ei ole lapsia.

4 Tietorakenne

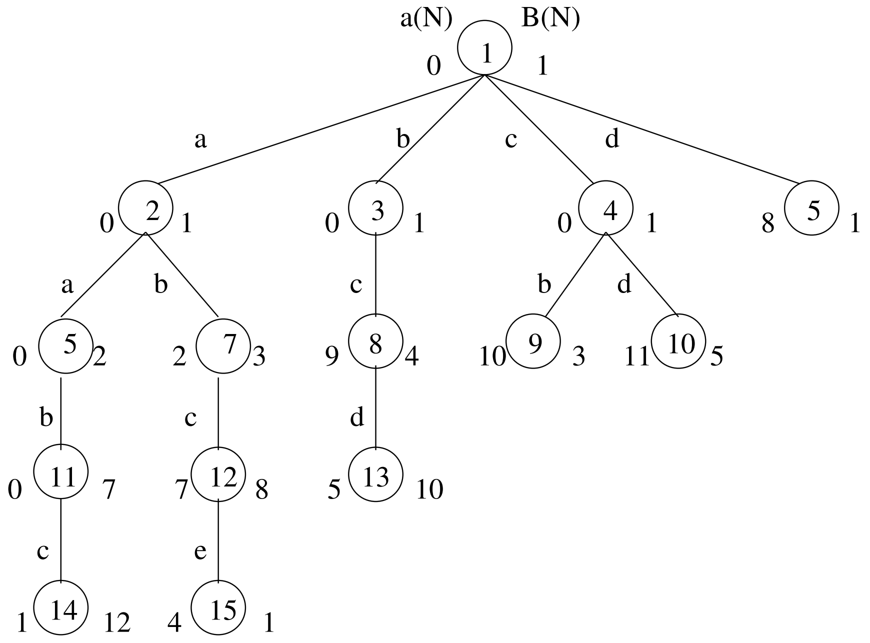
XTrie-järjestelmä indeksoi XML-dokumenttien suodatukseen käytettäviä polkulausekkeita. Indeksi koostuu kahdesta osasta: Trie-puusta T ja osamerkkijonoista tietoa sisältävästä taulusta ST. Puuta käytetään helpottamaan osamerkkijonojen sovittamista XML-syötedokumentin kanssa, ST:n funktio on säilyttää informaatiota kustakin polkulausekkeesta. Hakemiston rakenne näkyy kuvassa 4.

ST**-taulukkoon** talletetaan rivi kutakin polkulausekkeiden osamerkkijonoa kohti. Rivit on järjestetty siten, että samaan polkulausekkeeseen kuuluvat osamerkkijonot ovat peräkkäin. Lisäksi eri polkulausekkeissa esiintyvät identtiset osamerkkijonot on linkitetty listaan ($Next$-sarake). Talletettavat sarakkeet ovat:

- **ParentRow** on ST-taulun monikon numero, jossa kyseisen osamerkkijonon äitimerkkijonon tiedot on talletettuna. Jos osamerkkijono alkaa juuren alta,

p1 = //a/a/b/c/*/a/b	s1=aabc	s2=ab		
p2 = /a/b[c/e]/*/b/c/d	s3=ab	s4=abce	s5=bcd	
p3 = /a/b[c/*/d]//b/c	s6=ab	s7=abc	s8=d	s9=bc
p4 = //c/b//c/d/*/*/d	s10=cb	s11=cd	s12=d	

s	Parent Row	Rel Level	Rank	Num Child	Next
1	0	[4, ∞]	1	1	0
2	1	[3,3]	1	0	3
3	0	[2,2]	1	2	6
4	3	[2,2]	1	0	0
5	3	[4,4]	2	0	0
6	0	[2,2]	1	2	0
7	6	[1,1]	1	1	0
8	7	[2,2]	1	0	12
9	6	[2, ∞]	2	0	0
10	0	[2, ∞]	1	1	0
11	10	[2, ∞]	1	1	0
12	11	[3,3]	1	0	0



Kuva 4: Indeksin rakenne. XPE-joukko osamerkkijonoineen sekä niistä muodostuvat ST-taulukko ja trie-puu T.

$ParentRow = 0$.

- **RelLevel** kertoo kyseessä olevan osamerkkijonon viimeisen solmun suhteellisen eron sen äitiosamerkkijonoon. Käytännössä osamerkkijonon RelLevel = $[< solmujenlkm >, \infty]$, jos osamerkkijonoa vastaavaa polkulauseketta edeltää "//" ja $[< solmujenlkm + x >, < solmujenlkm + x >]$, jos polkulausekkeen edellä on "/", jolloin x merkitsee lausekkeessa esiintyvien "*"-merkkien lukumäärää. Solmujen lukumäärään ei ole laskettu mukaan jo käsitellyn äitiosamerkkijonon solmuja (esim. 'abce' rivillä 4 [2,2], koska 'ab' on jo aiemmin käsitelty).

- **Rank** kertoo sen, monesko lapsiosamerkkijono kyseinen osamerkkijono on äidilleen.

- **NumChild** kertoo, montako lapsiosamerkkijonoa tällä osamerkkijonolla on.

- **Next** on osoitin linkitettyyn listaan, joka yhdistää samaan loogiseen lohkoon kuuluvat, s.o. identtiset, osamerkkijonot. Viimeisellä listan jäsenellä arvo on 0.

Puu T rakennetaan polkulausekkeen yksinkertaisten hajotusten pohjalta. Se sisältää kaikkien polkulausekkeiden osamerkkijonot. Kuhunkin puun solmusta toiseen johtavaan kaareen liittyy elementin nimi ja kullakin solmulla on nimike (*label*), jossa on yhdistettynä ko. solmuun juuresta johtavien kaarten elementit. Juuren nimike on tyhjä. Lisäksi kuhunkin solmuun liittyy kaksi osoitinta:

- **Substring pointer** $\alpha(N)$: jos nimeke(N)$\in S$ (eli kyseessä on osamerkkijonon viimeinen solmu), $\alpha(N)$ osoittaa ko. osamerkkijonon ensimmäisen esiintymän riviin ST-taulussa. Muuten osoitin on tyhjä. Käytännössä osoitin siis kertoo, koska osamerkkijono päättyy.

- **Max-suffix pointer** $\beta(N)$ osoittaa johonkin puun sisäsolmuun ja sen tarkoitus on varmistaa sovittamisalgoritmin oikeellisuus. Etenkin $\beta(N) = N'$, jos nimike(N') on pisin mahdollinen nimike(N):n suffiksi kaikkien sisäsolmujen joukossa. Ellei N' ole olemassa, $\beta(N)$ osoittaa juurisolmuun (arvonaan 1).

5 Algoritmit

XML-dokumenttien suodattamisessa käytetään edellä esiteltyjä, suodattimen polkulausekkeet sisältäviä tietorakenteita. En puutu esitelmässäni järjestelmän algoritmisiin yksityiskohtiin, vaan pyrin kuvaamaan toiminnan yleisellä tasolla.

Perusidea on seuraava: trie-puuta käytetään sopivien osamerkkijonojen havaitsemiseen samalla, kun jäsennetään XML-dokumenttia. Löytyneet osamerkkijonoduplikaatit haetaan ST-taulun Next-kentän perusteella. Edellä mainittujen staattisten tietorakenteiden lisäksi tarvitaan dynaaminen taulukko $\mathcal{B}$, johon talletetaan tietoa dokumentista löytyneistä osamerkkijonoista kullakin dokumentin tasolla. Taulukon ylläpitoa tehdään kaikilla algoritmin tasoilla tarpeen mukaan.

Toiminta jakautuu kolmeen osaan: SEARCH-algoritmi etsii puusta osamerkkijonon, joka sopii XML-muotoiseen syötedokumenttiin. MATCH-SUBSTRING etsii polkulausekkeet, joissa sopiva osamerkkijono on viimeisenä ja PROPAGATE-UPDATE etsii sopivat osamerkkijonovanhemmat lehdestä lähtien.

SEARCH. Haku alkaa juurisolmusta N. Kun XML-dokumentin jäsennyksessä tavataan alkumerkki, seurataan puun kaarta, jossa ko. elementin nimi esiintyy, seuraavaan solmuun N'. Kussakin solmussa on osoitin $\alpha(N') \neq 0$, jos siihen päättyy osamerkkijono. Tällöin käynnistetään MATCH SUBSTRING -aliohjelma. Sen suorituksen päätyttyä palataan käsittelemään dokumenttipuun jäljellä olevaa osaa alkaen siis elementin loppulipukkeesta. Algoritmin suoritusta jatketaan, kunnes koko syötedokumentti on jäsennetty.

MATCH SUBSTRING. Algoritmin suoritus siis alkaa, kun osamerkkijonon osuma havaitaan. Tämä algoritmi tarkistaa osamerkkijonon s osumat ja palauttaa tunnisteet niille XPE:ille, joissa viimeisenä osamerkkijonona on s. Lisäksi se käynnistää PROPAGATE-UPDATE -algoritmin, joka huolehtii sen tarkistamisesta, että koko polkulauseke (eikä vain sen osamerkkijono) sopii XML-dokumenttiin.

PROPAGATE-UPDATE. Tutkii lehtiosamerkkijonon esi-isien sopimisen polkulausekkeisiin. Kertoo siis sen, sopiiko syötedokumentti annettuun kokonaiseen polkulausekkeeseen vai tulisiko se suodattaa pois.

6 Evaluointi

XTrie-järjestelmä haastaa XFilter-järjestelmän, joka sekin pyrkii suodattamaan dokumentteja XPath-lausekkeihin perustuen. XTrien kehittäjät sanovat menetelmänsä olevan tilan- ja ajankäytön kannalta ylivertainen XFilteriin verrattuna.

XFilter ei perustu polkulausekkeiden hajottamiseen osamerkkijonoihin ja puurakenteeseen. Sen toteutus nojaa äärellisiin automaatteihin, joista jokainen vastaa yhden puuhahmon sovittamisesta. Automaattijoukko on indeksoitu yksittäisten elementtinimien mukaan hajautusrakenteeksi.

XTrien tilansäästön takana on osamerkkijonojen pienempi lukumäärä yksittäisten elementtinimien määrään verrattuna. Lisäksi toistuvien hahmojen osittaiset osumat käsitellään XFilterissä kukin erikseen, kun taas XTrie hylkää ne tarpeettomina. Testeissä XTrien kehittäjät totesivat parametreistä riippuen oman menetelmänsä olevan kahdesta neljään kertaan nopeampi kuin kilpailijansa XFilterin.

Lähteet

CFG+02 Chan, C-Y., Felber, P., Garofalakis, M., Rastogi, R., Efficient Filtering of XML Documents with XPath Expressions. *Proceedings of IEEE ICDE'2002*, San Jose, California, February 2002, pp. 235-244.

Lin01 Lindén, G., Document formatting. XSL. XSLT. XPath. *The XML Metalanguage -kurssin luentokalvot*, autumn 2001.

Erk01 Erkiö, H., Dokumenttien indeksointi. *Tiedonhakumenetelmät-kurssin luentokalvot*, kevät 2001.