Metsätehon raportti 151
17.6.2003

StanForD-XML -tiedostojen jatkokäsittely

Juha-Antti Sorsa

StanForD-XML -tiedostojen jatkokäsittely

Juha-Antti Sorsa

Metsätehon raportti 151
17.6.2003

Ryhmähanke: Koskitukki Oy, Metsähallitus, Metsäliitto Osuuskunta, Pölkky Oy, Stora Enso Oyj, UPM-Kymmene Oyj, Vapo Timber Oy ja Yksityismetsätalouden Työnantajat r.y.

Asiasanat: StanForD (Standard for Forest Data and Communications), XML (eXtensible Markup Language), XSL (eXtensible Style Language), XML-ohjelmointirajapinnat, DOM, SAX

Helsinki 2003

SISÄLLYS

TIIVISTELMÄ

Hakkuukoneen tiedonsiirtostandardin XML-versiota (StanForD-XML) on nyt kehitetty ja tutkittu noin kolmen vuoden ajan Metsätehon ja SkogForskin yhteistyönä. Sen tuloksena on syntynyt XML Schema -määritykset tärkeimmille standarditiedostoille: APT, PRD, STM ja PRI. Lisäksi on tehty muunnosohjelmat, joilla nykyisen standardin mukaisia tiedostoja voidaan muuntaa XML:ksi.

Tässä dokumentissa tarkastellaan erilaisia tapoja käsitellä tai erityisesti jatkokäsitellä StanForD-XML -tiedostoja. Tarkoitus on pyrkiä helpottamaan uusien standarditiedostojen mahdollista käyttöönottoa osana yritysten tietojärjestelmiä.

XML-tiedostojen muuntamiseen käytetään paljon XSL-kieltä. Sen avulla on melko helppoa ja nopeata tehdä muunnoksia esitystavasta toiseen. Usein sitä käytetään muuntamaan XML-dokumentti HTML-dokumentiksi, joka voidaan esittää web-selaimella. Tämä mahdollistaa helposti esimerkiksi PRD-tietojen esittämisen jo hakkuukoneen tietokoneella. Jos XML-tiedostosta halutaan tuottaa "julkaisukelpoinen" esitys, se on mahdollista tehdä muunnoksen yhteydessä XSL-FO -muotoilukielellä.

Jotta StanForD-XML -tiedostoja voitaisiin tehokkaasti hyödyntää, pitää ne tallentaa tietokantaan. Nykyiset tunnetut relaatiopohjaiset tietokantajärjestelmät sisältävät kaikki eritasoista tukea XML-tiedostojen tietokantaan tallentamiseen ja kyselyiden tekemiseen. On myös mahdollista ottaa käyttöön puhtaasti XML-dokumenttien tallennukseen erikoistuneita tietokantajärjestelmiä.

Usein voi olla tarvetta liittää hakkuukoneilta saatua dataa osaksi yritysten sovelluksia. Tällöin tarvitaan käytetyssä ohjelmointikielessä rajapinta, jonka kautta voidaan XML-dokumenteissa oleva data siirtää osaksi sovelluksen tietorakenteita. Tunnetuimmat ohjelmointirajapinnat ovat SAX ja DOM, mutta nykyjään ohjelmointiympäristöt tarjoavat myös näiden aika matalan tason rajapintojen lisäksi omia korkeamman tason rajapintoja.

1 JOHDANTO

Hakkuukoneen tiedonsiirtostandardi (Standard for Forest Data and Communications, StanForD) on laadittu asiantuntijatyönä Ruotsissa jo vuonna 1987. Standardia on sen jälkeen kehitetty konevalmistajien ja eri käyttäjien yhteistyönä. Nykyinen standardi on tämän päivän tietotekniikan tasoon verrattuna teknisesti vanhentunut. Lisäksi standardissa on havaittu myös sisällöllisiä ja rakenteellisia ongelmia. Metsätehon aloitteesta päätettiinkin aloittaa alustava kehitystyö hakkuukoneen tiedonsiirtostandardin uudistamiseksi. Keskeistä tässä työssä on ollut nykyisen standardiesitysmuodon muuntaminen XML-esitykseksi. Tätä työtä on tehty vuosina 2001-2002 sekä SkogForskissa että Metsätehossa. Metsätehossa kehitystyö on tehty pääosin projektissa 252.

Kehitystyön aluksi tuotettiin raportti "Towards StanForD-XML", jossa käydään läpi nykyisen standardin ja kehitettävän standardin välisiä rakenteellisia ja teknisiä eroja. Lisäksi raportissa pohditaan erilaisia kehitysvaihtoehtoja uudelle standardille. Tämän raportin pohjalta on tehty formaalit XML-määritykset seuraaville standarditiedostoille: APT, PRD, STM ja PRI. Kunkin tiedoston tietosisältö ja rakenne on määritelty XML Schema -kuvauskielellä ja nämä määritykset ovat saatavissa joko SkogForskin ylläpitämiltä hakkuukoneen tiedonsiirtostandardin kotisivuilta tai Metsätehon tietopalvelun julkisesta verkosta (ks. lähteet liitteessä 1).

Kehitystyön edetessä todettiin tarpeelliseksi tuottaa joukko muunnosohjelmia, joilla nykyisen standardin mukaisia tiedostoja voidaan muuntaa XML muotoon. Muunnosohjelmat on toteutettu samoille tiedostotyypeille, joille on myös skeema-määritykset. Muunnosohjelmat helpottavat sekä StanForD-XML -standardin omaksumista että siirtymistä uudenmuotoisen standardin käyttöön sekä hakkuukoneen että yrityksen tietojärjestelmissä. Muunnosohjelmat ja niiden käyttöohje on pakattu yhdeksi tiedostoksi *ohjelmat.zip* ja se on ladattavissa Metsätehon tietopalvelun julkisesta verkosta alisivujen Tuotteet/Ohjelmistot alta. Liitteessä 2 on lueteltu kaikki tiedostot, joita tuo paketti sisältää.

Muunnosohjelmien avulla voimme kuvitella hypoteettista tilannetta, että meillä on hakkuukoneissa järjestelmät, jotka pystyvät tuottamaan StanForD-XML:n mukaisia hakkuukonetiedostoja. Nyt herääkin kysymys: Mitäpä me niillä sitten teemme ja miten? Selvää on, että ainakin samoja asioita, joita nykyisen standardin mukaisilla hakkuukonetiedostoilla tehdään. XML:n käyttö helpottaa kuitenkin uusienkin hyödyntämistapojen käyttöönottoa, koska se suosittuna IT-teknologiana tarjoaa siihen paljon menetelmiä ja työkaluja.

Seuraavassa vaiheessa StanForD-XML -standardin kehitystyötä tulisi metsäkonevalmistajien ja metsäyhtiöiden ottaa kokeilukäyttöön uuden standardin mukaisia tiedostoja. Tämä edellyttää ohjelmointityötä sekä metsäkoneiden että metsäyhtiöiden tietojärjestelmissä. Alkuun on mahdollista hyödyntää edellä esiteltyjä muunnosohjelmia, mutta päämääränä on, että StanForD-

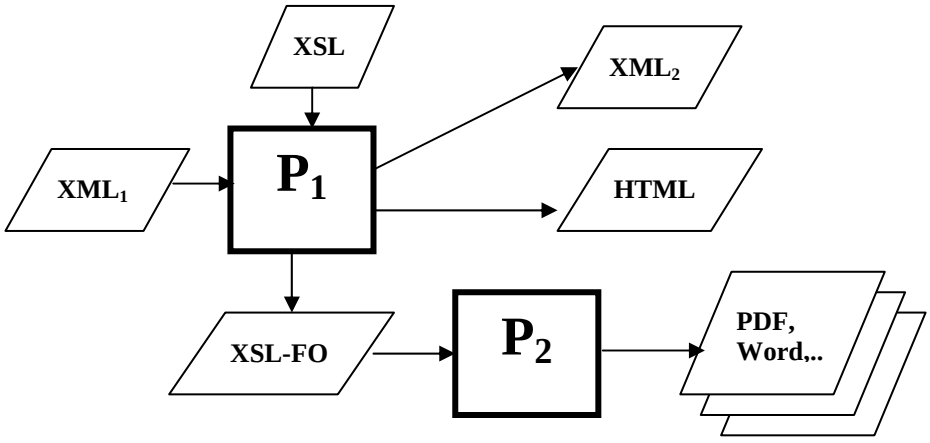
XML -tiedostoja voitaisiin myöhemmin käyttää kuten nykyisen standardin mukaisia tiedostoja tänään.

Tässä dokumentissa tarkastellaan erilaisia tärkeimpiä tapoja hyödyntää StanForD-XML -tiedostoja. Luvussa 2 esitellään, miten standarditiedostoja voidaan muuntaa ja käsitellä XSL-kielen avulla. Luvussa 3 tarkastellaan lyhyesti nykyisten tietokantajärjestelmien tukea XML-datan käsittelyyn ja viimeiseksi luvussa 4 käydään läpi keskeisimmät ohjelmointikielirajapinnat XML-tiedostojen manipulointiin.

2 XSL

2.1 XSL-tyylitiedostot

Tässä luvussa tarkastelemme XSL-tiedostojen (Extensible Stylesheet Language) hyödyntämistä StanForD-XML -tiedostojen prosessoinnissa. XSL on W3C:n (World Wide Web Consortium) "standardoima" kieli kuten XML ja sitä voidaan käyttää kun halutaan muuntaa XML-dokumentti toiseen XML-formaattiin. XSL muodostuu kahdesta erillisestä määrityksestä: XSLT (XSL Transformations, http://www.w3.org/TR/xslt) ja XSL-FO (XSL Formatting Objects, http://www.w3.org/TR/xsl/). XSLT on muunnoskieli, jonka avulla XML-dokumentteja voidaan muunnella monin eri tavoin. XSL-FO (usein tästä käytetään vain nimeä XSL) on muotoilukieli, jolla varsinaisesti vasta esitetään XML-dokumentin tyyli ja muotoilu. XSL:n käyttöä havainnollistaa kuva 1.



Kuva 1. XSL-muunnos.

Yllä olevassa kuvassa P_1 esittää XSL-prosessoria eli eräänlaista tulkkiohjelmaa, jolle annetaan syötteenä muunnettava XML-dokumentti (XML_1) ja muunnosohjelma XSL-tiedostossa. Prosessoinnin tuloksena syntyy tulosdokumentti, joka voi olla esimerkiksi uusi XML-dokumentti (XML_2), HTML-

dokumentti tai XSL-FO -dokumentti. HTML-dokumentti voidaan esittää sellaisenaan selaimessa, mutta XSL-FO -dokumentti joudutaan vielä muuntamaan lopulliseen esityskelpoiseen muotoon erillisellä ohjelmalla (P_2). Tulevaisuudessa tämä muunnosohjelma sisällytetään osaksi sovelluksia (esim. selaimet), jolloin sen käyttö on huomaamatonta. Seuraavissa kappaleissa käydään tarkemmin läpi miten XSL-tyylitiedostoja voidaan käyttää prosessoitaessa XML-StanForD -tiedostoja.

2.2 XSL-muunnoksen suoritus

XSL-muunnokset voidaan suorittaa tarpeesta riippuen eri sovelluksissa. Internet Explorerissa on versiosta 5.1 alkaen ollut sisäänrakennettuna XSL-muunnos, joka suoritetaan automaattisesti, kun selaimeen ladataan XML-dokumentti. Oletusarvoinen toiminta tuottaa XML-dokumentista puhdasta tekstiesitystä luettavamman, lisäämällä tekstiin värejä ja lihavointeja sekä +/- -symboleilla toimivaa esityksen laajentamista/supistamista. Oletusarvoinen toiminta voidaan korvata omalla, lisäämällä ladattavan XML-dokumentin alkuun seuraava teksti (tummennettu):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="prd.xsl"?>
...
```

Tummennetulla rivillä ilmaistaan, että tämä dokumentti esitetään selaimessa käyttäen tyylitiedostoa *prd.xsl*. Jos halutaan tehdä XSL-muunnos web-ympäristössä selaimesta riippumatta, pitää muunnos tehdä web-palvelimella. Microsoftin IIS-palvelimella muunnos tehdään osana ASP-koodia ja Java-pohjaisissa järjestelmissä Servleteissä.

XSL-muunnos voidaan tehdä myös komentotulkista käynnistettävillä työkaluilla tai se voidaan liittää osaksi sovellusta. XSL-ohjelmointirajapintoja on tarjolla jo useisiin ympäristöihin. Esimerkiksi Microsoftin DOM-toteutus (DOM ks. kappale 4.2) tarjoaa mahdollisuuden tehdä XSL-muunnos DOM-esitysmuodolle. Viitteitä XSL-toteutuksiin on lähteissä.

2.3 XSLT

Kuten edellä jo mainittiin, XSLT on muunnoskieli, jolla XML-dokumentti voidaan muuntaa toiseen muotoon. Usein tarve voi olla tuottaa uusi XML-dokumentti, joka kuitenkin eroaa rakenteeltaan alkuperäisestä. Muunnoksessa voidaan hyvin monipuolisesti muokata dokumentin rakennetta esimerkiksi seuraavilla tavoilla:

- uusien elementtien lisäys
- olemassa olevien elementtien poisto
- attribuuttien muuntaminen elementeiksi
- elementtien muuntaminen attribuuteiksi
- elementtien järjestyksen muuttaminen

Eräs käytetyimmistä muunnoksista on tuottaa XML-dokumentista HTML-dokumentti, jolloin se voidaan esittää selaimessa. Esimerkkinä XSLT-muunnoksesta on toteutettu StanForD-XML muotoisen PRD-tiedoston muunnos html-sivuksi, jossa esitetään kunkin puutavaralajin pölkkyjen tilavuus- ja lukumäärätaulukot.

XSLT-dokumentti sisältää joukon template-elementtejä, joissa annetaan muunnettavan dokumentin elementtien ja attribuuttien korvausohjeita. Seuraavassa esimerkkikoodi puutavaralajielementin korvaussäännöstä PRD-tiedostossa.

```
<xsl:template match="Assortment">
      <br></br>
      <h3><xsl:text>Puutavaralaji</xsl:text></h3>
      <xsl:text>Nimi: </xsl:text>
      <b><xsl:value-of select="AssortmentName"/></b>
      <br></br>
      <xsl:text>Koodi: </xsl:text>
      <b><xsl:value-of select="AssortmentCode"/></b>
      <xsl:apply-templates select="NumberOfLogs"/>
      <xsl:apply-templates select="VolumeOfLogs"/>
      <hr></hr>
</xsl:template>
```

Yllä olevassa esimerkkikoodissa kaikki kursiivilla olevat tekstit ovat HTML-kieltä ja ne tulostuvat tulosdokumenttiin sellaisenaan. Huomattavaa on, että koska tulosdokumentti on hyvin muotoiltu XML-dokumentti, pitää kaikissa html-elementeissä olla aina sekä alku- että loppumerkki. Lyhenteitä (vain alkumerkki) ei voi käyttää. Esimerkin tummennetut tekstit ovat XSL-kieltä ja niiden merkitykset ovat seuraavat:

```
<xsl:template match="Assortment">
```

Tämä on elementtihahmon tunnistus. Kaikille Assortment-elementeille suoritetaan tässä annetut korvaussäännöt.

```
<xsl:text>
```

Tämän elementin sisältö tulostetaan sellaisenaan tulosdokumenttiin.

```
xsl:value-of select="AssortmentName"/>
```

Tulosdokumenttiin tulostetaan ko. puutavaralajin AssortmentName-elementin arvo.

```
<xsl:apply-templates select="NumberOfLogs"/>
```

Tässä ilmaistaan, että elementtien korvaussääntöjä sovelletaan ko. puutavaralajin NumberOfLogs-elementeille. Koko muunnosohjelma on tiedostossa *prd.xsl* ja xml-tiedosto, jossa tuota tyylitiedostoa käytetään, on nimeltään *test_xsl_prd.xml*. Nämä tiedostot löytyvät projektin kotisivulta pakatusta ohjelmapaketista *ohjelmat.zip*. Avaamalla *test_xsl_prd.xml*-tiedosto selaimessa (Internet Explorer), tulostuu muotoiltu esitys PRD-tiedostosta.

2.4 Havaintoja XSLT-muunnoksen tekemisestä PRD-tiedostolle

Kun StanForD-XML -tiedostojen rakennetta suunniteltiin, ei tarkastelussa huomioitu varsinaisesti tiedostojen jatkokäyttöä ja sen aiheuttamia vaatimuksia tiedostojen rakenteen suhteen. Kun yksi hakkuukonevalmistajien edustajista osoitti mielenkiintoa PRD-tiedoston jatkokäyttöön, aloitettiin myös tarkastella tätä aihetta. Tavoitteena oli tehdä StanForD-XML mukaisesta PRD-tiedostosta selaimella esitettävä tulostus, josta ilmenisi hakkuussa tehdyt puutavaralajit sekä pölkkyjen lukumäärät ja tilavuudet kussakin puutavaralajissa.

Yksinkertaisen selaimella esitettävän html-tiedoston tuottamiseen XML-tiedostosta paras työkalu on XSLT-kieli. Niinpä se valittiin myös esimerkin toteutustavaksi. Edellisessä kappaleessa esiteltiin lyhyesti kuinka tämä muunnoskieli toimii.

Puutavaralajin pölkkyjen lukumäärät ja tilavuudet on luonnollisinta esittää taulukkona, jossa sarakkeina ovat pituusluokat ja riveinä läpimittaluokat. Luettavuuden kannalta on selvää, että taulukossa pitää olla sarake- ja riviotsikot, joista ilmenevät luokkarajat. Kuvassa 2 on esimerkki tällaisesta taulukosta.

Lpm\Pit	**400**	**430**	**460**	**490**	**520**	**550**
140	0	0	0	0	0	0
150	0	0	0	0	0	0
155	3	0	0	0	1	0
160	3	0	0	0	1	0
170	0	0	1	0	0	0
180	0	1	0	0	0	0
200	1	0	1	0	0	0

Kuva 2. Puutavaralajin pölkkyjen lukumäärätaulukko.

Yllä olevan HTML-taulukon tuottaminen PRD-tiedostosta ei ollutkaan aivan helppo tehtävä. Erityisen vaikeaa on tuottaa rivi- ja sarakeotsikot, koska niitä ei ole tallennettu osaksi matriisia. Yllä olevan taulukon tallennusrakenne XML-tiedostossa on nähtävissä seuraavassa koodissa.

```
...
<NumberOfDiameterClasses>17</NumberOfDiameterClasses>
<NumberOfLengthClasses>7</NumberOfLengthClasses>
<LowerDiameterLimit>150 160 170 190 220 230 250 270 290 310 330 350
370 390 410 430 550 900 </LowerDiameterLimit>
<LowerLengthLimit>402 430 460 490 500 520 550 565
</LowerLengthLimit>
<NumberOfLogs>
     <NumberOfLogsRow>0 0 0 1 1 2 0 </NumberOfLogsRow>
     <NumberOfLogsRow>0 0 0 0 0 0 0 </NumberOfLogsRow>
     <NumberOfLogsRow>0 0 0 0 0 2 0 </NumberOfLogsRow>
     <NumberOfLogsRow>1 2 1 1 0 0 0 </NumberOfLogsRow>
     <NumberOfLogsRow>0 0 0 0 0 0 0 </NumberOfLogsRow>
     <NumberOfLogsRow>0 0 0 2 2 1 1 </NumberOfLogsRow>
     <NumberOfLogsRow>2 0 0 1 0 0 0 </NumberOfLogsRow>
     <NumberOfLogsRow>0 0 0 0 1 1 0 </NumberOfLogsRow>
     <NumberOfLogsRow>0 1 1 1 0 0 0 </NumberOfLogsRow>
     <NumberOfLogsRow>0 0 0 0 0 0 0 </NumberOfLogsRow>
     <NumberOfLogsRow>0 0 0 0 0 0 0 </NumberOfLogsRow>
     <NumberOfLogsRow>0 0 0 0 0 0 0 </NumberOfLogsRow>
     <NumberOfLogsRow>0 0 0 0 3 1 0 </NumberOfLogsRow>
     <NumberOfLogsRow>0 1 1 1 1 0 0 </NumberOfLogsRow>
     <NumberOfLogsRow>0 0 0 0 0 0 3 </NumberOfLogsRow>
     <NumberOfLogsRow>4 2 2 0 1 0 0 </NumberOfLogsRow>
     <NumberOfLogsRow>0 0 0 0 0 0 0 </NumberOfLogsRow>
</NumberOfLogs>
...
```

Sarakeotsikot ovat elementissä `<LowerLengthLimit>` ja riviotsikot ovat elementissä `<LowerDiameterLimit>`. Otsikkojen tuottaminen HTML-esitykseen edellyttää rekursiivisten template-määrittelyjen ja kutsujen käyttöä sekä elementtien arvoina olevien merkkijonojen osittamista osana rekursiivisia kutsuja. Tällaiset XSLT-ohjelmarakenteet vaativat jo aika hyvää ymmärrystä ja kokemusta ohjelmoinnista.

Taulukkoon olisi hyvä saada myös sarake- ja rivisummat sekä mahdollisesti myös summien prosenttiosuudet. Näitä ei esimerkkitoteutuksessa yritetty toteuttaa. Rivisummat pystytään kyllä tekemään vastaavanlaisilla rekursiivisilla template-kutsuilla kuin otsikotkin, mutta sarakesummien toteuttamiseen ei ole ratkaisuehdotusta.

Vaikka XSLT-kielellä pystyy monenlaisiin muunnosohjelmiin, ei sillä pystytä tekemään kaikkea. Sen ilmaisuvoima ei ole niin suuri kuin perinteisten ohjelmointikielten. Eräs kielen keskeinen rajoittava ominaisuus on se, että siinä ei ole käytössä muuttujia, joihin laskennan tilaa voisi tallentaa ja päivittää.

Jos haluamme helpommin tuottaa XSLT-kielellä selaimessa esitettäviä esityksiä PRD-tiedostosta, pitäisi PRD-tiedoston XML-rakennetta muuttaa tätä tavoitetta tukevaksi. Matriisiesitys pitäisi muuttaa sellaiseksi, että se sisältäisi varsinaisen datan lisäksi myös sarake- ja riviotsikot. Lisäksi jos summatietoa halutaan esittää, pitäisi nekin sisällyttää osaksi matriisiesitystä.

Jos XSL-kieltä ei haluta käyttää PRD-tiedoston esittämiseen, voidaan PRD-dokumenttia käsitellä erillisessä sovelluksessa, joka ohjelmoidaan itse kielellä, jossa on käytettävissä XML-ohjelmointirajapinta. Tällöin sovelluksen

ohjelmoijalla on käytettävissä koko käytettävän ohjelmointikielen ilmaisuvoima ja hän pystyy manipuloimaan XML-tietoa haluamallaan tavalla.

2.5 XSL-FO

Kuten luvun alussa todettiin XSL-standardi muodostuu kahdesta osasta. Edellisissä kappaleissa on käyty läpi muunnoskieltä XSLT ja tässä kappaleessa esitellään lyhyesti XSL-FO. Kun XML-dokumentista halutaan tuottaa ulkoasultaan viimeistelty "painotuote", ei HTML-kieli riitä sen esittämiseen. XSL-FO on kieli, jolla tällainen viimeistelty dokumentin ulkoasu voidaan tuottaa.

XSL-FO -tiedosto tuotetaan vastaavalla tavalla kuin HTML-tiedostokin osana XSLT-muunnosta. Seuraava koodi on poimittu *prd_fo.xsl*-tiedostosta ja se esittää samaa puutavaralajielementin korvaussääntöä kuin sivulla 8 ollut HTML-kieltä tuottava muunnos.

```
<xsl:template match="Assortment">
   <fo:block break-after="page">
      <fo:block font-size="14pt" font-weight="bold"
            space-after="12pt">Puutavaralaji</fo:block>
      <fo:block font-weight="bold">Nimi: <xsl:value-of
            select="AssortmentName"/></fo:block>
      <fo:block font-weight="bold" space-after="24pt">
            Koodi:<xsl:value-of select="AssortmentCode"/></fo:block>
      <xsl:apply-templates select="NumberOfLogs"/>
      <xsl:apply-templates select="VolumeOfLogs"/>
      <fo:block></fo:block>
   </fo:block>
</xsl:template>
```

XSL-FO kielen muotoilusäännöt erottuvat esimerkissä kursiivilla ja siitä että ne aina alkavat etuliitteellä *fo:*. Kun PRD-dokumentti prosessoidaan tällä XSL-tiedostolla, syntyy XML-dokumentti, jossa alkuperäisen dokumentin dataan on sisällytetty XSL-FO -muotoilusäännöt. XML-FO -tiedosto voidaan sitten muuntaa esimerkiksi pdf-tiedostoksi erillisen työkalun avulla. Kuvassa 1 sivulla 6 tätä työkalua edustaa prosessori P_2. Eräs ensimmäisistä tällaisista työkaluista, jotka muuntavat XSL-FO -tiedoston pdf-tiedostoksi, on Apache-projektissa tuotettu FOP (ks. lähteet liite 1). Se käynnistetään komentotulkista seuraavasti:

```
fop -xsl prd_fo.xsl –xml prd_no_xsl.xml –pdf prd.pdf
```

Ohjelmalle **fop** annetaan parametrina seuraavien tiedostojen nimet: muunnoksen sisältävä xsl-tiedosto, muunnettava xml-tiedosto ja tuloksena syntyvä pdf-tiedosto.

3 XML JA TIETOKANNAT

Jotta hakkuukoneilta saatavaa dataa voitaisiin hyödyntää myöhemmin, pitää data tallentaa tarkoituksenmukaisesti. Yksinkertaisimmillaan tiedostot tallennetaan yrityksen tietojärjestelmiin sellaisenaan. Tällainen puhtaasti tiedostopohjainen järjestelmä on kuitenkin sekä hankala käyttää että ylläpitää, ja erityisesti tiedon hakeminen siitä on kovin vaivalloista. Jotta hakkuukonedataa voitaisiin hyödyntää, pitää se tallentaa kunnolliseen tiedonhallintajärjestelmään, joka tänä päivänä pääsääntöisesti on tietokanta. Tietokannoista yleisimmin käytössä on relaatiotietokanta.

Tässä luvussa tarkastellaan tilannetta, että meillä on StanForD-XML:n mukaisia hakkuukonetiedostoja ja haluamme tallentaa ne tai niissä olevan datan tietokantaan.

3.1 Tietoa vai dokumentteja

Kenties tärkein tekijä, kun suunnitellaan XML datan tallentamista tietokantaan, on tarkastella itse datan rakennetta. XML data voi olla joko tieto- tai dokumenttikeskeistä (*data- or document-centric*). Tämä eroavaisuus on hyvä ymmärtää, koska sillä on vaikutusta siihen kuinka XML data tallennetaan tietokantaan ja millaista tietokantajärjestelmää kannattaa käyttää. Seuraavassa tarkastellaan erikseen kumpaakin eri tapausta.

3.1.1 Tietokeskeinen XML-data

Tietokeskeistä XML dataa käytetään usein tiedonsiirtoon tietojärjestelmien välillä ja se, että tässä tarkoituksessa käytetään XML:ää ei sinänsä tuo lisäarvoa noille järjestelmille. Hyvä esimerkki tietokeskeisestä XML datasta ovat StanForD-XML -tiedostot.

Tietokeskeisen XML datan ominaisuuksia ovat: melko säännöllinen rakenne, hienojakoiset tietoalkiot, vähän tai ei ollenkaan sekasisältöisiä (*mixed content*) elementtejä[1] ja datan järjestyksellä ei sinänsä ole merkitystä.

3.1.2 Dokumenttikeskeinen XML-data

Dokumenttikeskeiset XML-tiedostot ovat niin kuin luokitteleva nimikin sanoo dokumentteja, jotka on tarkoitettu ihmisten katseltaviksi. Esimerkkejä tällaisesta datasta ovat kirjat, sähköpostit ja mainokset. Niiden ominaisuuksia ovat vähärakenteisuus tai epäsäännöllinen rakenne, yksittäisten dataalkioiden suuri koko (elementin sisältö voi olla koko dokumentti) ja yleensä datan järjestys dokumentissa on merkitsevä. StanFord-XML -tiedostot, kuten edellisessä kappaleessa todettiin, eivät miltään osin ole dokumenttikeskeisiä.

[1] Sekasisältöiset elementit voivat sisältää arvonaan sekä toisia elementtejä että tekstiä. StanForD-XML -tiedostoissa ei ole sekasisältöisiä elementtejä.

3.1.3 Dataa, dokumentteja ja tietokannat

Käytännössä jako tieto- ja dokumenttikeskeisiin XML-tiedostoihin ei ole aina välttämättä selvä. Esimerkiksi muutoin tietokeskeinen XML-tiedosto, kuten jokin lasku, voi sisältää osanaan kuvaustekstiä, joka ei ole selkeästi rakenteista ja hienojakoista dataa. Toisaalta dokumenttikeskeinen tiedosto (esim. jonkin koneen käyttöohje), sisältää selvästi hienojakoista ja selvärakenteista tietoa kuten kirjoittajan nimi, versionumero ja päivämäärä.

Vaikka XML-tiedostot eivät aina olekaan puhtaasti joko tieto- tai dokumenttikeskeisiä, niin niitä kannattaa analysoida, koska se helpottaa oikeanlaisen tietokannan valintaa. Yleissääntönä voidaan pitää, että tietokeskeinen XML data tallennetaan perinteisiin relaatio-, hierarkkisiin tai oliotietokantoihin. Näistä selvästi käytetyin on relaatiotietokanta. XML data tallennetaan näihin joko erillisellä väliohjelmistolla (*middlewar*e) tai sitten tietokannan itsensä tarjoamilla välineillä. Dokumenttikeskeinen XML data voidaan tallentaa erityisiin XML-tietokantoihin tai dokumenttien hallintajärjestelmiin.

Yllä esitetyt säännötkin ovat ohjeellisia. Dokumenttikeskeistä dataakin voidaan tallentaa ja käyttää myös relaatiotietokannasta, kun tarvitaan vain vähän XML-erityispiirteitä. Lisäksi johtaviin relaatiotietokantatuotteisiin on lisätty yhä enemmän XML-piirteitä. Toisaalta myös XML-tietokannat tarjoavat mahdollisuuden tallentaa XML-dokumentteja ulkoisiin relaatiotietokantoihin.

3.2 XML-datan tallennus ja haku

3.2.1 Skeemojen kohdentaminen

XML-dokumentin ja tietokannan skeemojen välinen kohdentaminen (*schema mapping*) suoritetaan elementeille, attribuuteille ja tekstialkioille. Useat fyysiset ja loogiset rakenteet, joita XML-dokumentit sisältävät katoavat sidonnassa. Esimerkkinä loogisesta rakenteesta on XML-dokumentissa samalla tasolla olevien elementtien järjestys. Sitä ei skeemojen sidonnassa välttämättä enää ylläpidetä. Tämä on ymmärrettävää, sillä tietokantaan tallennetaan vain itse XML-dokumentissa oleva data. Tietokannan kannalta on yhdentekevää onko esimerkiksi puutavaralajin nimi ennen koodia vai päinvastoin. Tästä seuraa se, että jos XML-dokumentti kierrätetään (*round-tripping*) tietokannan kautta, niin tuloksena ei välttämättä ole rakenteellisesti identtinen XML-dokumentti, vaikka itse data-alkiot ovatkin samoja.

Seuraavissa alikappaleissa tarkastellaan kahta yleisintä skeemojen kohdentamistapaa: taulupohjainen kohdentaminen (*table-based mapping*) ja olio-relaatio kohdentaminen (*object-relational mapping*).

3.2.1.1 Taulupohjainen kohdentaminen

Taulupohjaista kohdentamista käytetään monissa väliohjelmistoissa, jotka siirtävät dataa kahden tietokannan välillä. Tällöin ensiksi siirrettävä data sarjallistetaan väliohjelmiston avulla XML-dokumentiksi, joka välitetään tietoverkossa toiselle tietokannalle. XML-dokumentin sisältämä data siirretään sitten taas väliohjelmiston avulla tietokannan tauluihin.

Taulupohjaisen kohdentamisen perusidea on, että XML-dokumentti mallinnetaan yhtenä tai useampana tauluna seuraavasti:

```
<database>
    <table>
        <row>
            <column1>...</column1>
            <column2>...</column2>
            ...
        </row>
        <row>
            ...
        </row>
        ...
    </table>
    <table>
        ...
    </table>
    ...
<database>
```

Taulupohjaisessa muunnoksessa on usein mahdollista määrittää siirretäänkö tietokannan taulun sarakkeiden arvot elementtien vai attribuuttien arvoksi vastaavassa XML-dokumentissa. Lisäksi elementit ja attribuutit voidaan nimetä vapaasti.

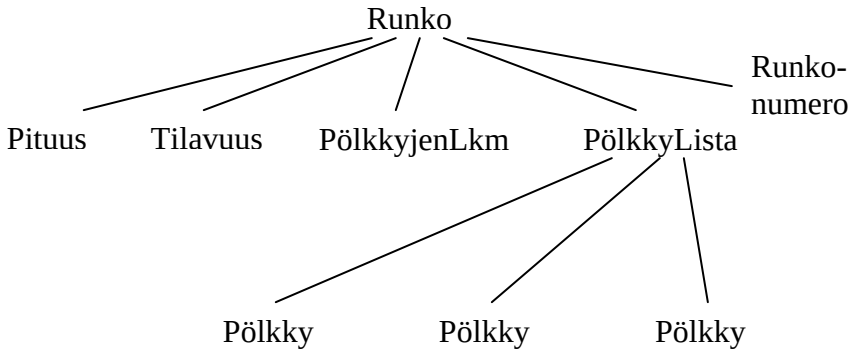
Taulupohjainen kohdennus on hyödyllinen menetelmä, kun siirretään tietoa kahden relaatiotietokannan välillä. Menetelmän selvä puute on, että jos XML-dokumentti ei noudata juuri yllä esitettyä rakennetta, niin taulupohjaista kohdennusta ei voi käyttää. StanForD-XML -tiedostot eivät noudata taulumuotoista esitystä, joten tätä menetelmää ei niissä voida suoraan hyödyntää.

3.2.1.2 Olio-relaatio kohdentaminen

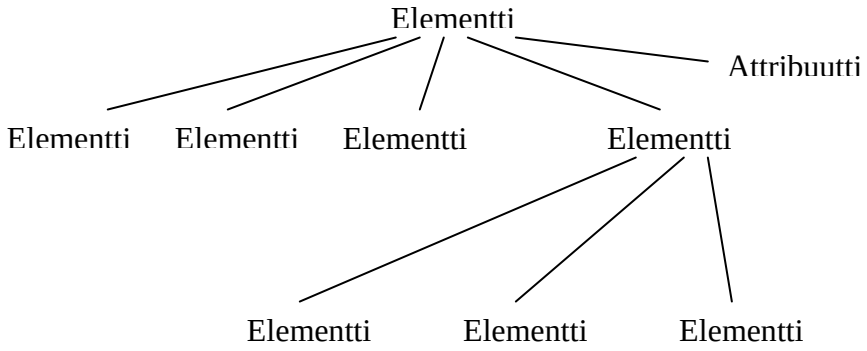
Olio-relaatio kohdentamista käytetään kaikissa XML:ää tukevissa relaatiotietokannoissa sekä joissakin väliohjelmistotuotteissa. Tässä menetelmässä XML-dokumentti mallinnetaan puumaisena olioesitysmuotona. Rakenteiset elementit mallinnetaan oliomallin luokiksi (*classes*) ja attribuutit sekä yksinkertaiset elementit skalaariominaisuuksiksi *(scalar properties*). Tämä oliomalli kohdennetaan sitten relaatiotietokantaan perinteisillä olio-relaatio kohdennustekniikoilla. Näissä tekniikoissa luokat kohdennetaan tietokannan tauluiksi, skalaariset arvot kohdennetaan taulun sarakkeiksi ja olioarvoiset ominaisuudet (eli viitteet toisiin olioihin) kohdennetaan avain/viiteavain-

pareiksi (*primary key/ foreign key*). Jos taas meillä on tietokantatoteutuksena olio- tai hierarkkinen tietokanta, voidaan XML-dokumenttia vastaava olioesitys tallentaa suoraan tietokantarakenteiksi.

On tärkeää huomata, että tässä kohdentamisessa käytettävä oliomalli ei ole sama DOM-malli, joka esitellään kappaleessa 4.2. DOM mallintaa dokumenttia itseään, kun taas tässä esitelty malli kuvaa dokumentissa olevaa dataa. Esimerkiksi seuraava kaavio esittää rungon yksinkertaistettua oliomallia.



Kun taas seuraava kaavio esittää vastaavan XML-dokumentin DOM-esitystä.



Se, onko XML-dokumentin oliomalliesitys vain abstraktio, jolla havainnollistetaan tietokantaan muunnosta, vai luodaanko siitä todellinen ilmentymä, vaihtelee tuotteittain. Kumpi vaihtoehdoista on hyödyllisempi vaihtelee tarpeen mukaan.

3.2.2 Kyselykielet

Usein on tarvetta muuttaa XML-dokumentin rakennetta ennen kuin se tallennetaan tietokantaan. Tähän tarkoitukseen käytetään usein XSLT-muunnosta. Koska XSLT-prosessointi voi olla hidasta, jotkin tuotteet ovat

sisällyttäneet rajoitetun määrän muunnoksia niiden kohdentamistoimintoihin.

Pidemmän aikavälin ratkaisu tähän ongelmaan ovat aidot XML-kyselykielet, jotka palauttavat tuloksenaan XML-dokumentteja. Jo tällä hetkellä monissa tuotteissa tuetaan kyselyjä, joilla relaatiotietokannoista voidaan tuottaa XML-dokumentteja. Kyselyt on toteutettu joko mallipohjaisilla tai SQL-pohjaisilla kyselykielillä.

3.2.2.1 Mallipohjaiset kyselykielet

Yleisimmin käytettyjä kyselytoteutuksia, jotka palauttavat arvonaan XML-dokumentteja relaatiotietokannoista ovat mallipohjaiset kyselykielet (*template-based query languages*). Näissä kielissä ei dokumenttia ja tietokantaa ole kohdennettu mitenkään, vaan relaatiotietokannan kyselyn SELECT-lauseet sijoitetaan mallielementin sisään. Seuraava XML-esimerkki havainnollistaa mallipohjaisen kyselyn toteutusta. SelectStatement-elementti sisältää kyselyn suorittavan SELECT-lauseen ja $-alkuiset sarakkeiden nimet osoittavat, mihin kyselyn tulos sijoitetaan.

```
<?xml version="1.0"?>
<PuulajiLuettelo>
     <Otsake>Tietokannassa on tiedot seuraavista
               puulajeista:</Otsake>
     <SelectStatement>SELECT PuulajinNimi,PuulajinKoodi FROM
                    Puulajit</SelectStatement
          <Puulaji>
               <PuulajinNimi>$nimi</PuulajinNimi>
               <PuulajinKoodi>$koodi</PuulajinKoodi>
          </Puulaji>
</PuulajiLuettelo>
```

Yllä olevan kyselymallin prosessointi voisi tuottaa seuraavanlaisen tuloksen.

```
<?xml version="1.0"?>
<PuulajiLuettelo>
     <Otsake>Tietokannassa on tiedot seuraavista
               puulajeista:</Otsake>
     <Puulajit>
          <Puulaji>
               <PuulajinNimi>Mänty</PuulajinNimi>
               <PuulajinKoodi>1</PuulajinKoodi>
          </Puulaji>
          <Puulaji>
               <PuulajinNimi>Kuusi</PuulajinNimi>
               <PuulajinKoodi>2</PuulajinKoodi>
          </Puulaji>
          <Puulaji>
               <PuulajinNimi>Koivu</PuulajinNimi>
               <PuulajinKoodi>3</PuulajinKoodi>
          </Puulaji>
     </Puulajit>
</PuulajiLuettelo>
```

Mallipohjaiset kyselykielet ovat erittäin joustavia. Kyselyn tulos voidaan sijoittaa mihin tahansa osaan dokumenttia. Esimerkiksi tulosta voidaan käyttää parametrina toiselle SELECT-lauseelle. Usein kielissä on ehto- ja toistolauserakenteet sekä mahdollisuus määritellä muuttujia ja funktioita.

3.2.2.2 SQL-pohjaiset kyselykielet

Toinen tapa toteuttaa XML-kyselyjä relaatiotietokannoissa, on käyttää SQL-pohjaista kyselykieltä. Nämä kielet on XML-piirteillä laajennettuja SQL-kieliä. Perusideana on laajentaa SELECT-lausetta, niin että sillä voidaan tuottaa XML-dokumentti.

Tietokantayritykset ovat nähneet SQL:n laajentamisen XML-piirteillä niin tärkeäksi, että ne ovat lähteneet kehittämään yhteistä kieltä, jolle on annettu nimeksi SQL/XML. Standardointiorganisaatiot ANSI ja ISO ovat myös lähteneet mukaan kielen kehittämiseen. Seuraava koodi on esimerkki SQL/XML-kyselystä, jolla haetaan tietokantaan talletetuista leimikon Perustiedot-taulusta joitakin leimikon numero 3 tietoja.

```
SELECT XMLElement("Leimikko",
     XMLAttribute(Kohdeavain),
     XMLElement("Pvm",Ajankohta),
     XMLElement("Pinta_ala",Pintaala),
     XMLElement("Pääpuulaji",Paapuulaji),
     XMLElement("Hakkuutapa",Hakkuutapa) AS xmldocument
FROM Perustiedot WHERE Kohdeavain="3"
```

Kyselyn tulos voisi olla seuraavanlainen XML-dokumentti, joka sijoitettaisiin xmldocument-tyyppiseen sarakkeeseen.

```
<Leimikko Kohdeavain="3">
     <Pvm>12/01/2002</Pvm>
     <Pinta_ala>4,3</Pinta_ala>
     <Pääpuulaji>Kuusi</Pääpuulaji>
     <Hakkuutapa>Päätehakkuu</Hakkuutapa>
</Leimikko>
```

3.2.2.3 XML-kyselykielet

Malli- ja SQL-pohjaisia kyselykieliä voidaan käyttää vain relaatiotietokannoille. Jos haluamme suorittaa kyselyjä aidoille XML-tietokannoille tai haluamme kohdistaa kyselyn suoraan XML-dokumenttiin, tarvitsemme XML-kyselykielen. Tällainen kieli on W3C:n kehittämä ja tällä hetkellä (04.02.2003) Final Draft statuksen omaava XML-kyselykieli nimeltään XQuery. XQuery sisältää osajoukkonaan W3C:n jo standardoidun XPath-kielen, jolla voidaan määritellä kyselyissä tarvittavat valintalausekkeet. XPath on jo käytössä osana XSLT-kieltä. Kun XQuery vahvistetaan W3C:n standardiksi ja se alkaa olla käytössä sovelluksissa, tulee se korvaamaan juuri XSLT:n käytön helpompana ja tehokkaampana vaihtoehtona.

XML-kyselykieliä voidaan käyttää myös relaatiotietokannoista tehtäviin kyselyihin, kunhan vain tietokannan data mallinnetaan XML:ksi.

4 OHJELMOINTIRAJAPINNAT

XML-dokumentteja voidaan käsitellä melko monipuolisesti korkean tason välineillä kuten XSL-, XSLT- ja XQuery-kielillä. Ne eivät aina kuitenkaan riitä, vaan monissa sovelluksissa tarvitaan "perinteistä" ohjelmointia, jotta XML-dokumenteissa olevaa dataa voidaan täydellisesti hallita. Ohjelmointikielten ja –ympäristöjen toteutuksissa XML-dokumentteja voidaan käsitellä erilaisten ohjelmointirajapintojen kautta. Tässä luvussa esitellään tunnetuimpia näistä rajapinnoista.

4.1 SAX

SAX (*Simple API for XML*) on tapahtumapohjainen ohjelmointirajapinta XML-dokumenttien käsittelyyn. Sen toiminnan perusideana on XML-dokumentin jäsennyksen yhteydessä laukaista tapahtumia, aina kun esimääritellyt kohdat ohitetaan dokumentin jäsennyksessä. Tällaisia kohtia ovat esimerkiksi XML-dokumentin alku ja loppu, elementtien alkumerkit ja loppumerkit sekä varsinainen data merkkeinä. Ohjelmoija on sitten vastuussa siitä, mitä hän tekee näille tapahtumille, joita SAX-jäsentäjä tuottaa. Yleensä toteutettava sovellus muodostuu jonkin oliomallin ympärille ja tämä malli luodaan XML-dokumentista luettavasta datasta. Tämä edellyttää, että ohjelmoija tuottaa oman tapahtumankäsittelijän, joka "kuuntelee" tapahtumia, joita SAX-tuottaa. Käsittelijä ohjelmoidaan sellaiseksi, että se ymmärtää tapahtumien järjestyksen ja rakenteen, joka siis vastaa XML-dokumentin rakennetta, ja tuottaa halutun oliomallin mukaisen ilmentymän.

SAX-toteutus on erittäin tehokas, koska se ei tee mitään muuta kuin käy läpi XML-dokumentin sarjallisesti ja tuottaa läpikäydessään tapahtumia. SAX käyttää vain vakiomäärän keskusmuistia, eikä siis XML-dokumenttien koolla ole väliä (vrt. DOM). Yleisrasitetta voi syntyä kuitenkin siitä, miten ohjelmoija toteuttaa tapahtumankäsittelijän. SAX on siis tehokas, koska se on yksinkertainen, mutta näin ollen se siirtää paljon ohjelmointityötä sovelluksen tekijälle.

4.2 DOM

DOM (Document Object Model) on ohjelmointirajapinta, joka esittää XML-dokumentin sovelluksen tekijälle hierarkkisena puuna. DOM-jäsentäjä luo XML-dokumentista puun, joka muodostuu solmuista, jotka noudattavat täydellisesti XML-dokumentin rakennetta. Puun sisäsolmuina ovat XML-dokumentin elementit ja attribuutit ja lehtisolmuihin on tallennettu varsinainen data. Kappaleessa 3.2.1.2 oli yksinkertainen esimerkki DOM-puun rakenteesta. Jos ohjelmoija haluaa ottaa vain data-alkiot DOM-puun lehdistä ja rakentaa niistä esimerkiksi oman oliomallinsa mukaisen rakenteen, joutuu hän koodaamaan puun läpikäynnin ja olioiden luonnit. Tällaiseen tarkoitukseen tehokkaampaa on kuitenkin käyttää SAXia.

DOM tuottaa aina keskusmuistiin koko XML-dokumenttia vastaavan puurakenteen. Keskusmuistia kuluu sitä enemmän mitä suurempi dokumentti on. Tämä voi aiheuttaa sen, että DOMia ei voida käyttää, jos dokumenttitiedostot ovat suuria.

4.3 Muut rajapinnat

Sekä SAXia että DOMia voidaan pitää matalan tason ohjelmointirajapintoina XML-dokumentteihin. SAXissa sovellusohjelmoija joutuu itse luomaan kaikki rakenteet, joihin data sijoitetaan, ja DOMissa taas oliomallina on puu, jolla ei rakenteena ole välttämättä käyttöä. Niinpä viime vuosina on tullut tarjolle paljon tuotteita, joilla XML-dokumentteja voidaan ottaa helpommin käyttöön osaksi sovelluskehitystä. Näitä tuotteita voidaan yhteisellä nimellä kutsua XML-datan sidontatuotteiksi (*XML data binding products*). Näille tuotteille yhteistä on, että niissä pyritään tuottamaan XML-dokumenttiin tallennetusta datasta suoraan käyttökelpoinen olioesitys ohjelmoijan käytettäväksi. Nämä tuotteet voidaan jakaa kahteen pääluokkaan

- Suunnitteluvaiheen (*design-time*) tuotteet. Nämä tuotteet vaativat aina eksplisiittisen kohdennuksen XML-dokumentista oliomalliin. Usein tämä kohdennus voidaan tehdä graafisen työkalun avulla. Näissä tuotteissa on usein myös mahdollista tuottaa skeemakuvauksesta (XML Schema tai DTD) XML-dokumentin rakennetta vastaavat ohjelmointikielen luokat. Nämä tuotteet tarjoavat joustavamman tavan kohdentaa dokumentin rakenteita oliomalliksi kuin suoritusaikaiset tuotteet.

- Suoritusaikaiset (*run-time*) tuotteet. Nämä tuotteet eivät vaadi mitään esimääriteltyä kohdennusta, vaan ne muuntavat suoraviivaisesti dokumentin olioiksi ja päinvastoin. Tämä tarkoittaa, että tuotteen käyttäjällä ei juuri ole mahdollisuuksia vaikuttaa siihen, kuinka oliomallin luokat kohdennetaan XML-dokumentin rakenteisiin.

LIITE 1

Lähteet ja www-kotisivuja

Metsätehon raportit

Sorsa, J-A. & Vuorenpää, T. 2000. XML:n käyttö hakkuukoneen tiedonsiirtostandardissa. Metsätehon raportti 88.

Sorsa, J-A. 2002. Towards StanForD-XML. Metsätehon raportti 131.

Metsätehon tietopalvelun kotisivu: http://www.metsateho.fi
Ohjelmistot löytyvät alisivujen Tuotteet/Ohjelmistot alta.

W3C:n (World Wide Web Consortium)

Kotisivu: http://www.w3.org/
XSL ja XSLT: http://www.w3.org/Style/XSL/
DOM: http://www.w3.org/DOM/

Ohjelmointirajapintoja

SAX: http://www.saxproject.org/
DOM: http://www.w3.org/DOM/

XML ja tietokannat

Hyviä artikkeleita ja linkkejä: http://www.rpbourret.com/xml/

Hakkuukoneen tiedonsiirtostandardin kotisivu

http://www.skogforsk.se/marknad/stanford/estart.htm

Hyödyllisiä XML-sivuja

XML.COM http://www.xml.com/

Apache-projekti: http://xml.apache.org/

Laaja kokoelma XML-ohjelmistoja: http://www.xmlsoftware.com/

LIITE 2

Ohjelmat.zip -tiedoston sisällön kuvaus

APT_to_XML.exe: APT-tiedoston muunnosohjelma

PRD_to_XML.exe: PRD-tiedoston muunnosohjelma

STM_to_XML.exe: STM-tiedoston muunnosohjelma

stanford_variables.txt: aputiedosto muunnosohjelmia varten

StanForD to XML.rtf: muunnosohjelmien käyttöohje

apt.xsd: APT-tiedoston XML Schema -määritys

prd.xsd: PRD-tiedoston XML Schema -määritys

stm.xsd: STM-tiedoston XML Schema -määritys

test_no_xsl_prd.xml: PRD-esimerkkitiedosto, jossa ei XSL-muunnosta

test_xsl_prd.xml: PRD-esimerkkitiedosto, jossa mukana XSL-muunnos prd.xsl

prd.xsl: PRD-tiedoston XSL-muunnos html-tiedostoksi

prd_fo.xsl: PRD-tiedoston XSL-muunnos XML-FO -tiedostoksi