XML-kyselykielistä loppukäyttäjän näkökulmasta

Minna Lehtonen

Tampereen yliopisto
Informaatiotieteiden yksikkö
Tietojenkäsittelyoppi
Pro gradu -tutkielma
Ohjaaja: Timo Niemi
Huhtikuu 2011

Tampereen yliopisto
Informaatiotieteiden yksikkö
Tietojenkäsittelyoppi
Minna Lehtonen: XML kyselykielistä loppukäyttäjän näkökulmasta
Pro gradu -tutkielma, 60 sivua
Huhtikuu 2011

XML on merkkauskielten keskuudessa johtavassa asemassa. Se on erityisesti internetissä käytävää sovellusten välistä tiedon vaihtoa ja siirtoa koskeva standardi. Monet tiedon esitystavat joko perustuvat suoraan XML:ään, kuten RDF, tai ovat siihen helposti muunnettavissa, kuten esimerkiksi HTML. Myös muihin tietomalleihin perustuvat järjestelmät, kuten relaatiotietokantajärjestelmät, tukevat XML-tiedon esittämistä. XML-pohjainen tietojen esittäminen ja käsittely ovat tämän Pro gradun kannalta keskeisiä teemoja. XML-tiedon hakuun, käsittelyyn ja kyselyyn on kehitetty useita erilaisia järjestelmiä, joista harvat ottavat hyvin huomioon loppukäyttäjien näkökulman. XML-kyselykielet on suunnattu enimmäkseen jo valmiiksi ohjelmointitaitoisille henkilöille, ts. henkilöille joilla on jo tarkka käsitys XML-syntaksista ja valmiudet käyttää monimutkaisiakin kyselyjen ilmaisuja. Tässä Pro gradussa luokitellaan yleisiä XML-kyselykieliä tavallisten loppukäyttäjien näkökulmasta. Erityistä huomiota kiinnitetään eri kielten yksityiskohtiin koskien niiden käyttämisen edellytyksiä ja ilmaisuvoimaa. Kielen ilmaisuvoimaa määritetään esimerkiksi kysymyksillä siitä tukeeko kieli tiedon aggregointia, harmonisointia, ryhmittelyä tai arvotusta. Tässä Pro gradussa tarkastelluista kielistä voidaan tehdä selkeitä päätelmiä siitä, millä kyselykielellä on suurempi ilmaisuvoima ja millä korkeampi deklaratiivisuuden aste.

Avainsanat ja -sanonnat: XML, kyselykielet, XPath, XQuery, XML-QL, RXQL, Xing, XML-GL

Sisällys

1. Johdanto

XML [W3C, 2006] on lyhenne englannin kielen sanoista eXtensible Markup Language, eli laajennettavissa oleva merkkauskieli. Alun perin XML onkin tarkoitettu juuri dokumenttien merkkauskieleksi. Merkkauskieltä (engl. markup language) käytetään antamaan dokumentille rakenne. Dokumenttien rakenne merkitään tunnisteilla (engl. tag), joilla annetaan semanttisia merkityksiä dokumentin eri sisältöosille.

Toisin kuin HTML (engl. HyperText Markup Language) [W3C, 1999a], XML-merkkausstandardi ei sisällä tiedon visuaaliseen esitystapaan liittyvää informaatiota, vaan se sisältää dokumentin tietosisällön hierarkkiseen organisointiin ja rakenteelliseen esittämiseen liittyvää tietoa. XML, kuten myös HTML, kuuluu SGML -kieliperheeseen (engl. Standard Generalized Markup Language) [W3C, 1999b]. SGML antaa yleiset puitteet minkä tahansa merkkauskielen määrittelyyn. Se on siis sellaisenaan itse liian yleinen käytettäväksi tiedon merkkaukseen tietoverkossa.

XML on kieli dokumenttien esittämistä varten, ei dokumenttien käsittelyä varten. Dokumenttien esittämiskielellä tarkoitetaan sellaista kieltä, jossa merkataan dokumenttiin tietyillä tunnisteilla ennestään rakenteettomia osia siten, että ne saavat semanttisia merkityksiä.

XML on nykyään tosiasiallinen standardi vaihdettaessa tietoja verkossa eri sovellusten välillä. Tällä hetkellä suurin osa tiedoista yrityksissä talletetaan kuitenkin relaatiotietokantoihin. Monet suuret yritykset kuitenkin tukevat XML-tietojen talletusta näihin relaatiotietokantoihin. SQL/XML-kyselykieli [SQLX.org, 2004] on eräs sellainen standardi, joka on kehitetty tukemaan XML:n esittämistä ja käyttämistä relaatiotietokantajärjestelmissä. Näissä järjestelmissä SQL-kieltä [ISO/IEC 9075, 2008] käytetään tietojen poimintaan ja valintaan [SQLX.org, 2004]. SQL/XML onkin varsinaisen SQL-standardin laajennos. Sen avulla on mahdollista varastoida XML-dokumentteja relationaaliseen SQL-tietokantaan ja tehdä XPathilla [W3C, 2007c] ja XQuery:llä [W3C, 2007a] kyselyitä näihin dokumentteihin perustuen [SQLX.org, 2004]. Tämän laajennoksen avulla on myös mahdollista muuntaa järjestelmässä jo olemassa olevaa SQL-tietoa XML-dokumenttien muotoon [SQLX.org, 2004]. Kokeneelle SQL:n käyttäjälle SQL/XML-kielen oppiminen tuskin tuottaakaan ongelmia. Yleisesti XML-tietojen talletuksesta relaatiotietokantaan voi kuitenkin seurata haasteita. Esimerkiksi relaatiotietokannassa tietojen järjestystä ei voida säilyttää. On siis ongelmallista tallettaa tällaista järjestyksen säilyttämisen vaativaa tietoa XML-tietoa relaatiotietokantaan järkevästi [Näppilä et al., 2010].

Kaaviokieliä käytetään XML-dokumenttien tietosisällön ja rakenteen määrittelyyn joko yksittäisessä XML-dokumentissa tai XML-dokumenttien kokoelmassa. DTD [W3C, 2006] ja XML Schema [W3C, 2001] ovat yleisimpiä kaaviokieliä tähän tarkoitukseen. Niiden käyttö XML-dokumentin kanssa ei ole pakollista, joten voi olla

sellaisia XML-dokumentteja, joilla ei ole taustallaan minkäänlaista rakennekaaviota. DTD (engl. Document Type Definition), ilmaisee ne elementit ja attribuutit, joiden esiintymä dokumentissa saa olla sekä näiden elementtien ja attribuuttien mahdolliset keskinäiset suhteet. XML Schema on puolestaan tarkempi ja ilmaisuvoimaisempi kuin DTD, sillä se määrittelee lisäksi elementtien ja attribuuttien tietotyypit. Itse asiassa DTD on kontekstivapaa kielioppi ja täten se ei noudata XML:n syntaksia, toisin kuin XML Schema, joka itse on aina myös eräänlainen XML-dokumentti.

XML-pohjaisten tietolähteiden määrä on jatkuvassa kasvussa. Esimerkiksi XML-pohjaiset yritysten väliset sovellutukset ovat yleistymässä. Näissä sovellutuksissa hyödynnetään tietylle yrityssektorille kuuluvien yritysten keskinäisessä kommunikoinnissa ko. sektorille DTD:llä määriteltyä terminologiaa. [Lampathaki et al., 2009]

2. Rakenteiset dokumentit

Dokumenttirakenteiden avulla organisoidaan dokumenttien osia loogisten elementtien perusteella. Näitä loogisia elementtejä ovat esimerkiksi otsikot, kappaleet, osiot jne. Dokumenttirakenteita voidaan myös tarkastella esimerkiksi muotoilun ja asettelun piirteiden, kuten esimerkiksi sivut ja sarakkeet, kautta. [Saïd and Evrard, 2001]

Relaatiotietokannoissa kaavio- ja ilmentymätasot ovat erillisiä. Näissä tietokannoissa relaation kaaviotaso koostuu relaation nimestä ja attribuuttien nimistä. Relaatiotietokannoissa ilmentymätaso taas koostuu n-jonoista, jotka puolestaan koostuvat kaaviotasolla ilmaistujen attribuuttien arvoista. Kaaviotasolla ilmaistaan siis dokumenttien rakenteen kuvaus, eli nimetyt tietoyksiköt ja niiden keskinäiset suhteet. Ilmentymätasolla puolestaan ilmaistaan dokumenttien varsinaiset tiedot, jotka on organisoitu kaaviotason rakenteiden mukaisesti.

Relaatiotietokannat on suunnattu rakenteisen tiedon esittämiseen ja käsittelyyn. Suurin osa olemassa olevasta informaatiosta on kuitenkin talletettu rakenteettomassa muodossa, kuten esimerkiksi tekstidokumentteina. XML-dokumentit edustavat niin sanottua puolirakenteista lähestymistapaa. Puolirakenteisessa tiedossa kaavio- ja ilmentymätaso ovat yhdistettynä. Tiedolla on siis jonkin verran rakennetta, mutta rakenne ei ole täysin säännöllistä eikä sitä voida kuvata kiinteällä kaaviolla. Esimerkiksi DTD ja XML Schema esittelevät potentiaaliset yhteydet, mutta eivät ilmaise mikä niistä vallitsee tietyssä dokumentissa. Puolirakenteisella tiedolla voi olla erilaisia rakenteita ja tiedon vaihtelevia esitystapoja. Puolirakenteisessa tiedossa tunnisteet ovat vapaasti nimettyjä (vrt. HTML, jossa kaikkia tunnisteita ei voi vapaasti nimetä).

Tietyllä tunnisteella (alku- ja lopputunniste) erotetussa dokumentin osassa voidaan erottaa uusia osia käyttämällä muita tunnisteita. Täten sisäkkäiset tunnisteet luovat dokumentille tietyn hierarkkisen rakenteen. XML-dokumentin merkkaus alkaa yleensä

ilmauksella, jossa kerrotaan dokumentin XML-versio sekä käytettävä merkistökoodaus (kuten esimerkiksi UTF-8). Esimerkiksi näin:

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

Muulta osin XML-dokumentti merkitään tunnisteilla, jotka muodostavat hierarkkisen puumaisen rakenteen. XML-hierarkia koostuu elementeistä ja attribuuteista. Elementit koostuvat alku- ja lopputunnisteista sekä niiden välillä olevasta dokumentin varsinaisesta tekstisisällöstä. Elementtitunniste sisältää aina elementin nimen. Elementtitunniste erotetaan dokumentin muusta sisällöstä merkitsemällä sen nimi kulmasulkeiden sisään.

Elementtihierarkia esitetään sisäkkäisinä rakenteina siten, että yksi elementti sisältää dokumentin kaikki muut elementit. Tätä yhtä elementtiä kutsutaan juurielementiksi. Attribuutit liittyvät aina tiettyihin elementteihin ja ilmaisevat joitakin ko. elementtiin liittyviä ominaisuuksia. Ne siis antavat lisätietoa elementeistä, joiden yhteyteen ne on liitetty. Attribuuteilla ei ole omia tunnisteita. Attribuutin nimi ja sen arvo merkitään siis rakenteellisesti elementtitunnisteen sisään. Elementin sisältö on kaikki se osa dokumentista, mikä sen alku- ja lopputunnisteiden välillä esiintyy. Tämä osa voi olla dokumentin tekstisisältöä tai muita elementtejä. Saman nimisten elementtien määrää ei ole erikseen rajoitettu, joten samasta elementistä voi olla useita esiintymiä, joiden sisällöt vaihtelevat.

XML-esityksen voidaan ajatella koostuvan solmuista ja niitä yhdistävistä kaarista. Solmuja on kolmen tyyppisiä. Ensimmäisen tyypin solmut ovat sisäisiä elementtisolmuja, jotka vastaavat XML-dokumentin elementtitunnisteita. Toisen tyypin solmut ovat sisäisiä attribuuttisolmuja, jotka vastaavat XML-dokumentin elementteihin liittyviä attribuutteja. Kolmas tyyppi sisältää puun lehtisolmuja, jotka ovat siis arvosolmuja ja vastaavat dokumentin tunnisteisiin liittyviä tietoarvoja. Puun kaaret kuvaavat rakenteellisia suhteita dokumentin elementtien, attribuuttien ja arvojen välillä. XML-tieto voidaan siis esittää puurakenteena. XML-dokumenttia voidaan sanoa ns. järjestetyksi ja nimetyksi puuksi (engl. labeled and ordered tree). Jossain määrin XML-tietoa voidaan tallentaa myös perinteisiä relaatiotietokantoja hyödyntämällä. Yleisesti voidaan siis sanoa, että XML:n toteutuksessa on käytetty pääasiassa joko puurakenteeseen perustuvaa tallennustapaa tai relaatiotietokantoja. [Gou and Chirkova, 2007]

```
<KIRJASTO1>

   <KIRJASTO_NIMI>Kirjasto 1</KIRJASTO_NIMI>

   <KIRJA>
     <NIMI>Tuulen viemää</NIMI>
     <KIRJOITTAJA>
       <ENIMI>Margaret</ENIMI>
       <SNIMI>Mitchell</SNIMI>
     </KIRJOITTAJA>
   </KIRJA>

   <KIRJA julkaisuvuosi = ”1991”>
     <NIMI>Scarlett</NIMI>
     <ISBN>951-1-11012-8</ISBN>
     <KIRJOITTAJA>
       <ENIMI>Alexandra</ENIMI>
       <SNIMI>Ripley</SNIMI>
     </KIRJOITTAJA>
   </KIRJA>

</KIRJASTO1>
```

1. Esimerkkidokumentti *kirjasto1.xml*

Kuvassa 1 on annettu XML-esimerkkidokumentti *kirjasto1.xml*, jota tullaan käyttämään tutkielman muissa luvuissa olevien esimerkkikyselyjen demonstroinnissa. Tässä XML-esimerkkidokumentissa havainnollistetaan sitä, miten XML-tieto rakentuu XML-dokumentissa.

Elementtien nimet ilmaistaan kulmasulkeiden ’< ja >’ sisässä. Kukin elementti rajataan alku- ja lopputunnisteella. Alkutunnisteessa on vain elementin nimi. Lopputunnisteessa merkitään, elementin nimen lisäksi, nimen eteen aina merkki ’/’. Kuvan 1 esimerkkidokumentin `<KIRJA>` -elementin kahdella ilmentymällä on keskenään erilaiset sisällöt. Jälkimmäisessä elementtiesiintymässä nimittäin on tietoa, jota ensimmäisessä ei ole. Jälkimmäinen `<KIRJA>` -elementtiesiintymä sisältää elementtiesiintymän `<ISBN>`, jota ei ensimmäisessä ole. Jälkimmäinen `<KIRJA>` -esiintymä sisältää myös esiintymän attribuutista `julkaisuvuosi`, jota ei myöskään ole ensimmäisessä `<KIRJA>` -esiintymässä. Se ettei samoja tietoalkioita löydy samaan tarkoitukseen luoduista elementtiesiintymistä, havainnollistaa hyvin XML-dokumentin puolirakenteisuutta. Tietoa ei siis ole organisoitu säännönmukaisesti.

RDF (engl. Resource Description Framework) [W3C, 2004] toimii keskeisenä välineenä semanttisen verkon tietosisältöjen eli ontologioiden esittämisessä. RDF kuvataan XML:llä ja sen avulla voidaan välittää informaatiota verkossa olevien tietoresurssien välillä. RDF käyttää XML:ää yhteisenä syntaksina metatietojen vaihdossa ja prosessoinnissa [Connolly and Begg, 2005].

RDF-kieltä käytetään resurssien kuvaamiseen ja yleensä erityisesti verkkodokumenttien ja verkkosivustojen kuvaamiseen. Useat alakielet tai -kieliopit käyttävät RDF:ää ja määrittelevät joitain XML-elementtejä resurssien esittämiseen. Suosituin tällainen alakielistandardi, jota käytetään RDF:n esittämisessä, on nimeltään Dublin Core. RDF:n syntaksi koostuu kolmesta osasta, joten sitä kutsutaan kolmikoksi. Tästä kolmikosta koostetaan RDF-ilmaus. Kolmikon osat ovat resurssi, nimetty ominaisuus ja ominaisuuden arvo. *Resurssi* on ilmauksen aihe, joka on tyypillisesti jokin verkkodokumentti ja se identifioidaan URI:lla (engl. Uniform Resource Identifier). *Nimetty ominaisuus* on kyseiseen resurssiin liittyvä piirre, kuten esimerkiksi resurssin luoja. *Ominaisuuden arvo* on resurssin sisältö, ilmauksen objekti, joka ilmaisee, mikä arvo ominaisuudella on kyseisessä resurssissa. Mikäli nimetty ominaisuus täten olisi resurssin luoja, voisi ominaisuuden arvo olla resurssin luojan nimi. [Holzner, 2003]

3. XML-käsittelykielet ja XML-kyselykielet

XML-käsittelykielet voidaan jakaa karkeasti kahteen eri kategoriaan. Ensimmäisen kategorian kielet ovat niin sanotusti laskennallisesti täydellisiä dokumenttien ohjelmointikieliä ja toisen kategorian kielet ovat varsinaisia XML-kyselykieliä. Laskennallisesti täydellisellä XML-ohjelmointikielellä voidaan esittää mikä tahansa ohjelmoitavissa oleva tehtävä.

XQuery on tyypillinen laskennallisesti täydellinen XML-käsittelykieli. Tällaisessa kielessä tiedot mallinnetaan nimettyinä ja järjestettyinä puina. Laskennallisesti täydelliset kielet on tarkoitettu lähinnä ohjelmisto- ja tietokantakehittäjille varsinaisten loppukäyttäjien sijasta. Niiden käyttö vaatii runsaasti pohjatietoja niin kielen syntaksista kuin käsittelyn kohteena olevasta dokumentti- ja tietokantarakenteesta. Käsittelykielellä voidaan muokata XML-dokumentissa olevia tietoja varsinaisten hakujen lisäksi.

Varsinaista XML-kyselykieltä puolestaan käytetään puhtaasti tiedon hakuun XML-dokumenttien kokoelmasta. XML-kyselykieltä pitäisi voida käyttää vähäisilläkin pohjatiedoilla.

Sekä XML:ssä että ohjelmoinnissa on olemassa kahdenlaisia kieliä. Näitä kieliä on tietokeskeisiin ja dokumenttikeskeisiin tiedon käsittelytarpeisiin. Dokumenttikeskeisiin tarpeisiin tarkoitetut kielet ovat IR-tyyppisiä (engl. Information Retrieval) tiedonhakukieliä. Kolmannen tyyppiset kielet ovat sellaisia, jotka tukevat molempia.

3.1. XML-ohjelmointikielet

Kaikki XML-käsittelykielet perustuvat enemmän tai vähemmän XPathiin (engl. XML Path Language, eli XML polkukieli). W3C konsortio on kehittänyt ja suositellut XPathin ja XQueryn alun perin. Ne ovat valtavirtaa edustavia tietokantatyylisiä XML-kieliä.

Koska tieto XML-dokumentissa mallinnetaan järjestettynä puurakenteena, on käsittelyssäkin otettava huomioon eri hierarkiatasot. XPath määrittelee polun haluttuun elementtiin dokumentin juuresta tai muusta käsittelyn kohteena olevasta elementtisolmusta. XQueryn polkuilmaukset perustuvat suoraan XPathin syntaksiin. Käsittelykielet sisältävät ilmaisuja kyselyn tulosten aggregointiin, järjestämiseen ja niiden ryhmittelyyn. Aggregoinnilla tarkoitetaan pienemmille havaintoyksiköille laskettujen tunnuslukujen yhdistämistä tai yhteenlaskemista laajempien havaintoyksiköiden tasolle [SuomiSanakirja.fi].

XPath esimerkkikysely 1: etsi dokumentista kirjasto1.xml kaikki elementin `KIRJA` esiintymät

```
/KIRJASTO1/KIRJA  tai  //KIRJA
```

Kyselyn tulos:

```
<KIRJA>
   <NIMI>Tuulen viemää</NIMI>
   <KIRJOITTAJA>
     <ENIMI>Margaret</ENIMI>
     <SNIMI>Mitchell</SNIMI>
   </KIRJOITTAJA>
</KIRJA>

<KIRJA julkaisuvuosi = ”1991”>
   <NIMI>Scarlett</NIMI>
   <ISBN>951-1-11012-8</ISBN>
   <KIRJOITTAJA>
     <ENIMI>Alexandra</ENIMI>
     <SNIMI>Ripley</SNIMI>
   </KIRJOITTAJA>
</KIRJA>
```

---

2. XPath -esimerkkikysely 1: elementtiesiintymien poiminta

XPath -kysely voi sisältää polkuilmauksen, joka alkaa joko merkinnällä ’/’ tai ’//’. Nämä merkinnät edustavat implisiittistä viittausta juurielementtisolmuun siinä

ympäristössä, jossa kysely suoritetaan [Connolly and Begg, 2005]. Esimerkiksi luvussa 2 olevan kuvan 1 XML-dokumentissa voitaisiin XPathilla viitata mihin tahansa `KIRJA` -elementtiesiintymään ilmaisulla '`/KIRJASTO1/KIRJA`', kuten XPath-ilmauksessa kuvassa 2. Merkintä '/' polkuilmauksessa merkitsee välitöntä lapsi-vanhempi -suhdetta peräkkäisten elementtien välillä. Merkin vasemmalla puolella ilmaistaan vanhempielementin nimi ja sen oikealla siitä välittömästi riippuvan lapsielementin nimi.

Merkinnällä '//' puolestaan voidaan viitata mihin tahansa juurielementistä riippuvaan elementtiin tai joukkoon vanhempi-lapsi -elementtejä näiden merkinnän molemmin puolin annettujen elementtien välillä. Kuvan 1 dokumentissa voitaisiin siis viitata mihin tahansa `KIRJA` -elementtiesiintymään myös ilmauksella '`//KIRJA`', esimerkkikyselyn 1 mukaisesti.

Dokumentin ensimmäiseen `KIRJA` -elementtiesiintymään voidaan viitata ilmaisulla '`/KIRJASTO1/KIRJA[1]`', kuten kuvan 3 esimerkkikyselyssä 2. Numero elementin nimen perässä viittaa siis saman nimisten elementtien järjestyslukuun kyseisessä dokumentissa. Viitataan siis siihen järjestykseen, jossa elementtiesiintymät dokumentissa ilmenevät.

XPath esimerkkikysely 2: anna `KIRJA` -elementin ensimmäinen esiintymä esimerkkidokumentista kirjasto1.xml

```
/KIRJASTO1/KIRJA[1]
```

Kyselyn tulos:

```
<KIRJA>
   <NIMI>Tuulen viemää</NIMI>
   <KIRJOITTAJA>
     <ENIMI>Margaret</ENIMI>
     <SNIMI>Mitchell</SNIMI>
   </KIRJOITTAJA>
</KIRJA>
```

---

3. XPath -esimerkkikysely 2: tietyn elementin ensimmäisen esiintymän poiminta

XPathissa kahdella pisteellä '`..`' viitataan minkä tahansa elementin välittömään yläelementtiin, eli vanhempaan. XML-esimerkkidokumentissa (kuva 1) voitaisiin siis esimerkiksi viitata `KIRJA` -elementtiesiintymiin sen `NIMI` -alaelementtiesiintymistä käsin käyttämällä ilmausta '`NIMI/..`'. Yhdellä pisteellä voitaisiin puolestaan viitata nimettyyn tietoalkioon itseensä.

Kuvassa 4 annetussa kolmannessa XPath -esimerkkikyselyssä kysely tuottaa XML-esimerkkidokumentista *kirjasto1.xml* `KIRJA` -elementtiesiintymien `NIMI` -

elementtiesiintymien tekstisisällöt (eli kokoelmassa olevien kirjojen nimet). Tässä esimerkissä *kirjasto1.xml* -dokumentin oletetaan sijaitsevan kokoelmassa `KIRJASTOT`. Polun osa ’`text()`’ kyselyn lopussa merkitsee siis elementin tekstisisältöön viittaamista. Tällainen polun loppuosa voisi olla esimerkiksi myös ’`comment()`’, jolloin valittaisiin kommentti tai ’`node()`’, jolloin valittaisiin kyseinen tietoalkio. Kuvissa 2, 3 ja 4 annetaan sekä varsinaiset XPath -kyselyt että niiden tuottamat tulokset.

XPath esimerkkikysely 3: anna kaikkien `NIMI` -elementtiesiintymien tekstisisällöt esimerkkidokumentista kirjasto1.xml

```
/KIRJASTOT/KIRJASTO1/KIRJA/NIMI/text()
```

Kyselyn tulos:

```
Text: Tuulen viemää
Text: Scarlett
```

---

4. XPath -esimerkkikysely 3: tiettyjen elementtiesiintymien tekstisisältöjen kysely

XPath -polkuilmauksessa attribuutin nimen edessä olevalla ’`@`’ -merkillä erotetaan attribuuttien nimet elementtien nimistä. Attribuutti ilmaistaan polkuilmauksessa elementtinimen perässä suljettuna hakasulkeiden ’[’ ja ’]’ sisään. Esimerkki attribuuttien käytöstä löytyy kuvasta 5.

Dokumenttien käsittelykieli XQuery on johdettu alun perin XML-kyselykielestä nimeltä Quilt [Chamberlin et al., 2001], eikä suoraan XPathista, kuten yleisesti uskotaan. Quilt puolestaan on lainannut piirteitään mm. XPathista, sekä lisäksi muista kielistä, kuten XML-QL [Deutch et al., 1999], SQL ja XQL [Robie, 1999]. XQuery -kyselyt ovat järjestettyjä sekvenssejä atomisia arvoja tai solmuja. Atomiarvolla tarkoitetaan yksittäistä arvoa, joka liittyy yksinkertaiseen tietotyyppiin. Solmu taas voi olla dokumentti, attribuutti, teksti, nimiavaruus, käsittelyohje tai kommentti. XQuery tukee monenlaisia ilmauksia, joita voidaan esittää sisäkkäisinä rakenteina. [Connolly and Begg, 2005]

Kuvassa 5 havainnollistetaan esimerkin avulla kyselyn muodostamista XQuery:llä. Siinä dokumentista *kirjasto1.xml* etsitään `KIRJA` -elementtien esiintymistä sellaisten `KIRJOITTAJA` -elementtiesiintymien sisältämät `SNIMI` –elementtiesiintymät (eli kirjoittajan sukunimeä koskevat elementit), joissa `KIRJA` –elementtiesiintymän attribuutilla `julkaisuvuosi` on arvo ’`1991`’.

Kysely:
```
doc(”kirjasto1.xml”)/KIRJA[@julkaisuvuosi = ”1991”]//SNIMI
```

Tulos:
```
Ripley
```

5. XQuery -polkuesimerkki attribuuttiehdon kera

XQuery -kyselyn polkuilmauksen tuloksena on järjestetty lista tietoalkioita sekä niiden alialkioita. Kukin askel polkuilmauksessa (merkkien ’/’ tai ’//’ välinen osa) kuvaa liikettä dokumentin läpi johonkin suuntaan. Kullakin askeleella voidaan myös eliminoida elementtejä yhtä tai useampaa predikaattia (eli ehtoa) hyödyntäen [Connolly and Begg, 2005]. Polkuilmaus alkaa esimerkiksi viittaamalla tiettyyn tietolähteeseen, kuten kuvan 5 esimerkissä on tehty ilmauksella ’`doc(”kirjasto1.xml”)`’. Tämä ilmaus palauttaa kyseessä olevan nimetyn dokumentin (*kirjasto1.xml*) juurisolmun.

XQueryn syntaksi perustuu niin sanottuun FLOWR -ilmaukseen. FLOWR koostuu sanoista FOR, LET, ORDERBY, WHERE ja RETURN. FOR -silmukka ilmaisee kyselyssä toistettavan osan, eli iteraation. LET -osassa ilmaistaan kyselyn muuttujat. Muuttuja ilmaistaan lisäämällä muuttujan nimen eteen merkki ’`$`’. ORDERBY on valinnainen ilmaus, jota käytetään tulosten järjestyksen määrittämiseen antamalla jokin järjestysehto. Toisessa valinnaisessa osassa, WHERE, kyselylle voidaan antaa sellaisia ehtoja, jotka rajaavat tulosta. RETURN -osassa kuvataan tuloksen tietosisältö ja tulosdokumentin rakenne.

Kuvassa 6 annetaan esimerkki XQueryn LET -ilmauksesta. Esimerkissä muuttuja ’`k`’ saa arvoksensa kokoelmasta *kirjastot.dbxml*, dokumentista *kirjasto1.xml*, alaelementin `NIMI` esiintymät. RETURN -osassa palautetaan nämä `NIMI` -elementtiesiintymät `KIRJAT` -tunnisteiden sisällä. `KIRJAT` -tunnisteiden luonti siis määritellään tässä tuloksen määrittelyosassa; ne eivät siis ole osa alkuperäistä kyselyn kohteena olevaa dokumenttia. Toisin sanoen `NIMI` -elementtiesiintymät ovat `KIRJAT` -elementtiesiintymän alaelementtiesiintymiä.

Kysely:

```
let $k :=
'collection("kirjastot.dbxml")/kirjasto1//NIMI'
    return
    <KIRJAT>$k</KIRJAT>
```

Tulos:

```
<KIRJAT>
    <NIMI>Tuulen viemää</NIMI>
    <NIMI>Scarlett</NIMI>
</KIRJAT>
```

6. XQuery -kysely ja kyselyn tulos (LET -osassa muuttuja)

Polkuorientoituneiden XML-ohjelmointikielten käyttäjien on tunnettava dokumentin rakennetta ainakin osin ja ymmärrettävä mallinsovituksen käsite. Loppukäyttäjien on myös ymmärrettävä muuttujan käsite ja muuttujien soveltamisen periaatteet. Periaatteessa, mikäli halutaan kyetä myös määrittelemään uusia XML-dokumenttirakenteita kyselyiden tuloksiin, on käyttäjän hallittava myös XML-syntaksi.

3.2. XML-kyselykielet

XML-tietojen sovellutukset ja käsittelytarpeet voidaan jakaa kahteen perustyyppiin: tietokeskeisiin (engl. data centric) ja dokumenttikeskeisiin (engl. document centric) sovellutuksiin. Tietokeskeisissä sovellutuksissa tieto on mallinnettu pääasiallisesti säännönmukaisesti organisoitujen rakenteiden avulla. Dokumenttikeskeisissä sovellutuksissa puolestaan tietosisällöt ja rakenteet vaihtelevat ennalta ennustamattomalla tavalla. [Connolly and Begg, 2005]

Tietokeskeisissä sovellutuksissa käyttäjä on kiinnostunut XML-dokumenttien rakenneosasista ja kyselyn muodostuksessa käyttäjän on oltava selvillä siitä, miten käytettävissä olevat XML-dokumentit on todellisuudessa järjestetty [Näppilä et al., 2010]. Näissä sovellutuksissa tieto ilmenee säännöllisessä järjestyksessä ja on automatisoidusti (koneellisesti) luettavissa ja käsiteltävissä sen sijaan, että ihminen kykenisi sitä sellaisenaan lukemaan ja käsittelemään [Connolly and Begg, 2005]. XML-tieto voidaan esittää näissä tapauksissa varsin suoraviivaisesti strukturoitujen tietomallien, kuten relaatiotietokantojen tai oliotietokantojen avulla. Tietokeskeisissä sovellutuksissa XML-tieto voidaan varastoida esimerkiksi relaatiotietokantajärjestelmään, oliorelaatiotietokantajärjestelmään tai oliotietokantajärjestelmään [Connolly and Begg, 2005]. Tietokantoihin (engl. database)

suuntautuneet kyselykielet tukevat tietokeskeisiä sovellutuksia. Tietokeskeiset kyselyt saavat tarkkoja vastauksia.

Dokumenttikeskeisessä mallissa kiinnostuksen kohteena olevat dokumentit voivat olla esimerkiksi lehtiä, sähköpostiviestejä, artikkeleita, kirjoja, ohjeita, tuoteselostuksia jne. Tällaisten dokumenttien rakenne voi vaihdella ennustamattomasti. Niissä oleva tieto ei myöskään ole välttämättä säännöllistä eikä täydellistä [Connolly and Begg, 2005]. Dokumenttikeskeiset XML-dokumentit sisältävät elementtiesiintymiä, jotka koostuvat usein sekoituksesta puhdasta tekstitietoa, joka esitetään tavallisesti joidenkin elementtiesiintymien tietosisältöinä, ja integroituna muiden elementtiesiintymien kanssa [Näppilä et al., 2010]. Dokumenttikeskeisiä sovellutuksia tukevat tiedonhakuun suuntautuneet (engl. information retrieval, IR) suuntautuneet kyselykielet. Relaatiotietokantajärjestelmät eivät tue hyvin dokumenttikeskeisiä sovellutuksia. Sen sijaan sisällönhallintajärjestelmiä tarvitaan tärkeänä työvälineenä näiden dokumenttien käsittelyyn [Connolly and Begg, 2005].

Mikäli XML-tietoja varastoidaan relaatiotietokantaan XML-dokumentit jakautuvat useisiin relaatiotauluihin. Relaatiotietokannat vaativat lisäksi, että tieto järjestetään kiinteän kaavion mukaan. Käytännössä XML-tietorakenteet vaihtelevat usein ennalta määräämättömällä tavalla ja tietosisällöt voivat merkittävästi vaihdella, joten ongelmaksi muodostuu tiedon järjestyksen säilyttäminen. [Niemi et al., 2009]

Dokumenttikeskeisten sovellusten käyttäjät ovat yleensä kiinnostuneita XML-dokumenteista tai sellaisista dokumenttien osista, jotka sisältävät tietyt annetut avainsanat [Näppilä et al., 2010]. Käyttäjä on siis määritellyt kyselyssä oleelliset avainsanat eikä välttämättä tiedä mitään XML-rakenteista kyselyn kohteena olevassa XML-dokumenttikokoelmassa. Dokumenttikeskeiset kyselyt saavat siis tietokeskeisiin kyselyihin nähden epätarkkoja vastauksia.

XML-käsittelykielet XPath ja XQuery ovat tunnettuja yleiskäyttöisiä ohjelmointikieliä. Niitä ei voida kuitenkaan pitää varsinaisina XML-kyselykielinä. Niiden voidaan kuitenkin ajatella olevan tietokantasuuntautuneita, koska niissä käyttäjän on tunnettava käsiteltävä rakenne. Toisin kuin tietokantoihin suuntautuneilla XML-kyselyillä, ei tiedonhakuun suuntautuneille XML-kyselyille ole olemassa tällaisia standardeja kieliä [Gou and Chirkova, 2007]. Tietokantasuuntautuneiden XML-dokumenttien elementtien säännöllisemmän rakenteen lisäksi, dokumentit sisältävät myös syvempiä sisäkkäisiä rakenteita kuin dokumenttikeskeisissä dokumenteissa [Näppilä et al., 2010].

Tietokantasuuntautuneet kyselyt palauttavat kaikki mahdolliset kyselytulokset, jotka vastaavat kyselyllä annettuja tarkkoja rajoituksia ja vaatimuksia. Tiedonhakuun suuntautuneet kyselyt puolestaan tuottavat epätarkempia tuloksia. Nämä tulokset voidaan arvottaa perustuen niiden oleellisuuteen annettuun kyselyyn nähden ja vain

parhaiten arvotetut tulokset palautetaan käyttäjälle [Gou and Chirkova, 2007]. Kyselyiden arvotuksessa relevanssi ja tarkkuus ovat tyypillisesti huomion kohteena.

Edellä esitetty jaottelu tietokeskeisiin ja dokumenttikeskeisiin sovellutuksiin ei kuitenkaan ole ehdoton. Tieto-orientoituneita piirteitähän käsitellään SQL:llä ja dokumenttikeskeisiä piirteitä polkuorientoituneella XML-kielellä. On kuitenkin olemassa myös ns. hybridisovelluksia, eli yhdistelmäsovelluksia, joissa on sekä tietokeskeisten että dokumenttikeskeisten sovellutusten piirteitä. Tällaisia sovellutuksia varten on kehitetty SQL:ää ja XML:ää yhdistäviä kieliä, jotka mahdollistavat XML-tietojen tallennuksen relaatiotietokantoihin. Esimerkiksi luvussa 1 on esitelty lyhyesti tällainen kieli, SQL/XML. SQL/XML tekee mahdolliseksi relaatiotietokantaan tallennetun XML-tiedon poiminnan ja käsittelyn SQL:n avulla.

Puolirakenteista tietoa voidaan varastoida sellaiseen tietokantajärjestelmään, joka perustuu aidosti XML-tiedon omaan mallintamiseen. Tällaisella aidolla XML-tiedon omaan mallintamiseen perustuvalla tietokantajärjestelmällä ei tässä tarkoiteta minkään olemassa olevan tietokantajärjestelmän muokkausta, vaan puolirakenteisen tiedon mallinnusta varten luotua kokonaan omanlaistaan järjestelmää. Puolirakenteista tietoa voidaan myös varastoida perinteiseen tietokantaan kun XML:n ominaisuuksia ei niin paljoa tarvita. Esimerkiksi jos XML-dokumentin tietojen järjestyksellä ei ole suurta merkitystä, silloin tallennus perinteiseen relaatiokantaan on järkevää. Lisäksi näiden kahden tyypin, eli tietokeskeisten sovellutusten ja dokumenttikeskeisten sovellutusten, väliset rajat ovat hämärtymässä. Yhä useammat perinteiset tietokantajärjestelmät nimittäin lisäävät XML-kapasiteettiaan ja XML-tietokannat tukevat dokumenttien osien varastointia perinteisiin tietokantoihin [Connolly and Begg, 2005]. Microsoft, Oracle ja IBM tukevat jo relationaalisen tiedon esittämistä XML-muodossa relaatiotietokantajärjestelmissään [Rys et al., 2005].

XML-tiedon omaan mallintamiseen perustuva tietokantajärjestelmä määrittelee koko XML-dokumentille loogisen tietomallin, dokumentin pelkän tietosisällön sijaan. Dokumentteja tallennetaan ja niihin tehdään hakuja tämän mallin mukaisesti. Vähimmillään tietomallin on katettava XML:n elementit, attribuutit, tekstimuotoinen tietosisältö ja dokumentin järjestys. XML-dokumentin tulisi itse olla loogisen varastoinnin yksikkö, vaikka sitä ei varsinaisesti rajoitetakaan millään fyysisellä varastomallilla. Perinteiset tietokantajärjestelmät tai sovelluskohtaiset varasointimuodot, kuten indeksoidut ja pakatut tiedostot, eivät siis ole täysin pois suljettuja. Voidaan tunnistaa kahdenlaisia XML-tietokantajärjestelmiä: tekstipohjaiset ja malliperustaiset järjestelmät. Tekstipohjaiseen järjestelmään XML-tiedot tallennetaan tekstitiedostoina. Malliperustaiseen järjestelmään XML-tiedot varastoidaan puolestaan jonain sisäisenä puurakenteena. [Connolly and Begg, 2005]

XML-tietoa voidaan varastoida perinteiseen relaatiotietokantaan relaation attribuutin arvona, hajautetussa muodossa, kaaviottomassa muodossa tai jäsennetyssä

muodossa. Jos XML-tietoa tallennetaan relaation attribuutin arvona, on suhteellisen helppoa indeksoida kokonaisia dokumenttitekstejä käsitteellistä ja arvottavaa hakua varten. Yleiset kyselyt ja indeksointi saattavat kuitenkin vaatia haun kanssa samanaikaista jäsentämistä, jolloin suorituskyky kyseenalaistuu. Päivitettäessä tietoa on lisäksi koko XML-dokumentti korvattava aina uudella dokumentilla, sen sijaan, että päivitettäisiin vain dokumentin muuttunut osa. Jos XML-tietoa varastoidaan puolestaan hajautettuna eri relaatioattribuuteille ja relaatioille, voi dokumenttien varastointi edesauttaa tiettyjen elementtien arvojen indeksointia, olettaen että nämä elementit asetetaan omiin relaatioattribuutteihinsa. Tämän lähestymistavan myötä olisi myös mahdollista antaa lisätietoja XML:n hierarkkisesta luonteesta, jolloin olisi myöhemmin mahdollista koota ja järjestää tietoa alkuperäistä XML-dokumentin rakennetta vastaavaan muotoon ja päivittää XML-tietoja. Lähestymistapa vaatisi kuitenkin soveltuvan tietokantarakenteen luomisen. XML:ää voidaan varastoida myös ilman kaavioita tai jäsennetyssä muodossa siten, että muunnetaan XML-tiedot johonkin sisäiseen formaattiin ja tallennetaan sellaisenaan. Ilman kaaviota tallennetun XML-tiedon rakenteen rekursiivisuus voi aiheuttaa suorituskykyongelmia kun etsitään tiettyjä polkuja. Tätä varten on luotava indeksirakenne, joka sisältää polkuilmausten yhdistelmiä ja linkit tietoalkioihin ja niiden ylemmän tason tietoalkioihin (eli vanhempaan). Kun sovelias rakenne on luotu ja XML-tiedot syötetty tietokantaan, voidaan käyttää SQL:ää tiedon kyselyyn. [Connolly and Begg, 2005]

4. Lähestymistavat varsinaisiin XML-kyselykieliin

Kun puhutaan kyselykielistä, käyttäjien tiedontarpeet yleensä tyydytetään poimimalla tiedot sellaisesta tietolähteestä, jonka sisällön käyttäjä tuntee [Niemi and Järvelin, 2006]. Käyttäjä on itse vastuussa siitä, miten häntä kiinnostava tieto löydetään tietolähteestä. Käyttäjä siis määrittelee navigointitavan.

XML-kyselykielten (ja kyselykielten yleensä) tarkoituksena on antaa käyttäjälle mahdollisuus poimia ja valita tietoa jostain tietovarastoista. Tarkoitus on myös kyetä uudelleenstrukturoimaan yhdessä tai useammassa tietovarastossa olevia tietoja, jotta nämä tiedot vastaisivat käyttäjän haluamaa organisointitapaa. [Ceri et al., 1999]

Kyselykieli perustuu aina johonkin tietomalliin. Tietomalli on (matemaattinen) formalismi, joka koostuu tietoyksiköistä ja niiden keskinäisistä suhteista sekä joukosta operaatioita tällä tavalla organisoitujen tietojen käsittelemiseksi. Esimerkiksi relaatiomallissa käytettävät tiedot mallinnetaan relaatioina ja tiedon käsittelyn operaatiojoukko on kätketty johonkin kyselykieleen, kuten SQL:ään.

Se, miten tietoa navigoidaan, riippuu käytettävästä tietomallista. Polkusuuntautuneet kyselyt ovat tyypillisiä XML-tietojen yhteydessä, koska tiedot mallinnetaan nimettynä suunnattuna graafina. XML-dokumentissa elementtiin johtava

polku alkaa dokumentin juuresta ja päättyy kohde-elementtiin. [Niemi and Järvelin, 2006]

Perinteisesti XML-kyselykielten käytössä ei olla yleensä oletettu, että käyttäjä kykenisi käyttämään niitä ilman minkäänlaisia pohjatietoja XML:n syntaksista, dokumenttien rakenteista tai mallinsovitusmekanismeista. Käyttäjän ei myöskään oleteta osaavan ilman jonkinasteisia ohjelmointitaitoja tai tietokantatekniikoiden tuntemista käyttää kyselykieliä.

Mitä korkeampi deklaratiivisuuden aste kyselykielellä on, sitä parempi on kielen soveltuvuus loppukäyttäjälle. Deklaratiivinen lähestymistapa kuvaa mitä tehdään, kun taas proseduraalinen lähestymistapa kuvaa miten jokin asia tehdään. Eli tässä tapauksessa XML-kyselykieli on sitä deklaratiivisempi, mitä vähemmän loppukäyttäjän tarvitsee miettiä, miten tietojen käsittely tapahtuu. Koska polkuorientoituneiden XML-kielten käyttäjien on tunnettava dokumentin rakenne, ymmärrettävä mallinsovituksen ja muuttujan käsitteet sekä osattava XML-syntaksi, on esimerkiksi XPathissa ja XQuery:ssä deklaratiivisuuden aste melko matala. Ne eivät siis sovellu mille tahansa loppukäyttäjälle. Deklaratiivisuus on puolestaan melko korkealla asteella esimerkiksi RXQL-kyselykielessä [Näppilä et al., 2010], jota käsitellään tarkemmin luvun 5 alaluvussa 5.2.

Avainsanahaku on sellaisenaan todettu varsin käyttäjäystävälliseksi tavaksi kyselyjen tukemiseksi verkossa oleviin HTML-dokumentteihin. XML-dokumenttien yhteydessä, avainsanahaku kohdistuu XML-puihin. Sen avulla käyttäjien on mahdollista löytää heitä kiinnostavat tiedot ilman, että heidän olisi opeteltava jokin monimutkainen XML-kyselykieli. Heillä ei myöskään tarvitse olla aiempaa tietämystä tiedon prosessointiin liittyvistä yksityiskohtaisuuksista. Esimerkiksi pienimmän alimman yhteisen esivanhemman (engl. SLCA: Smallest Lowest Common Ancestor) semantiikan mukaan avainsanakyselyn tulos on pienin joukko tietoalkioita, jotka sisältävät kaikki avainsanat joko nimessään tai sisältämiensä lapsitietoalkioiden nimissä. Tulos voi olla myös joukko sellaisia tietoalkioita, joilla ei itsellään ole sellaisia lapsi- tai jälkeläisalkioita, joka myös sisältäisivät kaikki avainsanat. [Xu and Papakonstantinou, 2005]

XML-kyselykieliin yleisesti verrattuna, avainsanahaun puutteena on muun muassa se, että käyttäjällä ei ole mahdollisuutta määritellä tietojen rakenteita uudelleen kyselyiden yhteydessä. Koska kyselykieli ei mahdollista tätä loppukäyttäjälle, on käyttäjän haluaman rakenteen tuottaminen vastauksessa mahdotonta.

4.1. XML-kyselykielen suunnittelun vaatimukset

Ceri ja kumppanit [Ceri et al., 1999] ovat esittäneet XML-kyselykielen suunnittelulle joukon vaatimuksia. Näiden vaatimusten mukaan ensinnäkin XML-kyselykielen tulisi olla tarpeeksi joustava loppukäyttäjien tarpeille, jotta kyselyiden muodostaminen olisi

heille mahdollista. Kyselyitä halutaan kohdistaa kokonaisille verkkosivustoille kerrallaan tai jopa kaikkiin verkossa oleviin XML-dokumentteihin, jolloin huomioon pitää ottaa myös linkit eri dokumenttien välillä. Kyselyiden kohdistaminen kaikkiin sivustoihin tulisi voida tehdä kerralla sen sijaan, että kyselyitä tehtäisiin vain yhteen tiettyyn dokumenttiin kerrallaan. Kyselyissä olisi myös voitava käsitellä dokumentteihin liittyviä DTD:itä sekä XML-pohjaisia muita dokumentteja (kuten RDF-dokumentteja) käyttämällä samaa kyselytapaa kuin XML-dokumenttien käsittelyn yhteydessä.

Näiden vaatimusten [Ceri et al., 1999] lisäksi tietoa tulisi lisäksi voida poimia XML-dokumentista tehokkailla ja deklaratiivisilla mallinsovitukseen ja elementtien käsittelyyn soveltuvilla primitiiveillä. XML-lähtötiedostoa tulisi myös voida muokata siten, että olemassa olevien elementtien välille voidaan määritellä uusia linkkejä. Samoin uusia linkkejä on voitava perustaa uusien kyselyn tuloksena syntyvän XML-dokumentin elementtien välille. Hakupoluille tulisi voida määritellä säännöllisiä ilmauksia (engl. regular expression) tiettyjen varattujen merkkien tai sanojen avulla. Voitaisiin esimerkiksi antaa mahdollisuus seurata rekursiivisia ja satunnaisen pituisia polkuja sivustolla. Tämän yhteydessä voitaisiin myös määritellä ehtoja polun tietoalkioille.

Dokumenttien numeeriseen sisältöön liittyviä laskentatarpeita tulisi tukea kieleen sisäänrakennetuilla funktioilla. Edelleen voitaisiin sallia järjestyksen huomioon ottaminen kyselyssä. Esimerkiksi kyselyssä pitäisi voida ilmaista, että XML-tunnisteiden ja merkkitietojen suhteellinen esiintymisjärjestys on merkityksellinen. Kyselykielen tulisi tarjota myös ilmaus, jossa kerrotaan, että järjestyksellä ei ole merkitystä. Tämä voisi olla oletusarvoinen lähtökohta. Lopuksi XML-kyselykielten kyselyiden tekeminen tulisi olla helppoa ja ymmärrettävää jopa kokemattomammille käyttäjille. Lisäksi kyselyn tulosdokumenttiin liittyvän DTD:n tulisi olla heti ilmeinen käyttäjälle. [Ceri et al., 1999]

4.2. XML-kielten luokittelu

Tässä tutkielmassa kaikki XML:ään perustuvat kielet jaotellaan yleisellä tasolla neljään ryhmään (ks. kuva 7). Nämä neljä ryhmää ovat XML:ään pohjautuvat ohjelmointikielet, dokumenttikeskeisiä sovellutuksia tukevat XML-kielet, tietokeskeisiä sovellutuksia tukevat kyselykielet sekä tieto- ja dokumenttikeskeisiä (molempia) sovellutuksia tukevat kielet.

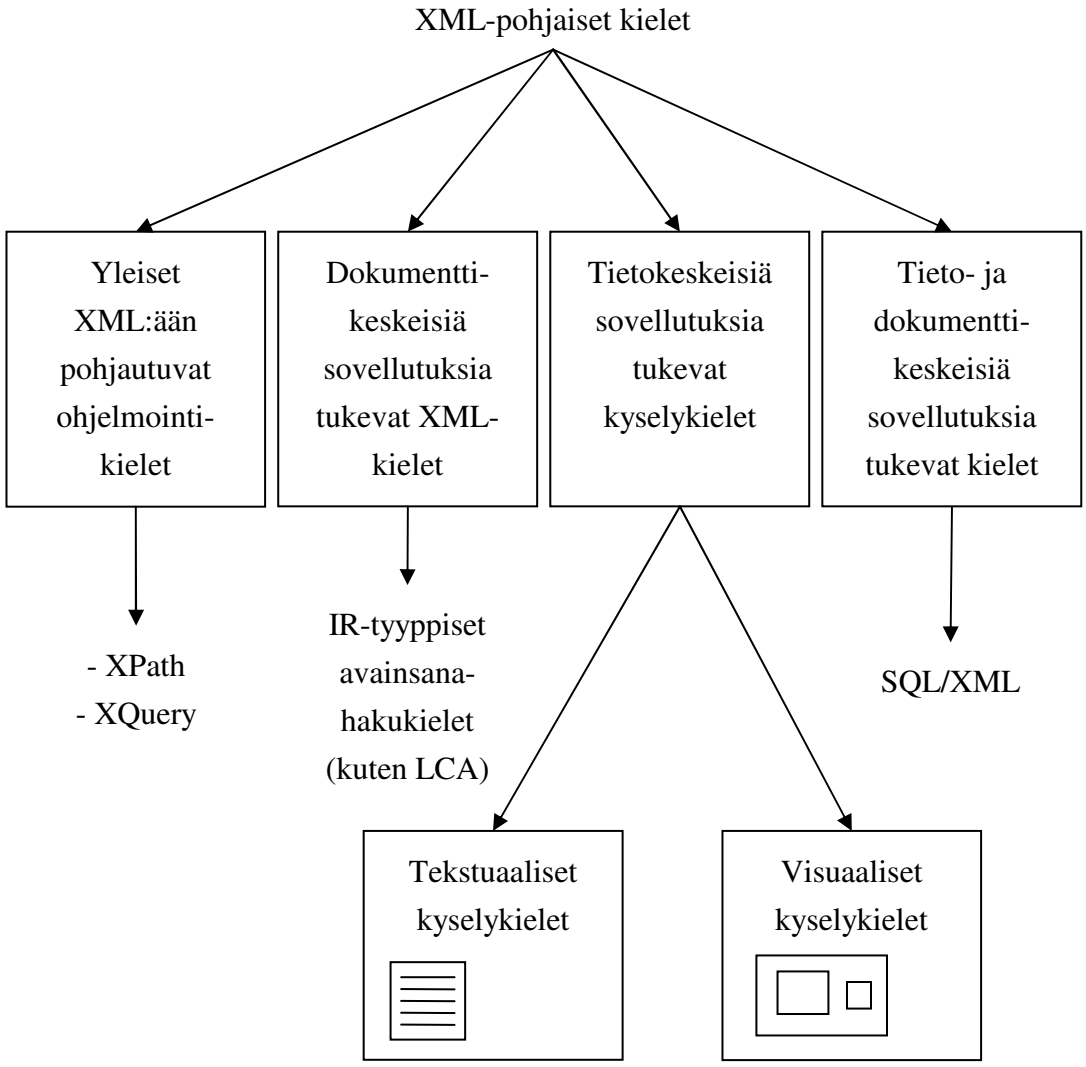
XML:ään pohjautuviin ohjelmointikieliin kuuluvat esimerkiksi XPath ja XQuery. XQuery on laskennallisesti täydellinen XML-kieli ja sisältää mm. XPathin ominaisuudet. IR (engl. Information Retrieval) -tyyppiset XML-kielet, jotka perustuvat aina johonkin poimintastrategiaan kuten LCA, tukevat dokumenttikeskeisiä sovellutuksia.

Tietokeskeisiä sovellutuksia tukevat XML-kyselykielet voidaan puolestaan jaotella tekstuaalisiin ja visuaalisiin kyselykieliin. Tekstipohjaiset kyselykielet koostuvat pelkästään tekstimuotoisista kyselyistä ja niiden tuloksista, kun taas visuaaliset kyselykielet sisältävät graafisia esityksiä dokumenteista ja kyselyistä.

XML-tietojen liittämiseksi relaatiotietoihin on ehdotettu mm. että polkuorientoituneita kyselyilmauksia liitettäisiin SQL:ään. Tämä on kuitenkin melko monimutkaista, sillä käyttäjän on synkronoitava relaatioiden manipulointi XML-puiden manipuloinnin kanssa [Niemi and Järvelin, 2006]. Kuitenkin tyypillisin esimerkki polkuorientoituneista tekstuaalisista kyselykielistä ovat niin sanotut SQL-tyyppiset (engl. SQL-like) kyselykielet, eli sellaiset kyselykielet, joiden rakenne muistuttaa SQL:n syntaksia. XML-relaatioon perustuen (ks. luku 5.2) on myös mahdollista kehittää sellaisia tekstimuotoisia XML-kyselykieliä, joiden ilmaisuaste on korkeampi kuin tavanomaisten polkuorientoituneiden kyselykielten [Niemi and Järvelin, 2006]. XML-relaatioesitys tukee XML-tiedon relaationaalista käsittelyä. Visuaaliset kyselykielet puolestaan perustuvat tiettyjen graafisten tarjolla olevien symbolien käyttämiseen kyselyn tukemisessa.

Kyselykieliä yleisesti ottaen ei ole tarkoitettu monimutkaisten ohjelmointitehtävien suorittamiseen suurista tietojoukoista [Simeoni et al., 2002]. Niiden ilmaisuvoima on yksinkertaisesti liian rajallinen. Tällaisia tarkoituksia varten on olemassa yleisiä XML:ään pohjautuvia ohjelmointikieliä, kuten XML-käsittelykielet (esimerkiksi XQuery). Kyselykielet on tarkoitettu lähinnä valitsemaan ja poimimaan tietoja saatavilla olevista tietovarannoista.

Tietokeskeisiä ja dokumenttikeskeisiä sovellutuksia tukevat kielet ovat SQL:ää ja XML:ää yhdistäviä kieliä (kuten SQL/XML), jotka tekevät mahdolliseksi XML-tietojen tallennuksen relaatiotietokantoihin mahdollistamalla niissä XML:llä esitettävät attribuutit [Connolly and Begg, 2005]. Kuvassa 7 on havainnollistettuna XML-pohjaisten kielten luokittelua eri kategorioihin. XML-pohjaisilla kielillä tarkoitetaan kieliä, joissa käsittelyn kohteena olevat kielet on organisoitu XML:llä. Seuraavaksi tarkastellaan eri kategorioihin kuuluvia XML-pohjaisia kieliä.



7. XML-pohjaisten kielten luokittelu

5. SQL-tyyppiset kyselykielet

SQL-tyyppiset kyselykielet (engl. SQL-like), ovat tekstimuotoisia XML-kyselykieliä. Tällaiset XML-kyselykielet pohjautuvat osittain myös SQL:n rakenteeseen. Nämä kielet on suunniteltu käsittelemään tietoa, joka on epäsäännöllistä (puolirakenteisuus) tai jonka rakenne tunnetaan vain osittain. Esimerkiksi XML-QL [Deutch et al., 1999], XQL [Robie, 1999], SQL/XML[SQLX.org, 2004] ja RXQL [Näppilä et al., 2010] ovat tällaisia SQL-tyyppisiä XML-kyselykieliä.

Loppukäyttäjän näkökulmasta kyselyjen tekemisen tulisi olla mahdollista ilman ohjelmointitaitoja tai käsittelyn kohteena olevien dokumenttirakenteiden hallintaa. Useimmissa XML-kyselykielissä tämä on vain jossain määrin mahdollista. Monia

pyrkimyksiä tähän suuntaan on kyllä tehty. XML-merkintätavan käyttö kyselyiden tekemisessä sellaisenaan ei ole kovin käyttäjäystävällistä. Tästä syystä onkin kehitetty esimerkiksi XSugar -järjestelmä [Brabrand et al., 2008] XML-syntaksin kääntämiseen vaihtoehtoiseksi syntaksiksi, joka ei siis ole XML:ää [Näppilä et al., 2010].

XQL-kieli pohjautuu XPathiin siinä mielessä, että siinä viitataan XML-dokumentin osiin polkuilmausten avulla. XQL laajentaa XPathia useilla operaatioilla, kuten tietojen liitos- ja tekstinhallintaominaisuuksilla [Erwig, 2003]. XQL-kyselyt ovat periaatteessa melko yksinkertaisia, koska ne poimivat ja valitsevat solmuja ja alipuita yhdestä lähtödokumentista ja antavat vastauksena poiminta- ja valintakriteerit täyttävän XML-dokumentin. XQL-kyselyssä on kaksi osaa: pakollinen polkuilmaus ja valinnainen suodatinilmaus. Polkuilmaus määrittelee kyselyn tulokseen tulevat tietoalkiot ja suodatinilmaus hyväksyy vain ne tietoalkiot, jotka tyydyttävät määritellyt kriteerit [Ives and Lu, 2000].

SQL/XML:ssä SQL:ää käytetään relationaalisesti organisoitujen tietojen käsittelyyn kun taas XML:llä kuvattujen attribuuttien käsittelyyn käytetään XQuerya. SQL:ään onkin jo SQL:2003 standardissa [ISO/IEC 9075-14, 2003] määritelty laajennoksia, joiden tarkoituksena on tukea relationaalisen ja XML-tiedon käsittelyä yhdessä olemassa olevissa relaatiotietokantajärjestelmissä. SQL/XML antaa XML-tietotyypille (eräs attribuuttityyppi) joukon operaatioita ja keinoja esittää relationaalisesti esitetty tieto XML:llä. Tämä standardi ei määrittele, miten käänteinen prosessi tulisi suorittaa. Eli se ei sisällä sellaisia keinoja, joiden avulla voitaisiin XML-tietoja muuttaa relationaaliseen esitystapaan. [Connolly and Begg, 2005]

Verkossa käytettäviä SQL-tyyppisiä kyselykieliä ovat esimerkiksi W3QS [Konopnicki and Shmueli, 1995] ja WebOQL [Arocena and Mendelzon, 1999]. W3QS, eli WWW Query System, on sellainen kyselyjärjestelmä, joka perustuu johonkin laajennettavaan järjestelmäarkkitehtuuriin. Laajennettavalla järjestelmäarkkitehtuurilla tarkoitetaan, että käytetään sellaisia suunnitteluperiaatteita, joissa otetaan huomioon kyseisen järjestelmän mahdollinen tuleva kasvu ja pyritään helpottamaan mahdollisten lisätoimintojen tai -osien lisääminen muuttamatta olemassa olevaa toiminnallisuutta. WebOQL (Web Object Query Language) on funktionaalinen kieli, jossa on mahdollisuus voimakkaiden operaattorien käyttämiseen. WebOQL perustuu heterogeenisen verkkosisällön monipuoliseen hyperpuuesitykseen [Erwig, 2003]. Sen suunnitteluun on vaikuttanut UnQL:n [Buneman et al., 1996] käyttämä tietomalli, joka on alun perin tarkoitettu yleiseksi puolirakenteisen tiedon kyselykieleksi [Erwig, 2003]. UnQL-kyselyillä itsellään on myös tuttu SQL-tyyppinen rakenne, mutta ne sisältävät voimakkaita polkuilmauksia, jotka on määritelty tiedon puurakenteiden perusteella [Simeoni et al., 2002].

Näppilä ja kumppanit [Näppilä et al., 2010] esittelevät uuden SQL-tyyppisen XML-kyselykielen, RXQL:n, XML-tiedon relationaaliseen esitystapaan perustuen (ks.

alaluku 5.2). Myös XML-QL on eräs tunnettu tekstimuotoinen SQL-tyyppinen XML-kyselykieli (ks. alaluku 5.1). Molempia kyselykieliä havainnollistetaan muutamien yksinkertaisten esimerkkien avulla. Kaikki esimerkit pohjautuvat luvussa 2 esitettyyn XML-esimerkkidokumenttiin (kuva 1).

5.1. XML-QL

XML-QL:n voidaan ajatella olevan melko deklaratiivinen kyselykieli. Sen käyttö kuitenkin vaatii käyttäjältä tiettyjä ennakkotietoja. Käyttäjän on ennen kaikkea ymmärrettävä mallinsovitustekniikka ja tunnettava muuttujien alustamisen periaatteet. XML-QL on niin sanotusti relationaalisesti täydellinen kyselykieli [Deutch et al., 1999]. Relationaalisesti täydellisellä kielellä voidaan suorittaa perusoperaatioita relaatioille ja sen ilmaisuvoima on sama kuin relaatioalgebran.

XML-QL -kyselykielessä haluttu tieto poimitaan XML-dokumenteista määrittelemällä XML-syntaksilla niin sanotut elementtimallit (engl. patterns), jotka sisältävät muuttujia. Nämä elementtimallit sovitetaan annettuihin dokumentteihin [Deutch et al., 1999]. Kun sovitus onnistuu, muuttujat sidotaan löydettyihin arvoihin. Elementtimallissa annetuilla muuttujilla viitataan joko joihinkin dokumentin elementti-ilmentymiin tai elementin nimeen.

Muuttujat ilmaistaan mallissa asettamalla taalamerkki ’$’ muuttujan nimen eteen. Elementti-ilmentymään viittaava muuttuja merkitään ilman tunnistetta merkitseviä kulmasulkeita ’*$muuttuja*’ ja elementin nimeen liittyvä muuttuja merkitään kulmasulkeiden kanssa ’<*$muuttuja*>’. Elementtien lopputunnisteet voidaan merkitä vain tyhjällä tunnisteella ’</>’ kirjoittamatta elementin nimeä. Mallinsovituksessa vastaavuus alkutunnisteeseen haetaan elementtien sisäkkäisyysjärjestyksessä.

XML-QL -kyselyssä CONSTRUCT -ilmauksessa määritellään vastauksena annettavan tulosdokumentin sisältö ja rakenne. WHERE -ilmauksessa puolestaan määritellään mitä kyselyn kohteena olevasta dokumentista poimitaan. Tästä annetaan esimerkki kuvassa 8.

Vaikka XML-QL on tekstuaalinen kyselykieli, se eroaa muista XML-kyselykielistä siinä, että kyselyt ilmaistaan XML-malleilla. Käytännössä tämä tarkoittaa sitä, että käyttäjä esittää kyselyn kuvaamalla miltä tuloksen haluaa näyttävän. Yleisempi tapa on kuvata miten kyselyn lopputulos rakennetaan.

Kuvassa 8 annetaan esimerkkikysely XML-QL:llä. Kyselyssä haetaan tulosdokumenttiin kaikki kirjat XML-esimerkkidokumentista *kirjasto1.xml*.

XML-QL -esimerkkikysely 1:

```
CONSTRUCT
    <KIRJAT> {
        WHERE
            <KIRJASTO1>
                <KIRJA></KIRJA> ELEMENT_AS $k
            </KIRJASTO1>
        CONSTRUCT $k }
    </KIRJAT>
```

Tulos:

```
<KIRJAT>
   <KIRJA>
     <NIMI>Tuulen viemää</NIMI>
     <KIRJOITTAJA>
       <ENIMI>Margaret</ENIMI>
       <SNIMI>Mitchell</SNIMI>
     </KIRJOITTAJA>
   </KIRJA>

   <KIRJA julkaisuvuosi = ”1991”>
     <NIMI>Scarlett</NIMI>
     <ISBN>951-1-11012-8</ISBN>
     <KIRJOITTAJA>
       <ENIMI>Alexandra</ENIMI>
       <SNIMI>Ripley</SNIMI>
     </KIRJOITTAJA>
   </KIRJA>
</KIRJAT>
```

8. XML-QL -esimerkkikysely 1 *kirjasto1.xml* -dokumentista

Kuvan 8 XML-QL -esimerkkikyselyssä ulommalla CONSTRUCT -lauseella ’`CONSTRUCT <KIRJAT>`’ rakennetaan uusi juurielementti `KIRJAT`. Juurielementin sisältö ilmaistaan aaltosulkeilla ’{}’ suljettuna. WHERE -lauseella poimitaan `KIRJASTO1` -elementin sisällä olevat `KIRJA` -elementtiesiintymät ja sidotaan ne muuttujaan `$k` ilmauksen ELEMENT_AS avulla. Löydetyt `KIRJA` -elementtiesiintymät sijoitetaan tuloksessa `KIRJAT` -juurielementin sisään jälkimmäisellä CONSTRUCT -lauseella ’`CONSTRUCT $k`’. Siinä sijoitetaan muuttuja `$k`:n alustukset.

Haluttaessa edellistä kuvan 8 kyselyä voidaan tarkentaa siten, että kyselyn tulos sisältää tietyn kirjoittajan teokset. Esimerkki tästä on esitetty kuvassa 9, jossa tuloksen halutaan sisältävän vain Mitchellin kirjoittamat teokset.

XML-QL esimerkkikysely 2:

```
CONSTRUCT
    <Mitchell> {
        WHERE
            <KIRJASTO1>
                <$k><$h>
                    <SNIMI>Mitchell</SNIMI>
                </></> ELEMENT_AS $m
            </KIRJASTO1>
        CONSTRUCT $m }
    </Mitchell>
```

Tulos:

```
<Mitchell>
   <KIRJA>
     <NIMI>Tuulen viemää</NIMI>
     <KIRJOITTAJA>
       <ENIMI>Margaret</ENIMI>
       <SNIMI>Mitchell</SNIMI>
     </KIRJOITTAJA>
   </KIRJA>
</Mitchell>
```

---

9. XML-QL -esimerkkikysely 2 *kirjasto1.xml* -dokumentista

Tässä toisessa XML-QL -kyselyssä kuvassa 9 käytettävä muuttuja `$k` merkitsee mitä tahansa `KIRJASTO1` -nimisen elementtiesiintymän alaelementtiesiintymää. Muuttuja `$h` puolestaan merkitsee mitä tahansa muuttujaan `$k` liittyvää alaelementtiesiintymää. Tässä tapauksessa muuttujat vastaavat siis `KIRJA`- (muuttuja `$k`) ja `KIRJOITTAJA` (muuttuja `$h`) -elementtien esiintymiä. Koska kyselyn kohteena oleva dokumentti '*kirjasto1.xml*' sisältää pelkästään kirjaston kirjoja, ei tätä ominaisuutta voida täydellisesti hyödyntää tässä esimerkissä. Mikäli esimerkkidokumentti sisältäisi `KIRJA` -elementtiesiintymien lisäksi myös muunlaisia elementtiesiintymiä, kuten esimerkiksi `LEHTI` -elementtiesiintymiä, voitaisiin muuttujalla `$k` viitata saman aikaisesti molempiin elementtityyppeihin.

Tässä esimerkkikyselyssä on hyödynnetty myös kulmasulkutunnisteita ’</>’, jotka tässä sulkevat muuttujat sisältävät tunnisteet. Asetettuja muuttujia vastaavien elementtiesiintymien sisäiselle `SNIMI` -elementtiesiintymälle asetetaan ehto, jonka mukaan tulokseen otetaan vain ne elementit, joilla on ’`Mitchell`’ kirjoittajan sukunimen arvona (eli `SNIMI` -elementin sisältönä). Löydetyt tuloselementit sidotaan muuttujaan `$m`, jota käytetään sitten jälkimmäisessä CONSTRUCT -ilmauksessa, jossa varsinainen tulos sijoitetaan tunnisteiden ’`<Mitchell>`’ ja ’`</Mitchell>`’ sisään.

Oletetaan, että esimerkkidokumentti ’*kirjasto1.xml*’ sisältäisi `KIRJA` -elementtiesiintymien lisäksi myös vastaavalla elementtien sisäkkäisyyden tasolla olevia `LEHTI` -elementtiesiintymiä. Toisin sanoen, esimerkkidokumentin puuvisualisoinnissa `KIRJA`- ja `LEHTI` -elementtiesiintymät olisivat keskenään rinnakkaisia sisarsolmuja. Oletetaan lisäksi, että joidenkin näistä `LEHTI` -elementtiesiintymistä kirjoittajan nimenä (etunimenä tai sukunimenä) on Mitchell. Nyt voidaan ilmaista edellinen kuvassa 9 esitetty kysely eri tavalla. Kuvassa 10 on tähän liittyvä XML-QL -esimerkkikysely.

XML-QL esimerkkikysely 3:

```
WHERE <$y>
           <NIMI>$n</NIMI>
           <KIRJOITTAJA>
               <$k>Mitchell</>
           </KIRJOITTAJA>
      </> $y IN {KIRJA, LEHTI}

CONSTRUCT <$y>
            <NIMI>$n</NIMI>
            <$k>Mitchell</>
          </>
```

Tulos:

```
<KIRJA>
   <NIMI>Tuulen viemää</NIMI>
   <SNIMI>Mitchell</SNIMI>
</KIRJA>

<LEHTI>
   <NIMI>Lehden nimi</NIMI>
   <ENIMI>Mitchell</ENIMI>
</LEHTI>

...
```

10. XML-QL -esimerkkikysely 3 *kirjasto1.xml* -dokumenttiin perustuen, jossa on myös kuvitteellisia `LEHTI` -elementtiesiintymiä

Kyselyiden muodostaminen XML-QL:llä XML-tiedoista on helppoa sellaiselle henkilölle, joka hallitsee XML:ää ja etenkin siihen perustuvan mallinsovituksen. XML-QL -kielen käyttöön nimittäin tarvitaan tietoa mm. XML-syntaksista, mikä ei tavallisten loppukäyttäjien mielestä ole miellyttävä merkintätapa. Ihanteellista olisi, jos loppukäyttäjän ei tarvitsisi hallita XML-syntaksia lainkaan. Vaikka XML-QL onkin omalla tavallaan erittäin joustava lähestymistapa, kyselyistä kuitenkin tulee usein melko pitkiä. Tämän luvun suhteellisen yksinkertaiset esimerkkikyselyt ovat tästä hyvä esimerkki. Mikäli monimutkaisuutta kyselyihin lisättäisiin, tulisi kyselyistä heti huomattavasti pidempiä. XML-QL -kielen käyttäjän on siis kyettävä ymmärtämään XML-syntaksia ja osattava alustaa muuttujat mallinsovituksen pohjalta käyttääkseen sitä tehokkaasti. [Erwig, 2003]

5.2. RXQL

RXQL-kyselykielen perustana on XML-relaatioesitys, joka esittää minkä tahansa XML-dokumentin relaationa. XML-relaatiolla on kaavio D(C, T, I). Siinä D on XML-dokumentin nimi, C on XML-komponentti (ts. elementin nimi, attribuutin nimi tai jommankumman arvo). T ilmaisee kyseisen XML-komponentin (C:n) tyypin ja I on sen yksikäsitteinen sijainti-indeksi XML-hierarkiassa.

RXQL-kielessä kyselyiden muodostaminen on deklaratiivista, ts. käyttäjän ei tarvitse käyttää XML-merkintätapaa, toistuvia (iteratiivisia) rakenteita tai polkuilmauksia lainkaan, toisin kuin tyypillisissä XML-kielissä [Näppilä et al., 2010]. Käyttäjän ei myöskään tarvitse hallita proseduraalista muuttujakäsitettä kyselyiden muodostuksessa.

RXQL-kyselykieli tukee tiedon poimintaa, valintaa, järjestämistä, ryhmittelyä, aggregointia, uudelleenstrukturointia ja uudelleen nimeämistä. Se tukee myös tulosten konstruointia ja tiedon harmonisointia. Näihin liittyvät toiminnot ovat myös käyttäjälle näkymättömissä. Koska kyselykieli on kohdistettu loppukäyttäjälle, ovat käytetyt ilmaisut suhteellisen korkealla abstraktiotasolla. Kielen perustana oleva XML-relaatioesitys, eli D(C, T, I), on yhteensopiva perinteisen relaatiomallin kanssa, joten RXQL:n toteuttamisessa voidaan hyödyntää osittain olemassa olevien relaatiotietokantajärjestelmien tiedon prosessointi- ja tallennusominaisuuksia.

XML-relaatio on aina poikkeuksetta kolmipaikkainen relaatio D(C, T, I), toisin kuin tavallisten relaatiotietokantojen relaatiot, joiden astelukua tai muotoa ei ole tällä tavoin rajoitettu. Toinen oleellinen ero XML-relaation ja relaatiotietokantojen relaatioiden välillä on se, että relaatiotietokannoissa monikko (eli relaation rivi) koostuu vain ilmentymätason tiedoista. Ilmentymätason tiedot ovat attribuuttien arvoja. XML-relaatioesityksessä puolestaan kolmikon ensimmäinen komponentti ’C’ voi kuulua joko kaaviotasolle (elementtien ja attribuuttien nimet) tai ilmentymätasolle (elementtien ja attribuuttien arvot) [Näppilä et al., 2010]. Näiden lisäksi XML-relaatioesityksen kolmas komponentti ’I’ sisältää järjestystiedot (eli ns. rakenneindeksin), joka voi liittyä kumpaan tahansa tasoon riippuen komponentin ilmentymästä [Näppilä et al., 2010]. Täten sitä ei voida tulkita kuuluvaksi tiettyyn tasoon.

RXQL:n syntaksissa kohdedokumentin kuvaus, tiedon poiminnan määrittely ja mahdollinen valintaosa ja aggregointiosa ovat selkeästi eroteltuna toisistaan. Tämä ei ole XML-kielten tyypillinen ominaisuus, sillä esimerkiksi XQuery-kielessä kohderakenteen poimintaan ja tiedon poimintaan sekä tiedon valintaan ja aggregointiin liittyvät ilmaukset ovat sekoitettuna keskenään. Tämä johtaakin usein suuriin ja monimutkaisin kyselymäärittelyihin. RXQL-kyselykielen käyttäjän ei tarvitse tietää tai huolehtia siitä, miten tulosdokumentti rakennetaan ohjelmallisesti. Kyselyssä täytyy vain antaa halutun XML-kohdedokumentin rakenne sisältöineen. Lisäksi käyttäjän on tunnettava dokumentissa olevien komponenttien nimet, joihin tuloksen rakentaminen

perustuu. Tarvittaessa hänen on mahdollista antaa myös tiedon valintakriteerit sekä tiedon poimintakriteerit. [Näppilä et al., 2010]

RXQL-kyselykielen syntaksi sisältää vain muutamia varattuja sanoja. Varattuja sanoja kyselyssä ovat CONSTRUCT, FROM, WHERE, GROUP BY ja ORDER BY. Näistä pakollisia osia kaikille kyselyille ovat CONSTRUCT -ilmaus ja FROM -ilmaus. Loput varattuihin sanoihin liittyvät ilmaukset ovat valinnaisia.

Kyselyn CONSTRUCT -osassa määritellään kohdedokumentin rakenne. Kohdedokumentin rakenteen määrittelyssä käyttäjä nimeää juurielementin, jonka lapset ilmaistaan sen perässä sulkeilla suljettuina ja kaikki mahdolliset alaelementit (mikäli niitä siis on) merkitään samaan tapaan sulkeiden sisään.

Kyselyn FROM -osa ilmaisee mistä dokumentista ja mistä dokumentin osasta tiedot poimitaan tulokseen. WHERE -ilmauksella kuvataan ehdot tietojen valinnalle ja poiminnalle. GROUP BY ilmaisee tietojen ryhmittelyn kriteerit ja samankaltainen ORDER BY -ilmaus määrittelee mitä tietoja käytetään perusteena elementtien ilmentymien esiintymisjärjestyksen määräämisessä tuloksessa.

Attribuutit erotetaan kyselyssä elementeistä merkinnällisesti siten, että niiden etuliitteenä käytetään merkkiä ’@’. Jos elementin etuliitteenä käytetään merkkiä ’+’, se tarkoittaa, että kyseisellä elementillä voi olla tuloksessa useampi kuin yksi esiintymä suhteessa elementtiesiintymään, josta se riippuu.

Kuvissa 11 ja 12 annetaan esimerkkejä RXQL-kyselyiden tekemisestä käyttäen edellä mainittuja varattuja sanoja ja niistä muodostettuja ilmauksia. Esimerkkikyselyt perustuvat luvussa 2 annettuun esimerkkidokumenttiin *kirjasto1.xml* (kuva 1).

RXQL –esimerkkikysely 1:

```
CONSTRUCT TEOKSET (+KIRJA (@JULKAISUVUOSI, NIMI, ISBN,
KIRJOITTAJA(ENIMI, SNIMI)))
FROM kirjasto1.xml[KIRJA, JULKAISUVUOSI, NIMI, ISBN,
KIRJOITTAJA, ENIMI, SNIMI]
```

Tulos:

```
<TEOKSET>
   <KIRJA>
     <NIMI>Tuulen viemää</NIMI>
     <KIRJOITTAJA>
       <ENIMI>Margaret</ENIMI>
       <SNIM>Mitchell</SNIMI>
     </KIRJOITTAJA>
   </KIRJA>

   <KIRJA JULKAISUVUOSI = ”1991”>
     <NIMI>Scarlett</NIMI>
     <ISBN>951-1-11012-8</ISBN>
     <KIRJOITTAJA>
       <ENIMI>Alexandra</ENIMI>
       <SNIMI>Ripley</SNIMI>
     </KIRJOITTAJA>
   </KIRJA>
</TEOKSET>
```

11. RXQL-esimerkkikysely 1 ja sen tulos

Kuvissa 11 ja 12 annettujen RXQL-esimerkkikyselyjen tulosdokumenttien juurielementeiksi molemmissa tapauksissa määritellään `<TEOKSET>`. Kuvan 11 esimerkkikysely poimii kaikki `KIRJA` -elementtiesiintymät dokumentista *kirjasto1.xml* ja kokoaa ne `CONSTRUCT` -osassa tulosdokumentiksi. Kuvan 11 tapauksessa tulosdokumentti ei eroa alkuperäisestä dokumentista juuri muuten kuin juurielementin nimessä. Lisäksi tuloksesta toki puuttuu alkuperäisdokumentissa esitetty `KIRJASTON_NIMI` -osa myös. Huomattavaa tässä on mm. se, ettei FROM -osassa ole tarpeen ilmaista elementtien rakennetta sulkumerkein, vaan ilmaista niiden tietoalkioiden nimet, jotka halutaan poimia kyselyn tulokseen.

RXQL -esimerkkikysely 2:

```
CONSTRUCT teokset (+teos (@nimike, tekijä))
FROM kirjasto1.xml[KIRJA AS teos, NIMI AS nimike, SNIMI AS
tekijä]
WHERE tekijä = Mitchell
```

Tulos:

```
<teokset>
   <teos nimike = ”Tuulen viemää”>
     <tekijä> Mitchell </tekijä>
   </teos>
</teokset>
```

12. RXQL-esimerkkikysely 2 ja sen tulos

Kuvassa 12 annetun esimerkkikyselyn tulosdokumentti koostuu `teos` –elementtien (joita voi olla yksi tai useampia) esiintymistä, joilla kullakin on attribuuttina `nimike` ja alaelementtinä `tekijä`. FROM -osassa poimitaan elementtiesiintymät elementeistä `KIRJA`, `NIMI` ja `SNIMI` dokumentista *kirjasto1.xml*. Tulosdokumenttiin nimetään uudelleen nämä elementtiesiintymät `KIRJA`, `NIMI` ja `SNIMI` AS -ilmauksella. Uudelleen nimetty `KIRJA` -elementti on nyt tulosdokumentissa elementtinimellä `teos`, `NIMI` on `nimike` ja `snimi` on `tekijä`. Elementti ’`nimike`’ ilmaistaan tulosdokumentissa attribuuttina, kun se lähtödokumentissa esitettiin elementtinä. WHERE -osassa määritellään mukaan otettaville elementtiesiintymille se ominaisuus, että niiden tekijänä on Mitchell.

RXQL-kyselyitä muodostettaessa käyttäjän ei tarvitse tietää eikä määritellä navigointipolkuja vaan pelkästään tarvittavien komponenttien nimet. Kyselyissä aggregointi toteutetaan käyttämällä saatavilla olevia aggregointifunktioita. Näitä funktioita ovat COUNT, SUM, AVG, MIN ja MAX. Aggregointifunktioilla voidaan siis laskea lukumääriä, summia, keskiarvoja, minimejä ja maksimeja. Tiedon valinnassa tavalliset valintaoperaatiot ’=’, ’!=’, ’<’, ’>’, ’<=’ ja ’>=’ ovat saatavilla.

GROUP BY tai ORDER BY -lauseessa käytettävien komponenttien nimien on esiinnyttävä joko FROM -lauseessa tai WHERE -lauseen aggregointi-ilmauksissa. GROUP BY -ilmausta käytetään komponenttien ilmentymien ryhmittelyyn tulosdokumentissa perustuen annettuihin komponentteihin. ORDER BY -lauseessa määritellään elementti-ilmentymien järjestys joko laskevaksi termillä DESC tai nousevaksi termillä ASC. Oletuksena on nouseva järjestys. [Näppilä et al., 2010]

RXQL-kyselykieli on siis SQL-tyyppinen kyselykieli. Sillä ei ole XML-syntaksia eikä käyttäjän siten tarvitse osata XML:ää. RXQL-kieli tukee kyselyiden muodostamista

deklaratiivisella tavalla. Deklaratiivisessa kyselymuodostuksessa käyttäjän ei tarvitse hallita proseduraalista ohjelmointia, mallinsovitustekniikoita tai muuttujakäsitettä. RXQL kykenee harmonisoimaan heterogeenisiä tietolähteitä, joten tiedon tehokas ryhmittely ja koonti on mahdollista.

5.3. XML-tiedon konfliktit

Kun kahdessa tai useammassa tietolähteessä samaa tarkoittava tieto esitetään eri tavoin tai kun eri asioita tarkoittavat tieto esitetään keskenään samalla tavalla, on kyse tiedon semanttisesta heterogeenisyydestä. Tiedon harmonisoinnin tarkoituksena on poistaa heterogeeniset tekijät kun tietoa poimitaan eri tietolähteistä. Eli harmonisoinnilla esitetään eri tietolähteissä esiintyvä sama tieto yhtenäisellä tavalla. Tiedon harmonisointia tarvitaan käytännössä sellaisissa sovelluksissa, joissa on käsiteltävä sellaisia tietolähteitä, joiden yhteiskäyttöisyyttä ei ole voitu ennakoida. Tällöin niiden sisällöt, rakenteet ja nimeämiskäytännöt poikkeavat toisistaan. [Niemi et al., 2009]

Tiedon harmonisoinnin suurena haasteena on ratkaista tietojen välillä vallitsevat konfliktit. Useita eri heterogeenisyyden tyyppejä voi ilmetä eri tietolähteiden välillä. Heterogeenisyyden muodot luokitellaan konfliktityypeiksi luokitellaan relaatiotietokantojen yhteydessä. Niemi ja kumppanit [Niemi et al., 2009] ovat laajentaneet tätä luokittelua ja mukauttaneet ne soveltumaan XML-tiedolle. XML-tiedon konfliktityyppejä on viittä eri tyyppiä.

1. *Arvojen väliset konfliktit* ilmenevät kun saman XML-attribuutin tai XML-elementin sisältöjä ei esitetä yhtenevällä tavalla kaikissa eri tietolähteissä. Esimerkiksi jos eri lähteissä käytetään eri mittayksiköitä ilmaisemaan tietoalkioiden arvoja.
2. *Attribuuttien ja elementtien väliset konfliktit* ilmenevät, kun sama tietoalkio ilmaistaan attribuuttina yhdessä tietolähteessä ja elementtinä jossain toisessa tietolähteessä. Tälle konfliktityypille ei voida löytää vastinetta relaatiotietojen yhteydessä.
3. *Dokumenttien väliset konfliktit* ilmenevät, kun sama tietosisältö on ilmaistu eri tietolähteissä erilaisten attribuuttirakenteiden tai elementtirakenteiden perusteella tai mahdollisesti jonkin tietolähteen XML-dokumentin tiedot on esitetty useana XML-dokumenttina toisessa tietolähteessä.
4. *Arvojen ja attribuuttien väliset konfliktit sekä arvojen ja elementtien väliset konfliktit* ilmenevät, kun jonkin tietolähteen attribuuttisisältö tai elementtisisältö esitetään jonkin toisen tietolähteen attribuutin nimenä tai elementin nimenä. Mikäli ristiriitainen elementin nimi kuuluu juurielementtiin, voidaan tätä konfliktityyppiä kutsua arvon ja dokumentin väliseksi konfliktiksi. Tämä vastaa arvon ja taulukon nimen

välistä konfliktia relaatiotietokantojen yhteydessä, koska XML:ssä juurielementin nimeä voidaan käyttää dokumentin tunnistamiseen.

5. *Attribuuttien väliset konfliktit ja elementtien väliset konfliktit* ilmenevät, kun tietoalkiot, eli attribuutit tai elementit, jotka tarkoittavat samaa asiaa, nimetään eri tavoin eri tietolähteissä. Konflikti ilmenee myös, kun tietoalkiot, jotka merkitsevät eri asioita, nimetään eri tietolähteissä samalla tavalla keskenään (eli identtisesti).

Nämä eri konfliktityypit edustavat erilaisia heterogeenisyyden tyyppejä. Esimerkiksi arvojen väliset konfliktit edustavat esityksellistä heterogeenisyyttä. Attribuuttien ja elementtien väliset konfliktit sekä dokumenttien väliset konfliktit puolestaan edustavat rakenteellista heterogeenisyyttä. Arvojen ja attribuuttien väliset konfliktit sekä arvojen ja elementtien väliset konfliktit edustavat semanttista heterogeenisyyttä, kuten myös viimeinen konfliktityyppi: attribuuttien väliset konfliktit ja elementtien väliset konfliktit. Erittäin heterogeeniset ja itsenäiset tietolähteet sisältävät tyypillisesti useita näistä eri konfliktityypeistä. XML-tiedon harmonisoinnin tarkoituksena on poistaa kaikki rakenteelliset ja semanttiset konfliktit eri tietolähteiden väliltä. [Niemi et al., 2009]

6. Visuaaliset XML-kyselykielet

Martin Erwig [Erwig, 2003] on määritellyt suunnittelutavoitteita sellaiselle XML-kyselykielelle, joka on erityisesti tarkoitettu ohjelmointitaidottomille loppukäyttäjille. Hänen mukaansa ei ole tarvetta määritellä jälleen yhtä uutta tekstuaalista XML-syntaksiin pohjautuvaa kyselykieltä. Tämä johtunee loppukäyttäjien haasteista oppia ja tulkita monimutkaisia syntaktisia merkintöjä, jotka haittaavat tiedon luettavuutta ja tulkitsemista. Erwig ehdottaa sen sijaan XML-tietojen visualisointia graafisten merkintöjen avulla.

Erwigin [Erwig, 2003] mukaan olisi suotavaa käyttää mallinsovitusta kyselyjärjestelmän perustana. Tätä voitaisiin hänen mielestään tukea osoita-ja-klikkaa -käyttöliittymien muodossa. Koska on kuitenkin mahdollista suunnitella sellaisiakin XML-kieliä, joissa mallinsovituspiirre on käyttäjältä piilotettu, voidaan tämä väite kyseenalaistaa. Erwigin mielestä kyselyjärjestelmän tulisi myös olla mahdollisimman yksinkertainen. Tämä voisi toimia siten, että tehtäisiin yksinkertaisista kyselyistä mahdollisimman helppoja ja vaadittaisiin enemmän työtä ja käsitteiden hallintaa kehittyneempien kyselyiden ilmaisuun. Poimintaa ja valitsemista käyttävien kyselykielten kyselyissä tämä jo hyvin pitkälti toteutuukin melkein kielen kuin kielen tapauksessa.

Nämä edellä esitellyt suunnittelutavoitteet toimivat Erwigin mukaan hyvin, kun tavoitteena on visuaalinen XML-kyselykieli. Ne toimivat hyvin myös kun halutaan lähestyä uuden kyselykielen suunnittelua täysin uudelta ja erilaiselta pohjalta kuin

perinteisten tekstipohjaisten XML-kielten tapauksessa. Nämä suunnittelupohdinnat johtavat sääntöpohjaisen visuaalisen XML-kielen valintaan, jossa kehitettävänä oleva kieli perustuisi yksinkertaiseen XML-tietojen visualisointiin. Erwig käyttää tästä yksinkertaisesta XML-tiedon visualisoinnista termiä dokumenttimetafora. Hyvin heterogeenisten tietojen käsittelyn valossa, visualisointi ei välttämättä toimi kovin helposti. Yleisesti kyselykielen ilmaisuvoimaa arvioidaan sen pohjalta, mitä piirteitä se tukee, esimerkiksi tiedon aggregointia, harmonisointia ja ryhmittelyä.

Erwig [Erwig, 2003] mainitsee kaksi näkökulmaa, joita voidaan käyttää perusteena visuaalisen merkintätavan suosimisessa tekstuaalisen sijaan. Ensinnäkin, kaikki Internetin käyttäjät voidaan nähdä potentiaalisina XML-tiedon käyttäjinä ja näin siis myös kyselykielen loppukäyttäjinä. Tämän vuoksi on haasteellista suunnitella tehokkaita mutta samalla helppokäyttöisiä kieliä XML:n käsittelyyn. Ei nimittäin voida yleisesti olettaa, että loppukäyttäjän kykenevät tai ovat edes halukkaita opettelemaan erikoistuneiden XML-kyselykielten käyttöä. Koska ei voida olettaa, että loppukäyttäjillä on mitään tietoja XML-syntaksista, DTD:istä, XML:n kyselykielistä tai käsittelykielistä millään tasolla, on loppukäyttäjille näkyvän kyselyrajapinnan piilotettava kaikki näiden tekniset piirteet.

Toinen perustelu visuaalisen merkinnän hyväksi Erwigin mukaan on se, että hakusanoihin perustuvat hakukoneet eivät ole tarpeeksi tehokkaita hyödyntämään XML-formaatin antamaa tiedon rakennetta. Hakusanajärjestelmät on sellaisenaan tarkoitettu ihan eri tarkoitusperiä varten. Loppukäyttäjälle tarkoitetun kyselyjärjestelmän olisi vältettävä muodollisen kielen (varattujen sanojen, muuttujien yms.) käyttämisen vaatimista käyttäjältä tai vähintäänkin vältettävä niiden paljastamista käyttäjälle. Relaatiotietokannoista saadut kokemukset ovat osoittaneet, että lomakeperustaiset kyselyt ja päivitystoiminnot ovat, esimerkiksi SQL:n sijaan, soveltuvampia loppukäyttäjien käyttöön [Erwig, 2003].

Visuaaliset kyselykielet on suunniteltu sillä periaatteella, että niiden käyttö olisi mahdollista myös sellaisille käyttäjille, joilla ei ole laajaa perustietämystä ohjelmoinnista, XML-syntaksista tai välttämättä edes tietokantojen saati tietokoneen kattavasta käytöstä. Useimmat visuaaliset kyselykielet, kuten Xing [Erwig, 2003] tai XML-GL [Ceri et al., 1999], käyttävät merkintätavan ilmaisemiseen esittämällä dokumentteja lomakemaisina sisäkkäisinä suorakulmiomalleina. Tämä visualisointi tukee käytettävyyttä, koska tiedon rakenteet tuodaan selkeästi esille. Koska XML-tieto voidaan esittää lähtökohtaisesti suunnattuna ja nimettynä puurakenteena, voisi olla luontevaa visualisoida tämä rakenne puuna. Puurakenne voi olla visuaalisena hierarkiana kuitenkin haasteellisempi käsittää melko suoraviivaiseen laatikkomalliin verrattuna. Suurten tietomäärien, laajojen dokumenttien ja tiedon heterogeenisyyden kasvaessa voi visualisoinnista kuitenkin tulla hankalasti tulkittavaa. Visuaalisten

kyselykielten voidaan siis ajatella soveltuvan vain tietyn kokoisten (pienten, keskisuurten) tietomäärien käsittelyyn.

VQL on esimerkiksi eräs visuaalinen kyselykieli. VQL [Vadaparty et al., 1993] (Visual Query Language) on lomakepohjainen kieli, jonka tarkoituksena on toimia yleisenä tietokantakielenä. VQL on sääntöpohjainen kieli ja visuaaliselta muotoilultaan melko tavalla samankaltainen kuin esimerkiksi Xing [Erwig, 2003] (ks. alaluku 6.2). Tämä johtuu siitä, että molemmat kielet käyttävät sisäkkäisiä suorakulmioita tiedon esittämiseen ja kyselyiden muodostamiseen. VQL on strukturoituun tietomalliin perustuva kyselykieli.

VQL-kyselykielessä on mahdollista tehdä kyselyitä myös kaavioissa (engl. schema) olevia tietoja koskien. Tämä on tärkeä ominaisuus, etenkin jos ja kun käyttäjällä on kaaviosta vain rajallinen määrä tietoa tai ei tietoa lainkaan. Xing:issä tämä on mahdollista, jos käytetään säännöllisiä lausekkeita (engl. regular expressions) tunnisteina (engl. tag) [Erwig, 2003]. Säännöllisellä lausekkeella tarkoitetaan sellaisia ilmauksia, jotka kirjoitetaan jollain muodollisella kielellä, jolla etsitään tiettyjä ehtoja käyttäen vastaavia ilmauksia merkkijonoille. Tarkastellaan siis kuuluuko jokin merkkijono kyseisen lausekkeen määrittelemään kieleen. Säännöllisten lausekkeiden käyttö lisää haasteellisuutta kyselykielen käyttöön, joten loppukäyttäjän kannalta on parempi välttää niiden käytön vaatimista. VQL ei kuitenkaan kykene ilmaisemaan kovinkaan ’syviä’ kyselyitä [Erwig, 2003]. Syvillä kyselyillä tarkoitetaan sellaisia kyselyitä, joissa haettava tieto sijaitsee dokumentissa ’syvällä’, usean sisäkkäisen hierarkiatason takana.

6.1. XML-GL

XML-GL on myös eräs visuaalinen kyselykieli, joka perustuu graafiseen tietomalliin. Siinä XML-tiedot esitetään graafisten symbolien avulla, jotka esittävät tietoalkioita, niiden ominaisuuksia ja niiden keskinäisiä suhteita. XML-GL ei perustu XML:ään ja se on kyselykielten tapaan tarkoitettu tiedon poimintaan ja valintaan XML-dokumenteista. Voidaan siis sanoa, että XML-GL -kielen ilmaisuvoima on kovin rajoittunutta. XML-GL -kyselykieli olettaa automaattisesti, että siihen XML-dokumenttiin, johon perustuen kyselyitä tehdään, on liitetty DTD, tai että sen rakenteet ovat vähintäänkin säännöllisiä. Toisin sanoen, rakenteellista heterogeenisyyttä ei sallita käsiteltävässä XML-dokumentissa [Näppilä et al., 2010].

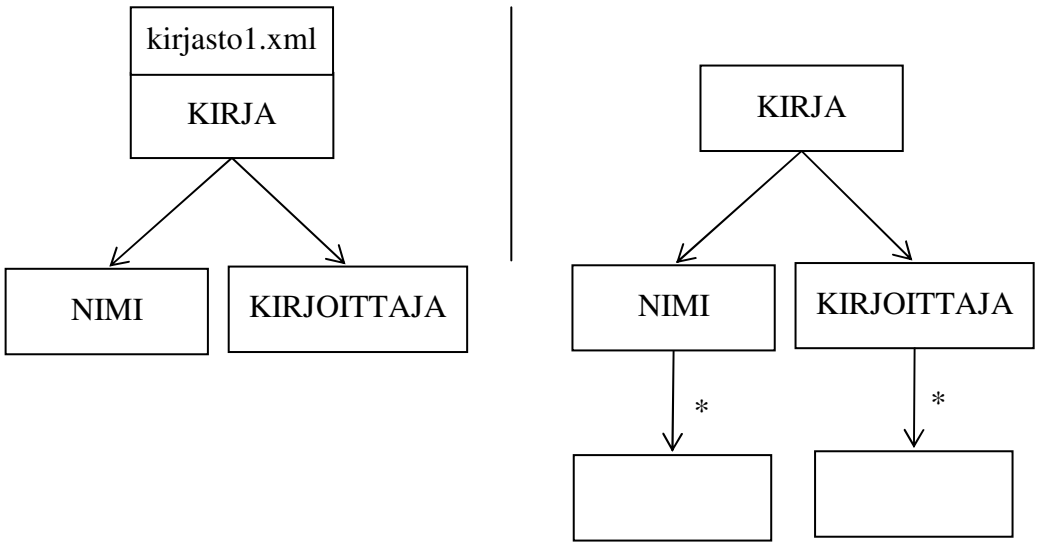
XML-GL -kysely koostuu neljästä osasta. *Poimintaosassa* (engl. extract part) tunnistetaan kyselyn laajuus osoittamalla sekä kohdedokumentit että kohde-elementit dokumenttien sisällä, eli käyttäjällä on valikoima lähtödokumentteja, joiden joukosta hän osoittaa sen dokumentin (kohdedokumentti), josta hän on kyselyssään kiinnostunut. Valinnaisessa *yhteensovitusosassa* (engl. match part) määritellään ne loogiset ehdot, jotka kohde-elementtien on tyydytettävä voidakseen olla osana kyselyn tulosta.

Leikkausosassa (engl. clip part) määritellään ne alaelementit poimituista elementeistä, jotka tyydyttävät yhteensovitusosan määritteet, sisällytettäväksi tulokseen. Valinnaisessa *rakennusosassa* (engl. construct part) määritellään ne uudet elementit, jotka liitetään tulosdokumenttiin ja niiden suhteet poimittuihin elementteihin. Sama kysely voidaan muodostaa käyttämällä eri rakenneosia, jotta saataisiin tulos eri tavalla muotoiltua. [Ceri et al., 1999]

Yksinkertaisin XML-GL -kyselyn muoto on *poiminta-leikkaus*, jossa siis käytetään poimintaosaa ja leikkausosaa. Se poimii osan XML-dokumentista ja tuottaa tuloksena uuden dokumentin, joka sisältää poimitut tiedot. Kysely esitetään graafisena 'kaaviona'. Poiminta-leikkaus -kyselyissä kaavion vasen puoli sisältää kyselyn poimintaosan. Kyselykaavion oikea puoli puolestaan sisältää leikkausosan, joka määrittelee tulosdokumentin DTD:n poimintaosassa määriteltyjen kohde-elementtien rakenteesta.

Kuvan 13 esimerkissä poimintaosa käsittelee kohde-elementtiä '*KIRJA*' sekä sen alaelementtejä '*NIMI*' ja '*KIRJOITTAJA*'. Muut mahdolliset '*KIRJA*' -elementin alaelementit jätetään huomioimatta. Esimerkissä kyselyn vasemmalla puolella poimitaan KIRJA, NIMI ja KIRJOITTAJA -elementtiesiintymät *kirjasto1.xml* -dokumentista (kuva 1). Merkki '*' kyselyn oikealla puolella merkitsee, että tulokseen liitetään kyseisten elementtien alaelementtien kaikki sisäkkäisyystasot. Taustaolettamuksena kuvan 13 kyselylle on se, että XML-esimerkkidokumentille on laadittu DTD-kuvaus.

XML-GL -esimerkkikysely:



Tulos:

```
<KIRJA>
   <NIMI>Tuulen viemää</NIMI>
   <KIRJOITTAJA>
     <ENIMI>Margaret</ENIMI>
     <SNIMI>Mitchell</SNIMI>
   </KIRJOITTAJA>
</KIRJA>

<KIRJA julkaisuvuosi = ”1991”>
   <NIMI>Scarlett</NIMI>
   <KIRJOITTAJA>
     <ENIMI>Alexandra</ENIMI>
     <SNIMI>Ripley</SNIMI>
   </KIRJOITTAJA>
</KIRJA>
```

13. XML-GL -esimerkkikysely: kirjojen nimien ja kirjoittajien poiminta *kirjasto1.xml* -dokumentista

6.2. Xing

Xing tulee englannin kielen sanoista XML in Graphics ja se lausutaan ’crossing’. Englannin kielinen nimitys tarkoittaa vapaasti käännettynä ’XML grafiikkana’. Xing on eräs visuaalinen XML-kyselykieli, jolla voidaan tehdä kyselyitä XML-dokumentteihin perustuen. Xing -kielen kyselyiden tekeminen perustuu dokumenttimalliin (engl.

document pattern), mallinsovitukseen ja sääntöihin. Xing on tarkoitettu erityisesti loppukäyttäjille ja se pyrkii välttämään varsinaista XML-syntaksia. Käyttäjän ei myöskään oleteta hallitsevan muuttujakäsitettä, joten Xing:iä voidaan pitää melko deklaratiivisena ja ilmaisuvoimaisena visuaalisena kyselykielenä. Xing tukee mallinsovitustekniikoiden osoita-ja-klikkaa –rajapintaa, tiedon aggregointia, ryhmittelyä ja järjestämistä.

Toisin kuin esimerkiksi XML-QL -kielessä, jossa kyselyistä tapaa tulla melko pitkiä ja jota käyttääkseen on ymmärrettävä sujuvasti XML-syntaksia, Xing:issä visuaalinen dokumenttimalli antaa mahdollisuuden muodostaa kyselyitä yksinkertaisesti lomakepohjaisen käyttöliittymärajapinnan kautta. XML-tietojen käsittely ja rakenteiden uudelleen organisoiminen Xing:issä tapahtuu käyttämällä sääntöjä. Kuitenkin esimerkiksi Näppilän ja kumppaneiden [Näppilä et al., 2010] mukaan XML-kyselykielten tulisi perustellusti olla sen verran korkealla abstraktion tasolla, että tiedon uudelleen organisointi on automaattista eikä näy käyttäjälle.

Loppukäyttäjät voivat muodostaa Xing-kyselyitä vain niiden visuaalisten tietojen pohjalta, jotka hänelle dokumenteista on saatavilla. Xing:in dokumenttimallit ovat visuaalisia esityksiä XML-dokumenteista. Säännöt, joita käytetään kyselyiden muodostamiseen ja XML-dokumenttien rakenteiden uudelleen organisoimiseen, esitetään näillä dokumenttimalleilla. Argumenttimalleilla määritellään tiedon valintakriteerit. Tulosmalli puolestaan kuvaa kyselyn tuloksen tietosisällön ja tietorakenteen.

Xing:in visuaalinen esitystapa kuvaa dokumentteja sellaisella tavalla, johon käyttäjät ovat yleisesti tottuneet. Siinä tieto on tyypillisesti jaettu kenttiin, jotka voivat sisältää tekstiä tai muuta rakenteita omaavaa tietoa (esimerkiksi kokoelman muita kenttiä) [Erwig, 2003]. Kyselyt muodostetaan Xing:issä määrittelemällä lomakemalli halutulle lopputulokselle. Käyttöliittymässä tällaista lomakepohjaista visualisointia voitaisiin tukea siten, että käyttäjälle mahdollistettaisiin valintojen tekeminen eri valikoiden kautta [Erwig, 2003].

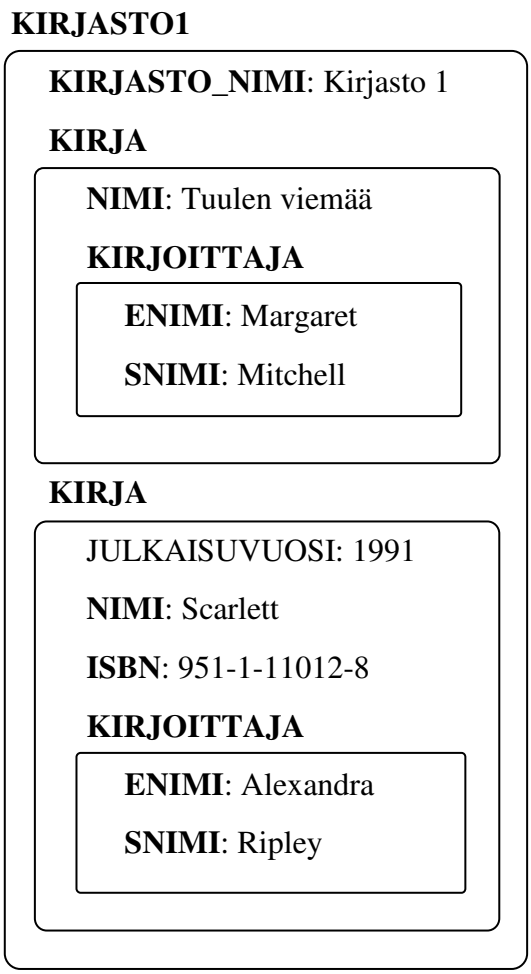
Xing perustuu dokumenttimetaforaan, jossa XML-dokumentin visualisointi on täydellinen. Täydellisyys merkitsee siis sitä, että visualisointi vastaa täysin tekstimuotoista XML-dokumenttia sisällöltään ja rakenteiltaan. Xing:in kyselymekanismi ei riipu mahdollisen DTD:n olemassaolosta. Niitä voidaan kuitenkin hyödyntää esimerkiksi käyttöliittymässä dokumenttimallien rakennuksen yhteydessä [Erwig, 2003].

Elementit esitetään laatikoina, joissa elementin nimet ilmaistaan tekstinä suoraan laatikon yläpuolella ja saman elementin sisältö ilmaistaan tämän laatikon sisäpuolella. Elementtien nimet esitetään lihavoituina, jotta ne voitaisiin selkeämmin erottaa muusta varsinaisesta tekstisisällöstä. Pelkkää tekstiä sisältävät elementit merkitään seuraavasti: ’**elementin_nimi**: elementin_tekstisisältö’. Attribuutit merkitään muuten samalla

tavoin, mutta ilman attribuutin nimen lihavointia. Tällainen lomakemainen esitystapa on useimmille ihmisille tuttua [Erwig, 2003]. Erwig ei ota kantaa siihen miten dokumentti silloin visualisoidaan, kun siinä on tuhansia esiintymiä useilla sisäkkäisillä hierarkiatasoilla ja niiden visualisointiin tarvitaan kymmeniä näyttöruutuja.

Tällaista laatikkomuotoista visualisointia voidaan suoraan käyttää myös dokumenttien kyselyssä. Tämä johtuu siitä, että käyttäjä voi suoraan nähdä, miltä dokumentit näyttävät ja osaa tämän perusteella käyttää niitä malleina tiedon poiminnassa. Pelkkää tulosmallia käyttämällä voidaan muodostaa sellaiset säännöt, jotka määrittelevät tuloksen rakenteen ja sisällön.

Kuvassa 14 on Xing -kyselykielellä tehty esitys kuvan 1 XML-esimerkkidokumentista *kirjasto1.xml*. Kuvan 14 esimerkissä havainnollistuu, miten sisäkkäiset elementtiesiintymät Xing:issä visualisoidaan sisäkkäisinä laatikkorakenteina (suorakulmioina).



14. Xing -esitys XML-dokumentista *kirjasto1.xml*

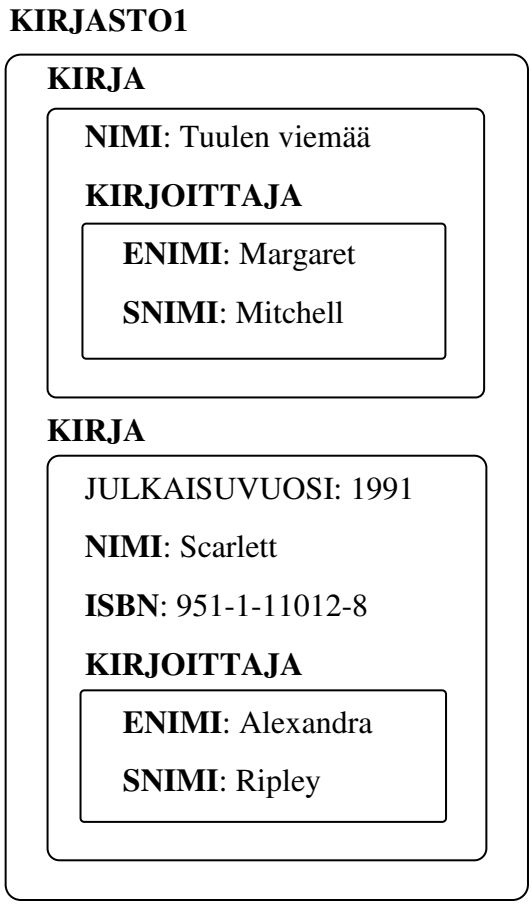
Xing:in visualisoinnissa laatikon otsikko on aina elementin nimi (laatikon sisällä oleviin alielementteihin nähden). Kuvassa 14 esimerkiksi juurielementti on nimeltään **KIRJASTO1**. Juurielementin välittöminä alaelementteinä ovat elementtiesiintymä nimeltä **KIRJASTO_NIMI** ja kaksi **KIRJA** -elementtiesiintymää. Kuvassa jälkimmäisellä **KIRJA** -elementtiesiintymällä on myös attribuutti JULKAISUVUOSI sekä alaelementtiesiintymä **ISBN**, joita dokumentin ensimmäisellä **KIRJA** -elementtiesiintymällä ei ole.

Visuaalinen mallinsovitus on Erwigin [Erwig, 2003] mukaan käyttäjän näkökulmasta hyvin soveltuva lähestymistapa sellaisille kyselykielille, jotka on erityisesti suunnattu vähän pohjatietoja omaaville loppukäyttäjille. Kuvassa 15 on annettu esimerkkikysely Xing -kielellä esitettynä. Tämä esimerkkikysely perustuu kuvan 14 dokumenttiin. Kysely hakee kaikki juurielementin, nimeltä **KIRJASTO1** sisältämät **KIRJA** -alaelementtiesiintymät. Kyselyn alla on annettu myös tulosdokumentin visualisointi. Suhteessa lähtödokumenttiin tulosdokumentti ei sisällä **KIRJASTO_NIMI** -elementin esiintymää.

Käyttäjä voi siis Xing:issä muodostaa kyselyn esittämällä lomakemaisesti sellaisen tulosdokumentin, joka sisältää etsittävän tiedon. Tätä esittämistä voitaisiin tukea käyttöliittymärajapinnassa siten, että kyselyt voidaan ennen pitkää rakentaa perustuen pelkästään valintojen tekemiseen esiin ponnahtavista valikoista [Erwig, 2003]. Käyttäjä voi myös muodostaa lomakkeen sisään toisia lomakkeita sisäkkäisiksi rakenteiksi. Tämä onkin välttämätöntä silloin, kun tulokseksi halutaan rakennetta omaavia dokumentteja.

Xing-esimerkkikysely 1:

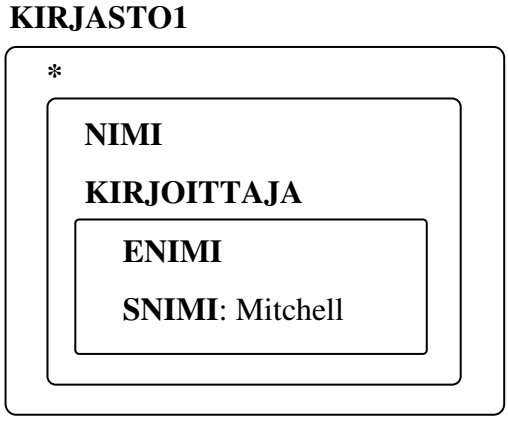
Tulos:



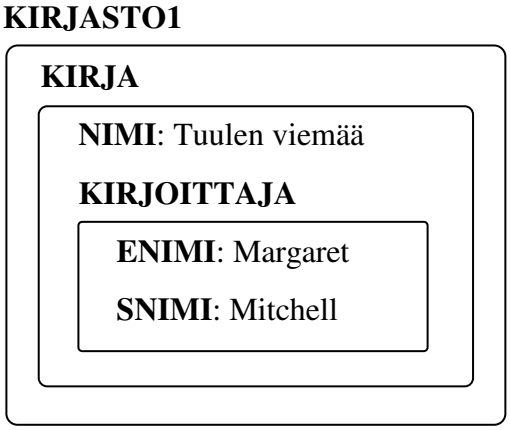
15. Xing -kysely, joka hakee kaikki **KIRJA** -elementit sekä tulosdokumentti

Kuvassa 15 annetussa esimerkkikyselyssä kaikki lienee vielä kohtuullisen suoraviivaista. Kuvassa 16 annetaan toinen esimerkkikysely Xing -kyselykielellä. Siinä haetaan kaikki sellaisten kirjoittajien kirjat, joiden sukunimi on Mitchell. Kyselyssä käytetään myös jokerimerkkiä ’*’ tekemään kyselystä yleisempi. Jokerimerkillä ilmaistaan, että se voidaan korvata minkä tahansa nimisellä elementtiesiintymällä, kunhan muut ehdot täyttyvät. Tämän esimerkin tapauksessa ei kuitenkaan ole mahdollista saada useampia tuloksia, sillä kirjoja, joiden kirjoittajan sukunimi on Mitchell, on kyselyn kohteen olevassa lähtödokumentissa vain yksi.

Xing-esimerkkikysely 2:



Tulos:

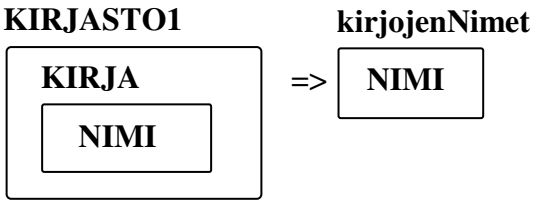


16. Xing-esimerkkikysely 2, jossa haku kohdistetaan tietyn elementin tekstiarvon perusteella sekä kyselyn tulosdokumentti

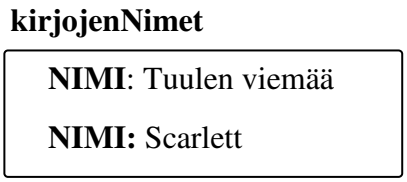
Jokerimerkki ’*’ merkitsee kuvassa 16 annetussa kyselyssä mitä tahansa (minkä nimistä tahansa) elementtiesiintymää, jolla on **NIMI** ja **KIRJOITTAJA** -nimiset alaelementit. Näiden lisäksi on elementillä **KIRJOITTAJA** oltava alaelementit **ENIMI** ja **SNIMI**. Kyselyssä ehdoksi on asetettu myös, että SNIMI -nimisessä elementtiesiintymässä arvon tekstisisällön on oltava ’Mitchell’.

Jokerimerkillä voidaan siis viitata joko useisiin saman elementin esiintymiin tai mihin tahansa eri nimisiin elementteihin samalla dokumenttihierarkian tasolla (kunhan muut ehdot täyttyvät). Kyselyssä nimeltä mainituista elementeistä on myös tulosdokumentissa ilmettävä esiintymiä (yksi tai useampia). Kyselyssä nimeltä mainittuja elementtejä sanotaan alustetuiksi elementeiksi. Sellainen tieto alustetaan, joka kyselyn tekijää kiinnostaa. Tulosdokumentin rakenteelle annetaan tarvittavat ehdot ja rajataan samalla tulosjoukkoa.

Xing-esimerkkikysely 3:



Tulos:



17. Xing-esimerkkikysely 3 ja tuloksen rakenteen muodostaminen uudenlaiseksi sekä kyselyn tulosdokumentti

Kuten aiemmin mainittiin, voidaan Xing -kyselykielessä käyttää dokumenttisääntöjä kyselyiden tulosten rakenteiden uudelleenmuokkaamiseen tai uudelleennimeämiseen. Argumenttimalli yhdistetään tuplanuolella => tulosmalliin. Nämä visuaaliset symbolit muodostavat yhdessä ns. dokumenttisäännön. Kuvassa 17 annetussa esimerkkikyselyssä haetaan kuvassa 14 olevan dokumentin *kirjasto1.xml* Xing-esityksestä kaikkien kirjojen nimet ja listataan ne tulosdokumenttiin uuden juurielementin **kirjojenNimet** alaelementeiksi.

Kuvan 17 kyselyn vasemmalla puolella annetaan XML-lähdedokumentin rakenne poimintaa varten. Eli annetaan tiedot siitä, miten halutut tiedot on järjestetty lähtödokumentissa. Kuvan esimerkin tapauksessa elementtinimen '**KIRJA**' tilalle kävisi myös jokerimerkki '*', sillä lähtödokumentin muilla saman hierarkiatason elementeillä ei ole haussa vaadittua alaelementtiä nimeltä **NIMI**. Tuplanuoli saa aikaan tulosdokumentin mukaisten rakenne-esiintymien konstruoinnin lähtödokumentista. Nuolen oikealla puolella kyselyssä määritellään kyselyn tuloksen rakenne tulosdokumentissa.

Haluttaessa voitaisiin Xing-kielistä kyselyä tarkentaa lisäämällä säännön malliin esimerkiksi pystyviiva '|' jonkin sen sisäisen mallin edelle. Tämä tarkoittaisi, että tätä sisempää mallia tulisi käyttää vain valintaan.

7. Logiikkaohjelmointiin perustuvat XML-kyselykielet

Logiikkaohjelmointikielten ja verkkoteknologioiden yhdistely on mielenkiintoista logiikkaohjelmoinnin sovellettavuuden kannalta. Tämä on kiinnostava tutkimuskenttä erityisesti XML-tietojen prosessoinnin näkökulmasta.

Koska XML-dokumentit ovat standardi formaatti tiedonvaihdon toteuttamiseksi sovellusten välillä, myös loogisten kielten pitäisi kyetä käsittelemään ja tekemään kyselyitä tällaisista dokumenteista. Loogisia kieliä voitaisiin toisaalta käyttää semanttisen tiedon poimintaan ja johdetun informaation tuottamiseen XML-dokumenteista. Tästä syystä loogisilla kielillä on tässä luonnollinen ja kiinnostava sovellusalue. [Almendros-Jiménez et al., 2007]

XML-kyselykieli voi perustua logiikkaohjelmointiin, jolloin XML-dokumentit esitetään logiikkaohjelmana. Logiikkaohjelmassa eräs vaihtoehto on esittää dokumentin kaavio (kuten DTD tai XML Schema) sääntöinä ja varsinainen dokumenttisisältö faktoina (Prologissa). Kyselyissä voidaan esimerkiksi XQuery -ilmauksia liittää tähän ohjelmointiparadigmaan perustuvaan kyselykieleen. Tämä tapahtuu kääntämällä XQueryn for-let-where-return -ilmaukset logiikan sääntöjen ja tavoitteiden (engl. goal) avulla logiikkaohjelmaksi [Almendros-Jiménez et al., 2009].

Deklaratiivisen ohjelmoinnin ja XML-tietojen prosessoinnin yhdistäminen on viime vuosina ollut kiinnostusta herättävä tutkimusalue ja uusia ehdotuksia XML-tietojen käsittelykieliksi on ehdotettu perustuen funktionaaliseen ohjelmointiin sekä logiikkaohjelmointiin [Almendros-Jiménez et al., 2009].

Kyselyissä esiintyvät XQuery -ilmaukset ovat käännettävissä vastaaviksi logiikkaohjelmoinnin ilmaisuiksi sääntöjen avulla. Nämä säännöt yhdistetään dokumentin sääntöihin ja faktoihin ja luodaan tavoite, jonka ajamisen tuloksena olevasta vastausten joukosta voidaan rakentaa uusi XML-dokumentti, joka vastaa annetun XQuery -ilmauksen toiminnallisuutta [Almendros-Jiménez et al., 2009]. XPath tai XQuery -kysely toteutetaan määrittelemällä logiikkaohjelma, joka kääntää kyselyn logiikkaohjelmointipohjaiseksi ilmaukseksi, joka kohdistuu logiikkaohjelmana esitettävään lähtödokumenttiin. Kyselyn palauttamista vastauksista voidaan rakentaa uusi XML-tulosdokumentti [Almendros-Jiménez, 2008].

7.1. RDF-dokumenttien kysely

Semanttisessa verkossa voidaan tarkastella kahdenlaisia RDF-kyselykieliä: SQL-tyyppiset (engl. SQL-like) RDF-kielet ja logiikkapohjaiset RDF-kielet. Almendros-Jiménez [Almendros-Jiménez et al., 2008] ehdottaa viittausominaisuudella laajennettua XQuerya RDF-kyselykieleksi. Siinä RDF-dokumentti (eräs XML-dokumentti) esitetään logiikkaohjelmana. Tässä tapauksessa kysely kohdistuu RDF-dokumenttiin (ks. RDF-dokumentin määrittely luvussa 2). RDF-dokumentti on itsekin tietyn tyyppinen XML-dokumentti. Tämä laajennos kyselyiden määrittelyssä sisältää myös vastausten

rakentamisen. Se sisältää myös boolean-tyyppiset predikaatit (ehdot) RDF:n suhdeviitteelle. RDF-metatiedoilla voidaan siis laajentaa verkossa olevaa tietoa, joka on esitetty HTML:llä tai XML:llä, antamalla siitä lisäinformaatiota.

Tämän RDF-kyselykielen, XIndalogin, ideana on tarjota logiikkaohjelmointipohjainen lähestymistapa käsittelemään viittausominaisuuksia semanttisessa verkossa. Tämä laajennos sallii RDF-dokumenttien käsittelyn XQueryn ja XPathin ilmauksilla. Se sallii sisäänrakennettujen predikaattien käyttämisen kyselyissä RDF-suhteiden viittauksessa. Se sallii myös RDF-dokumentteihin ja XML-dokumentteihin perustuvat kyselyt sekä rakentaa vastauksen XML-dokumentiksi.

RDF-laajennoksen syntaksilla käydään läpi RDF-kolmikoita XQueryn for-let-where-return -ilmausten avulla. Tämä periaatteessa yksinkertainen XQuery-kieli on laajennettu nimiavaruusrakenteella, joka sallii muissa lähteissä olevien osoitteiden ilmaisemisen ja RDF-dokumentin kolmikoiden läpikäynnin siihen luodulla for-ilmauksella. [Almendros-Jiménez, 2008]

Edellä esitelty lähestymistapa perusmuodossaan ei tue aggregointioperaatioita, vaan vaatisi niiden suorittamiseksi lisäominaisuuksia. RDF-laajennos XQuery-kieleen RDF-dokumenttien käsittelyä varten yhdistää RDF-dokumentit ja XML-dokumentit mahdollisiksi lähtödokumenteiksi ja tulosdokumenteiksi. Sisäänrakennettujen predikaattien avulla XQueryyn voidaan lisätä päättelymekanismi RDF:nä organisoidulle tiedolle [Almendros-Jiménez, 2008]. Ehdotettu toteutus voidaan toteuttaa automatisoidusti ja kehitteillä onkin ollut virallinen käännös XQuery-kielestä logiikkaohjelmaksi [Almendros-Jiménez, 2008].

7.2. XML-dokumentin kääntäminen logiikkaohjelmaksi

XML-dokumentti esitetään logiikkaohjelmana kääntämällä XML-lähtödokumentti lataamisen yhteydessä logiikkaohjelmaksi. XML-dokumentti myös käännetään logiikkaohjelmaksi. Säännöt, jotka ilmaisevat XML-dokumentin rakenteen ladataan päämuistiin ja faktat, jotka sisältävät dokumentin tietoalkioiden arvot, varastoidaan ulkoiseen muistiin, mikäli ne eivät mahdu päämuistiin. Tämän jälkeen käyttäjä voi kirjoittaa kyselyitä logiikkaohjelmoinnilla esitettyyn dokumenttiin perustuen. Kukin kyselyssä annettu XQuery-ilmaus käännetään vastaavaksi logiikkaohjelmaksi ja siitä muodostetaan oikeaksi todistettava tavoite. Tavoitteen ajon tuottamasta vastausjoukosta muodostetaan tulosdokumentti. [Almendros-Jiménez et al., 2009]

Jotta XML-dokumentin kääntäminen logiikkaohjelmaksi voidaan havainnollistaa mahdollisimman yksinkertaisella tavalla, käytetään seuraavassa (kuva 18) esimerkissä lyhennettyä/muunnettua versiota *kirjat.xml* luvussa 2 annetusta esimerkkidokumentista *kirjasto1.xml*.

```
<kirjat>

   <kirja julkaisuvuosi = ”1936”>
     <nimi>Tuulen viemää</nimi>
     <kirjoittaja>Margaret Mitchell</kirjoittaja>
   </kirja>

   <kirja julkaisuvuosi = ”1991”>
     <nimi>Scarlett</nimi>
     <kirjoittaja>Alexandra Ripley</kirjoittaja>
   </kirja>

</kirjat>
```

18. Lyhennetty XML-esimerkkidokumentti *kirjat.xml*

Koska XPath sisältyy XQueryyn, voidaan XQuery-ilmaukset kääntää kuten XPath-ilmaukset logiikkaohjelmoinniksi. XQuery-ilmaukset käännetään logiikkaohjelmaksi muodostamalla sääntöjä sekä dokumenttien liitokselle (engl. join) että for-let-where -ilmauksille. Logiikkaohjelmakäännöksen lopputulos voi olla esimerkiksi Prolog-ohjelma. Käännöksen tuloksena syntyneet säännöt yhdistetään sen jälkeen XML-lähtödokumenttia esittävien sääntöjen ja faktojen kanssa.

Kuvassa 19 näytetään, millaiseksi logiikkaohjelmaksi kuvan 18 XML-esimerkkidokumentti *kirjat.xml* käännetään. Esimerkkikäännös perustuu Almendros-Jiménezin [Almendros-Jiménez et al., 2009] antamaan esimerkkiin.

Säännöt:

```
kirjat(kirjattyyppi(Kirja, []), NKirjat, 1, Doku) :-
  kirja(Kirja, [NKirja|NKirjat], 2, Doku).

kirja(kirjatyyppi(Nimi, Kirjoittaja, [Julkaisuvuosi]),
  NKirja, 2, Doku) :-
   kirjoittaja(Kirjoittaja, [NKi|NKirja], 3, Doku),
   nimi(Nimi, [NNimi|NKirja], 3, Doku),
   julkaisuvuosi(Julkaisuvuosi, NKirja, 3, Doku).
```

Faktat:

```
julkaisuvuosi('1936', [1,1], 3, "kirjat.xml").

nimi('Tuulen viemää', [1,1,1], 3, "kirjat.xml").

kirjoittaja('Margaret Mitchell',
  [2,1,1], 3, "kirjat.xml").

julkaisuvuosi('1991', [2,1], 3, "kirjat.xml").

nimi('Scarlett', [1,2,1], 3, "kirjat.xml").

kirjoittaja('Alexandra Ripley', [2,2,1], 3, "kirjat.xml").
```

19. Esimerkkidokumentti *kirjat.xml* logiikkaohjelmana

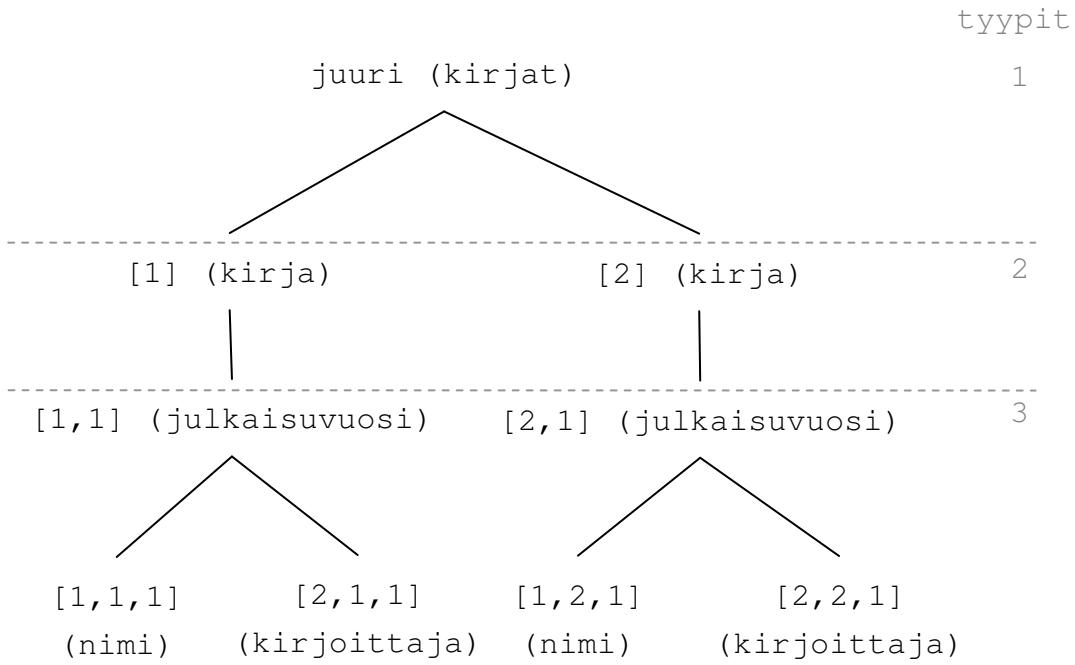
Säännöillä esitetään XML-dokumentin rakenne ja faktoissa esitetään XML-dokumentin varsinainen sisältö. Esimerkiksi kuvassa 19 säännöissä neljäs argumentti on 'Doku', joka ilmaisee dokumentin nimen. Kuvassa annetuissa faktoissa neljäs argumentti on puolestaan "kirjat.xml", joka ilmaisten XML-lähtödokumentin nimen. Tämä havainnollistaa hyvin sen, miten säännöissä ilmaistaan kaaviossa olevan komponentin ja faktoissa olevan sisällön vastaavuus.

Säännöissä XML-dokumentin tunnisteet on käännetty predikaattinimiksi (kirjat, kirja jne.). Kullakin predikaatilla esiintyy neljä argumenttia, joista ensimmäistä käytetään XML-dokumentin rakenteen esittämiseen. Toista argumenttia käytetään kunkin tietoalkion esiintymän numerointiin ja kolmatta tyyppien numerointiin. Viimeistä, eli neljättä, käytetään ilmaisemaan lähtödokumentin nimi. Oleellisinta on kyetä palauttamaan alkuperäinen XML-dokumentti näiden sääntöjen ja faktojen perusteella. [Almendros-Jiménez et al., 2009]

Ensimmäinen sääntöargumentti on itsekin predikaatti, jolla on muuten sama nimi kuin varsinaisella alkuperäisellä tunnisteella, mutta lisäksi loppupääte tyyppi. Toinen argumentti [NKirja|NKirjat] rakentuu siten, että lapsiesiintymää ilmaiseva numero lisätään aina sen vanhemman tietoalkion numeron eteen. Eli lukutapa

solmunumeroinnissa toimii ikään kuin oikealta vasemmalle (ks. kuvan 20 esimerkkidokumentin *kirjat.xml* tietoalkioiden numerointia havainnollistava puu). Kolmas argumentti numeroi tyypit kunkin omalla numerollaan. Tässä esimerkissä tyyppien numerointi tapahtuu seuraavasti: 1: `kirjat` ; 2: `kirja` ; 3: kaikki muut tietoalkiot (`julkaisuvuosi`, `nimi`, `kirjoittaja`).

Esimerkkidokumentin tietoalkioiden numerointia havainnollistava puu:



---

20. Esimerkkidokumentin *kirjat.xml* tietoalkioiden numerointia havainnollistava puu

7.3. Kyselyiden muodostaminen logiikkaohjelmoinnin avulla

Logiikkaohjelmoinnin soveltaminen XML-tiedon kyselyyn on hyvin ilmaisuvoimaista, sillä se on, kuten XQuery, laskennallisesti täydellinen. Loppukäyttäjän näkökulmasta tämä tarkoittaa kuitenkin sitä, että deklaratiivisuuden taso on hyvin alhainen, eli käyttäjän on omattava laajat ohjelmointitaidot hyödyntääkseen logiikkaohjelmointia XML-tiedon kyselyssä. Periaatteessa logiikkaohjelman prosessoinnin tulisi olla loppukäyttäjälle täysin näkymättömissä, mutta jotta sitä voisi kyselemiseen käyttää, on prosessointiin (XQuery-kyselyiden kääntämiseen) perehdyttävä.

Kuvan 18 XML-esimerkkidokumentille *kirjat.xml* voidaan esittää XQuery-kysely, joka käännetään logiikkaohjelmaksi. Kuvassa 21 olevassa XQuery-kyselyssä haetaan niiden kirjojen julkaisuvuodet ja nimet, jotka on julkaistu ennen vuotta 1990. Käännöksessä kartoitetaan kukin XQuery-ilmaus vastaavaksi logiikkaohjelmaksi ja tavoitteeksi. Kuvien 21 ja 22 esimerkki perustuu Almendros-Jiménezin ja kumppaneiden antamiin esimerkkeihin artikkelissa *Integrating XQuery and Logic Programming* [Almendros-Jiménez et al., 2009].

XQuery-kyselyn spesifiointi käyttäjän näkökulmasta:

```
for $kirja in document (”kirjat.xml”)/kirjat/kirja
return let $julkaisuvuosi := $kirja/@julkaisuvuosi
where $julkaisuvuosi < 1990

return

   <vanhatkirjat>
     {$julkaisuvuosi, $kirja/nimi}
   </vanhatkirjat>
```

Aiemmin luonnehdittujen kääntämisperiaatteiden mukaisesti kysely käännetään seuraavaksi logiikkaohjelmaksi.

```
vanhatkirjat(vanhatkirjattyyppi(Nimi,     [Julkaisuvuosi]),
[Solmu], [Tyyppi], [Doku]) :-
   join(Nimi, Julkaisuvuosi, Solmu, Tyyppi, Doku).

join(Nimi, Julkaisuvuosi, [Solmu], [Tyyppi], [Doku]) :-
  vkirja(Nimi,   Julkaisuvuosi,   Solmu,   Tyyppi,   Doku),
  rajoitteet(vkirja(Nimi, Julkaisuvuosi)).

rajoitteet(Vkirja) :- rl(Vkirja).
 rl(Vkirja) :- r(Vkirja).
 r(vkirja(Nimi, Julkaisuvuosi)) :-
 le(Julkaisuvuosi, 1990).

vkirja(Nimi, Julkaisuvuosi, [Solmu, Solmu],
[TNimi, TJulkaisuvuosi], ”kirjat.xml”) :-
   nimi(Nimi,[NNimi|Solmu],Tnimi, ”kirjat.xml”),
   julkaisuvuosi(Julkaisuvuosi, Solmu, TJulkaisuvuosi,
   ”kirjat.xml”).
```

---

21. XQuery -kyselyn esimerkkidokumentista *kirjat.xml* käännös logiikkaohjelmaksi

Kuvassa 21 annetussa logiikkaohjelmassa määritellään ensin XQueryn return-ilmauksen mukaan XML-tulosdokumentin rakenne predikaatilla ’vanhatkirjat’. Tulosdokumentin on siis tarkoitus sisältää nimi-elementtiesiintymä sekä julkaisuvuosi-attribuutti. Logiikkaohjelman toisella predikaatilla ’join’ luodaan tämä rakenne kutsumalla aluksi predikaattia ’vkirja’, joka vastaa XQueryn dokumenttimuuttujaa $kirja. Lisäksi predikaatti join kutsuu myös predikaattia nimeltä rajoitteet tarkistaakseen where-ilmauksessa määritellyn rajoituksen. Predikaatti ’vkirja’ kutsuu XPathin päätunnisteita (engl. head tag), jotka liittyvät dokumenttimuuttujaan $kirja.

Kuvassa 21 annetussa esimerkissä muuttujien $julkaisuvuosi ja $kirja/nimi päätunnisteet ovat $nimi ja $julkaisuvuosi. Erityinen predikaatti rajoitteet tarkistaa ehdon ”$julkaisuvuosi < 1990” koskien muuttujaa $kirja. Koska dokumenttimuuttujia sekä rajoitteita on molempia vain yksi, muunnoksella muodostetaan vain apupredikaatteja ’rl’ (lyhenteellä viitataan rajoitelistaan) ja ’r’ (rajoite) annetun ehdon tarkistamiseen. Predikaatti ’le’ merkitsee tämän esimerkin yhteydessä boolean-operaattoria ’<’ (engl. less, eli pienempi kuin). Tuloksena saatava XML-dokumentti olisi myös mahdollista rakentaa useiden eri XML-dokumenttien perusteella. Esimerkissä on kuitenkin käytetty vain yhtä.

Prosessointia varten generoidaan tavoite:

```
vanhatkirjat(VanhatKirjat, Solmu, Tyyppi, Doku)
```

Prosessointi tuottaa vastauksena seuraavan Prolog-termin:

```
VanhatKirjat(vanhatkirjattyyppi(”Tuulen viemää”,
[”1936”]), [[[[1,1], [1,1]]]], [[[3,3]]],
[[”kirjat.xml”]]
```

Vastaus vastaa siis XML-dokumenttia:

```
<vanhatkirjat julkaisuvuosi=”1936”>
   <nimi>Tuulen viemää</nimi>
</vanhatkirjat>
```

Lisäsäännöt XML-dokumentin kaavion ilmaisuun:

```
vanhatkirjat(vanhatkirjattyyppi(Nimi,[Julkaisuvuosi]),
[[[Tietoalkio1],[Tietoalkio2]]],[[[Tyyppi1,Tyyppi2]]],[[
Doku]]):-
nimi(Nimi,[NNimi|Tietoalkio1],Tyyppi1,Doku),
julkaisuvuosi(Julkaisuvuosi,Tietoalkio2,Tyyppi2,Doku).
```

22. Logiikkaohjelmakäännös: tavoite ja vastaus

Kunkin logiikkaohjelman päätunniste on laskettava jokaisen tavoitteen yhteydessä. Kuvan 21 päätunniste on `vanhatkirjat`. Päätunnisteen sijainti on 1. Täten voidaan muodostaa kuvassa 22 esitetty tavoite: ”`vanhatkirjat(VanhatKirjat, Solmu, Tyyppi, Doku)`”. Kuvassa 22 on annettu myös vastaus tälle tavoitteelle sekä siitä pääteltävissä oleva vastaava XML-tulosdokumentti. Jotta vastausjoukosta voitaisiin rakentaa XML-tulosdokumentti, on otettava huomioon muutama lisäsääntö XML-tulosdokumentin kaavion ilmaisua varten (nämä myös kuvassa 22). Samoin kuin syötedokumenteissa, XML-tulosdokumenteissa lapsisolmut numeroidaan suuremmilla luvuilla kuin niiden vanhemmat. Kuvan esimerkissä elementti `vanhatkirjat` on numeroitu [[[[1,1],[1,1]]]] ja sen lapsielementti nimi on numeroitu [1,1,1].

Almendros-Jiménezin ja kumppaneiden [Almendros-Jiménez et al., 2009] mielestä tämän lähestymistavan eräänä etuna on se, että se voidaan liittää osaksi mitä tahansa Prolog-toteutusta. Lisäksi XQuery-kyselyn lopputuloksen logiikkaohjelmaesitystavan yhteyteen voidaan liittää mitä tahansa logiikkaohjelmointimäärittelyjä ja saada aikaan esimerkiksi sääntöjä, joilla johdettua

informaatiota voidaan tuottaa. Toisin sanoen Prologin ilmaisuvoimaa voidaan käyttää tuottamaan johdettua informaatiota. Logiikkaohjelmia voidaan puolestaan käyttää esimerkiksi RDF-dokumenttien esittämiseen. Koska RDF-dokumentti on eräänlainen säännönmukaisesti organisoitu XML-dokumentti, voidaan täten myös XQuery-kyselyä hyödyntää RDF-dokumenttien käsittelyssä.

Toisaalta tällainen ehdotus hyödyntää myös XML-dokumenttien käsittelyn ja esittämisen tutkimusta relaatiotietokantajärjestelmien yhteydessä. Ehdotus logiikkapohjaisesta kyselykielestä semanttiselle verkolle yhdistää faktojen tehokkaan relaatiotietokantahakutavan hyödyt logiikkaohjelmoinnin päättelykykyjen kanssa. XPath on jo toteutettu logiikkapohjaisena esityksenä XIndalogissa. [Almendros-Jiménez et al., 2009]

7.4. XPath-kyselyt

Almendros-Jimenézin ja kumppaneiden [Almendros-Jiménez et al., 2007] antamien esimerkkien perusteella voidaan tässäkin esimerkiksi tuottaa XPath-kysely '*/kirjat/kirja/kirjoittaja*', joka hakee kirjatietokannasta kirjoittajien tiedot. Tällöin voidaan asettaa tavoite kuten kuvassa 23.

XPath-esimerkkikysely 1:

```
/kirjat/kirja/kirjoittaja
```

Kääntämisen tuloksena syntyy seuraava oikeaksi todistettava tavoite:

```
kirjoittaja(Kirjoittaja, Tietoalkio, 3)
```

Tavoitteen ajaminen tuottaa seuraavat vastaukset esimerkkidokumentin *kirjat.xml* yhteydessä:

```
kirjoittaja('Margaret Mitchell', [2,1,1], 3).
kirjoittaja('Alexandra Ripley', [2,2,1], 3).
```

Niillä on myös analoginen vastaavuus alla olevan XML-dokumentin kanssa, jossa vastauksen juurielementtinä käytetään tässä tunnistetta `<tulos>`:

```
<tulos>
   <kirjoittaja>Margaret Mitchell</kirjoittaja>
   <kirjoittaja>Alexandra Ripley</kirjoittaja>
</tulos>
```

---

23. XPath -esimerkkikysely 1 logiikkaohjelmana dokumentista *kirjat.xml*

Tarkastellaan seuraavaksi toista XPath-esimerkki-ilmausta: ’*/kirjat/kirja[kirjoittaja = ”Alexandra Ripley”]/nimi*’. Tässä tapauksessa mukana ilmauksessa on myös tulosjoukkoa rajaava ehto. Ehto on edellisen ilmauksen osa: ’*kirjoittaja = Alexandra Ripley*’. Nyt on otettava huomioon tavoite sekä yksi erikoissääntö. Esimerkkiä on havainnollistettu kuvassa 24.

XPath-esimerkkikysely 2:

```
/kirjat/kirja [kirjoittaja = ”Alexandra Ripley”]/nimi
```

Erikoissääntö:

```
kirja(kirjatyyppi(Kirjoittaja, Nimi, [Julkaisuvuosi],
 SolmuKirja, 2) :-
 kirjoittaja(Kirjoittaja,[SolmuKirjoittaja|SolmuKirja],3),
 nimi(Nimi,[SolmuNimi|SolmuKirja], 3).
```

Oikeaksi todistettava tavoite:

```
kirja(kirjatyyppi(’Alexandra Ripley’, Nimi,
[Julkaisuvuosi], Solmu, 2)
```

---

24. XPath -esimerkkikysely 2 logiikkaohjelmana dokumentista *kirjat.xml*

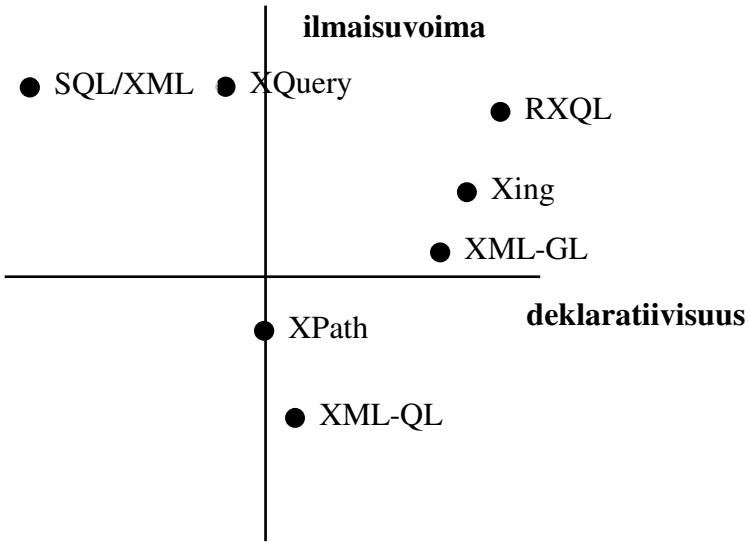
Kuvan 24 esimerkissä on evaluoinnissa tavoitteena käynnistää ensin ’Alexandra Ripley’ -nimisen kirjoittajan kirjojen haku ja noutaa erityisesti solmunumerot näille kirjoille. Mikäli saman kirjoittajan kirjoja olisi dokumentissa useita kappaleita, noudettaisiin ne kaikki solmunumeroineen. Esimerkkidokumentissa on kuitenkin vain yksi kirja, jonka hän on kirjoittanut. Argumentin alustus tavoitteessa tekee mahdolliseksi noutaa solmunumeroita, jonka jälkeen käyttäjän on myös mahdollista hakea kyseessä olevan kirjoittajan kirjojen nimet. Predikaatin ’`kirjoittaja(Kirjoittaja,[SolmuKirjoittaja|SolmuKirja],3)`’ käyttäminen on oleellista näiden nimien haulle, olettaen että tietoalkion numero on alustettu tässä predikaatissa. Tässä esimerkissä ensimmäiseksi käytetty fakta on ’`kirjoittaja(’Alexandra Ripley’, [2,2,1], 3)`’ solmunumerolla [2,2,1] ja tätä numeroa käytetään hakemaan fakta ’`nimi(’Scarlett’,[1,2,1],3)`’.

8. XML-kyselykielten vertailua

XPathiin perustuvissa kielissä, kuten XQuery, on käyttäjällä kyselyiden ja dokumenttien käsittelemiseksi oltava tietämystä XML-dokumentin rakenteesta. Joidenkin

kyselykielten kehityksessä on otettu askeleita sen puolesta, että käyttäjän ei tarvitse hallita varsinaisia ohjelmointitekniikoita ja XML-merkintätapaa. Kyselykielille toivottavia piirteitä ovat hyvä ilmaisuvoima ja helppokäyttöisyys. Kielen deklaratiivisuus on keino saavuttaa nämä ominaisuudet.

Kuvassa 25 on havainnollistettu nelikentän avulla tässä Pro gradussa tarkasteltujen kielten deklaratiivisuuden ja ilmaisuvoiman asteita. Mitä korkeammalle tietty kyselykieli kuvassa sijoittuu, sitä suurempi on sen ilmaisuvoima. Mitä enemmän oikealla kieli puolestaan kuviossa on, sitä korkeampi sen deklaratiivisuuden aste on.



25. Nelikenttä kyselykielten deklaratiivisuuden ja ilmaisuvoiman vertailusta

Kyselykieli on deklaratiivinen, kun sillä määritellään mieluummin, mitä kyselyn tuloksen pitää sisältää, kuin se, miten tulos saadaan aikaiseksi. Tämän määritelmän mukaan esimerkiksi kyselykielet XML-QL ja XML-GL ovat deklaratiivisia kieliä. XML-QL:llä on muuttujien yhdistäminen yhtenä osanaan ja se muistuttaa siten loogista kieltä, vaikkakin omanlaisellaan XML-tyylillä. Muuttujien yhdistämisellä tarkoitetaan kahden eri nimisen muuttujan yhdistämistä siten, että niitä voidaan vertailla keskenään (esimerkkinä muuttujan käyttö tunnisteessa, jolloin verrataan tietoalkioiden sisältöjä). Kyselyn muodostaminen XML-GL:llä puolestaan muistuttaa kyselyn muodostamista QBE:llä (engl. Query By Example). [Bonifati and Ceri, 2000]

Deklaratiivisuuden asteen voidaan myös sanoa olevan sitä korkeampi, mitä helpompi loppukäyttäjän on kieltä käyttää. Tämän määritelmän mukaan esimerkiksi XML-GL on visuaalisuutensa johdosta huomattavasti deklaratiivisempi kuin tekstimuotoinen XML-QL. Ennen kaikkea tämä johtuu siitä, että XML-QL:ssä käyttäjän täytyy ymmärtää mallinsovitukseen perustuva muuttujakäsite. Edelleen tämän

määritelmän mukaan, RXQL-kyselykieli häviää XML-GL:lle kyselyissä, jotka edellyttävät vain tietojen poimintaa ja valintaa. RXQL:llä voidaan kuitenkin tehdä deklaratiivisesti kyselyjä, joita XML-QL:lla taas ei voida tehdä. XPath, XQuery ja SQL/XML eivät ole kovin deklaratiivisia, sillä käyttäjältä vaaditaan ohjelmointi- ja mallinsovitustaitoja (ks. luku 9).

Ilmaisuvoima kertoo millaisia kyselyjä kyselykielellä voidaan tehdä. Tästä näkökulmasta esimerkiksi XML-QL on melko ilmaisuvoimainen, johtuen sen voimallisuudesta kyselyiden ilmaisussa. XML-GL taas on hieman vähemmän voimallinen kuin XML-QL, johtuen pääosin rajoituksistaan sisäkkäisten kyselyiden tukemisessa, tiettyjen määriteltyjen polkuilmausten tukemisessa sekä Skolem-funktioiden tukemisessa [Bonifati and Ceri, 2000]. XML-GL -kieli kuitenkin tukee kaikkea sitä mikä on luonnollisesti määritelty graafisella formalismilla [Bonifati and Ceri, 2000]. Sekä XML-QL että XML-GL on molemmat tehty tukemaan tietojen poimintaa, valintaa ja tuloksen muotoilua, joten ilmaisuvoimassa ei ole suuriakaan eroja.

Ilmaisuvoimaa voidaan arvioida myös sen perusteella, miten hyvin tietty kyselykieli tukee tiettyjä piirteitä, kuten esimerkiksi tiedon aggregointia, ryhmittelyä tai harmonisointia. Oleellista on myös tietää onko kieli laskennallisesti täydellinen. Tarkastelluista kielistä ainoastaan XQuery on laskennallisesti täydellinen. Myös luvussa 7 käsitelty polkuorientoituneiden XML-kielten ilmausten konvertointi logiikkaohjelmaksi ja mahdollisuus esittää siihen logiikkapohjaisia sääntöjä, tarjoaa mahdollisuuden laskennallisesti täydelliseen ilmaisuvoimaan. Suurin osa käsitellyistä kielistä tukee myös kyselytulosten aggregointia ja järjestämistä. Tarkastelluista kielistä ainoa, joka selkeästi tukee tiedon harmonisointia kyselyn yhteydessä on RXQL.

Kyselykielen helppokäyttöisyys riippuu kielen deklaratiivisuuden asteesta. Se on siis ominaisuus, joka ilmaisee miten helppoa XML-kyselyn ohjelmoijalle on kirjoittaa ja/tai ymmärtää kysely [Bonifati and Ceri, 2000]. Helppokäyttöisyys on sidoksissa myös erilaisten käyttäjäryhmien taustatietoihin. Esimerkiksi useimmat SQL-tyyppiset XML-kyselykielet lienevät melko suoraviivaisia ja helppoja oppia sellaisille XML:n tottuneille käyttäjille, joilla on jotain kokemusta SQL:n käyttämisestä relaatiotietokantojen yhteydessä. Tottuneet käyttäjät ovat sellaisia, jotka käyttävät tietokantoja, XML:ää ja niihin liittyviä sovelluksia usein esimerkiksi joko työssä tai opinnoissa. Helppokäyttöisyys on eri asia sellaisille loppukäyttäjille, joilla ei ole juuri aiempaa tietämystä XML-syntaksista, tietokannoista tai ohjelmointi- ja kyselykielistä. Helppokäyttöisyys tällaisten käyttäjien osalta edellyttää kyselykieliltä korkeaa deklaratiivisuuden tasoa.

Edellä kerrotun helppokäyttöisyyden määritelmän mukaisesti esimerkiksi XML-GL on helppokäyttöinen, koska se on graafisen käyttöliittymän tähden intuitiivinen käyttäjälle, eli sitä on helppo lukea ja ymmärtää. Sama pätee useimmille muille

visuaalisille kyselykielille, kuten Xing:lle. XML-QL (ja muut SQL-tyyppiset kyselykielet) puolestaan lienee helpompaa sellaisille tahoille, joilla on jo jotain tietämystä XML:stä [Bonifati and Ceri, 2000]. Mallinsovitus on kuitenkin kaikkien polkuorientoituneiden kielien yhteydessä hankalaa tavalliselle loppukäyttäjälle, vaikka tuntisikin XML-esitystavan. Seuraavassa kahdessa alaluvussa 8.1. ja 8.2. tarkastellaan vielä hieman tarkemmin XML-QL ja XML-GL -kyselykielten keskenään vertailtavissa olevia piirteitä sekä SQL/XML ja XQuery -kyselykielten vertailuominaisuuksia.

8.1. XML-QL ja XML-GL

XML-QL- ja XML-GL-kyselykielet voivat käsitellä annetun XML-dokumentin elementtejä joko järjestettyinä tai järjestämättöminä. XML-QL:llä kuitenkin pystytään tämän lisäksi tekemään kyselyitä kaavioon (schema) sisältyvään järjestykseen perustuen.

Tekstuaalisen XML-QL -kyselyn tulos on XML-lähtödokumentin osa. Tulos määritellään ’CONSTRUCT’ -lauseessa. CONSTRUCT-osan tunnisteiden nimet ja sisältö on täysin määriteltyä siten, että sisältö rakentuu tyypillisesti niistä tietoalkioiden sidoksista, jotka ehtojen evaluoinnissa päätellään. [Bonifati and Ceri, 2000]

Graafinen XML-GL -kysely puolestaan koostuu kahdesta joukosta suunnattuja, syklittömiä kaavioita. Nämä kaaviot esitetään rinnakkain ja erotetaan toisistaan pystysuoralla viivalla. Viivan vasemmalla puolella ilmaistaan kyselyn lähteet, eli valitut dokumentit, sekä predikaatit, eli täytettävät ehdot. Kyselyn oikealla puolella ilmaistaan tulosdokumentin haluttu rakenne. Vaikka yleinen esitystapa XML-QL:n ja XML-GL:n välillä eroaa merkittävästi toisistaan, kyselyssä määriteltävä dokumentin valintaoperaatio on käytettävissä molemmissa. [Bonifati and Ceri, 2000]

Kun tehdään kyselyitä puolirakenteiseen tietoon perustuen, etenkin kun tarkkaa rakennetta ei tunneta, on käytännöllistä soveltaa polkuilmauksiin sisältyvää navigointia. Tiiviimmän polkuilmauksen muodon ei tarvitse sisältää kaikkia polun elementtejä, sillä polun määrittelyssä voidaan käyttää jokerimerkkejä viittaamaan polun tunnistamattomaan osaan. Kaikki polkuorientoituneet XML-kielet tukevat tällaisia osittain määriteltyjä polkuilmauksia. [Bonifati and Ceri, 2000]

XML-QL tukee osittaisia polkuilmauksia, mutta XML-GL vain tietyssä määrin. XML-QL:ssä polkuilmaukset annetaan tunnistemääritteiden sisällä ja ne sallivat tiedon muuntamisen, yhdistelyt ja Kleenen tähtioperaattorin (tietyn joukon 0-n kertainen toisto). XML-GL:ssä ainoa tuettu osittainen polkuilmaus on satunnainen sisältyvyys käyttämällä jokerimerkkiä ’*’ tietoalkioiden välistä suhdetta ilmaisevan graafisen kaaren nimenä. Tämän avulla on mahdollista kulkea pitkin XML-GL -kaaviota ja saavuttaa millä tahansa hierarkian syvyysasteella oleva tietoalkio. [Bonifati and Ceri, 2000]

Jokerimerkkejä sisältävät ilmaukset saattavat olla päättymättömän prosessoinnin lähteenä, kun lähtödokumentti sisältää sisäkkäin saman nimisiä tietoalkioita. Täten

yleinen käytäntö on määritellä tiettyjä pysäytysehtoja yhteensovittamisen algoritmiin, joka sitoo tietoalkioiden ilmentymät polkuilmauksiin, kun sama sidos liittyy samaan kyselyn kohteena olevaan tietoalkioon useammin kuin kerran. XML-QL:ssä ei ole määritelty tällaista pysäytysehtoa, toisin kuin XML-GL:ssä. [Bonifati and Ceri, 2000]

Ehtonegaatio ilmentymäjoukolle tyydytetään, mikäli yksikään ilmentymistä ei tyydytä ehtoa. XML-GL -kielessä negaatio ilmaistaan graafisesti katkoviivalla piirretyllä kaarella suorakulmioiden välillä. XML-QL taas ei tue negaation esittämistä. Se tukee kylläkin eriarvoisuusvertailua yksinkertaisissa ehtoilmauksissa. [Bonifati and Ceri, 2000]

Tuloselementtejä voidaan aggregoida, ryhmitellä tai järjestää uudelleen erityisten funktioiden avulla. Tällainen funktio voi olla esimerkiksi SQL:stä tuttu ”group by” -ryhmittelyfunktio. XML-GL -kielen avulla poimitut tietoalkiot voidaan ryhmitellä kyseisten tietoalkioiden tiettyjen arvojen tai sisältöjen mukaan. Ryhmittelylause ”group by” sellaisenaan puuttuu kokonaan XML-QL:n nykyisestä syntaksista. [Bonifati and Ceri, 2000]

Skolem-funktio liittää annettuun arvoon luodun yksilöllisen (uniikin) OID:n (engl. Object Identifier, eli objektitunniste). Skolem-funktio sovellettuna samaan arvoon tuottaa saman OID:n. Näitä funktioita tarvitaan olemassa olevien dokumenttien yhdistämiseen yhdeksi dokumentiksi. [Bonifati and Ceri, 2000]

XML-QL -kielessä ja XML-GL -kielessä Skolem-funktio ottaa syötteenä listan muuttujia argumenttina ja palauttaa argumentille yksilöllisen elementin kutakin elementtisidosta ja/tai attribuuttisidosta kohden. XML-QL:ssä Skolem-funktio liittää elementti-ilmentymiin OID:t, johon voidaan sitten viitata muuttujia käyttämällä. XML-GL sen sijaan ei anna mahdollisuutta viitata täsmällisesti objektitunnisteisiin. [Bonifati and Ceri, 2000]

Aggregointifunktiot laskevat skalaarisia tunnuslukuja liittyen joukossa oleviin arvoihin. Klassisia SQL:n tukemia aggregointifunktioita ovat *min*, *max*, *sum*, *count* ja *avg*, jotka antavat minimin, maksimin, summan, lukumäärän ja keskiarvon annetussa joukossa. XML-QL -kyselykieli ei selkeästi tue aggregointifunktioita. XML-GL:ssä aggregointifunktiot esitetään visuaalisesti tietyillä graafisilla symboleilla. [Bonifati and Ceri, 2000]

Kuten SQL:ssä, kysely voi sisältää sisäkkäisiä alakyselyjä. XML-QL -kielessä kyselyjä voidaan muodostaa sisäkkäin millä tahansa sisäkkäisyyden tasolla. XML-GL ei tue tämän kaltaista kyselyjen sisäkkäisyyttä. Kysely voi olla binäärinen (kaksijakoinen), eli koostua kahden alikyselyn unionista, leikkauksesta tai erotuksesta. XML-QL tukee unionia ja leikkausta mutta ei erotusta. XML-GL:ssä kyselyt sallivat useita kaavioita kyselyn pystyviivan vasemmalle puolelle. Tämä merkitsee unionin ilmaisemista. Leikkaus puolestaan voidaan esittää soveltamalla toistuvasti negaatiota. [Bonifati and Ceri, 2000]

Tunnistemuuttujien olemassaolo tekee mahdolliseksi esittää täsmällisiä kyselyitä, jotka käsittelevät tunnisteiden nimiä niiden sisällön sijasta. XML-QL -kielessä tämä on mahdollista, sillä muuttujia voidaan liittää tunnisteisiin, joita voidaan käyttää tuloksen tunnisteina. XML-GL ei tue tunnisteisiin kohdistuvia kyselyjä. [Bonifati and Ceri, 2000]

Päivitysoperaatiot, eli elementtien lisäämiset, poistot ja niiden päivitykset, ilmaistaan XML-GL:ssä nuolien avulla. Nämä nuolet on nimetty seuraavasti: I (engl. insert), D (engl. delete) ja U (engl. update), eli sijoitus, poisto ja päivitys. Kullakin primitiivillä, paitsi poistolla, on vasemman puoleinen elementti ja oikeanpuoleinen kaavio. Vasen puoli on operaation kohde-elementti ja oikeanpuoleinen kaavio esittää arvoja, jotka lisätään tai korvataan primitiivillä. XML-QL:ssä ei ole päivityskieltä. [Bonifati and Ceri, 2000]

8.2. SQL/XML ja XQuery

Kun tarkastellaan kielen käytön tehokkuutta ja siitä saatavaa hyötyä, voidaan puhua suorituskyvystä (engl. performance). Kieli on sitä proseduraalisempi, mitä tarkempi varattujen sanojen ja muuttujasidosten järjestys sillä on verrattuna vähemmän proseduraaliseen kieleen. Mitä proseduraalisempi käytettävä kyselykieli on, sitä parempi voidaan suorituskyvyn sanoa olevan loppukäyttäjälle. [Graaumans, 2004]

SQL/XML ja XQuery -kyselykielillä on molemmilla tarkempi varattujen sanojen järjestys kuin esimerkiksi XSLT-kielellä [W3C, 2007b]. SQL/XML käyttää standardeja lauseita: SELECT, FROM, WHERE, jotka liittyvät relationaalisesti esitettyyn tietoon. Laajennosfunktiota käytetään XML-tyyppiarvojen käännökseen SQL:n tyyppiarvoiksi. XQuery:llä käsitellään relaation attribuutteja, joiden arvot ovat XML-dokumentteja. XQueryn määrittely perustuu FOR-, LET-, WHERE- ja RETURN-lauseisiin. Lisäksi XQuery:llä yhdistetään ko. lauseet toisiinsa käyttämällä muuttujia, jotka alustetaan XML-lähdedokumentista saatavilla arvoilla. Näitä muuttujasidoksia voidaan sitten käyttää muissa lauseissa suodatukseen ja uusien tulosten luomiseen. Luultavasti käyttäjät kykenevät käyttämään paremmin XQueryä kuin SQL/XML:ää, koska käyttäjän tarvitsee synkronoida kahden tyyppistä koodia, SQL:ää ja XQueryä toistensa kanssa SQL/XML:ssä. [Graaumans, 2004]

Sellainen kyselykieli, joka tarvitsee vähemmän ilmauksia tietyn kyselyn muodostukseen, on käyttäjän näkökulmasta tehokkaampi, kuin sellainen kyselykieli, joka tarvitsee enemmän ilmauksia. SQL/XML esimerkiksi tarvitsee enemmän ilmauksia saman kyselyn muodostamiseen kuin vastaavan kyselyn tekemiseen XQuery:llä. Tämä johtuu SQL/XML-kielen laajennosfunktioista, joita tarvitaan SQL/XML-kyselyn jokaiseen lauseeseen. Jos operaattoreilla, varatuilla sanoilla ja funktioilla on johdonmukaiset ja yksinkertaiset nimet, on kyselykielellä parempi suorituskyky kuin sellaisella kyselykielellä, jolla on epäintuitiiviset nimet ja semantiikat. SQL/XML on

vähemmän intuitiivinen operaattoreiden ja funktioiden nimeämiskäytännöissä ja semantiikoissa kuin XQuery. [Graaumans, 2004]

XQueryn ilmaisutehokkuus on parempi kuin SQL/XML:llä. Jos arvioidaan tiettyjen kyselyiden oikein muodostamista XQueryn ja SQL/XML:n välillä, on käyttäjien väärin muodostamien kyselyjen prosenttiosuus suurempi SQL/XML:llä. SQL/XML-kielellä on myös suurempi määrä eri vaiheita kyselyn ratkaisuprosessissa. [Graaumans, 2004]

9. Tutkimuksen tulokset

Yleisesti ei oleteta, että loppukäyttäjä kykenisi käyttämään XML-kyselykieliä ilman tietämystä dokumenttien rakenteista tai mallinsovitusmekanismeista. Yleisimpien kielten käyttäminen edellyttää myös ohjelmointitaitoja ja tietämystä tietokannoista. Arviot tässä Pro gradu-tutkielmassa tarkastelluista XML-kyselykielistä on koottu kuvan 26 taulukkoon. Taulukossa on eri XML-kyselykieliä vertailtu tiettyjen ominaisuuksien perusteella, jotka loppukäyttäjän näkökulmasta ovat keskeisiä. Taulukossa on lihavoituna kyseiseen kieleen liittyvä positiivinen arvio (kyllä tai ei -vaihtoehdoista) ja kursivoituna negatiivinen arvio. Taulukon arvio ”jossain määrin” merkitsee, että arvio on vain osittain positiivinen.

Taulukossa vertailtavat piirteet voidaan luokitella kahteen osaan. *Kielen käyttämisen edellytyksiin* liittyvät seuraavat taulukon piirteet: käyttäjän tietämys dokumentin rakenteesta, kielen deklaratiivisuuden korkea aste, mallinsovituksen käsitteen ymmärtäminen ja muuttujakäsitteen ymmärtämisen välttämättömyys sekä XML-merkintätavan hallitseminen. Ilmaisuvoimaan liittyviä tarkastelupiirteitä ovat taulukon seuraavat piirteet: tuki tiedon aggregoinnille, tuki tiedon koonnille/ryhmittelylle, tuki tulosten järjestämiselle, tuki tiedon harmonisoinnille sekä sisältääkö kieli laskennallisesti täydellisen ilmaisuvoiman.

9.1. Kielen käyttämisen edellytyksiin liittyvät piirteet

Käyttäjältä vaadittava tietämys dokumentin rakenteesta/sisällöstä

Suurin osa XML-kyselykielistä vaatii käyttäjältä jonkinasteista tietämystä kyselyn kohteena olevan XML-dokumentin rakenteesta. Joissain kielissä on kyselyissä mahdollista käyttää esimerkiksi jokerimerkkejä, kuten ’*’, ilmaisemaan sitä osaa kyselyssä olevassa polkuilmauksessa, jota käyttäjä ei tunne tai joka on mielenkiinnoton kyselyn näkökulmasta. Tässä tapauksessa riittää käyttäjän osittainen dokumenttirakenteen tunteminen.

Esimerkiksi RXQL-kielen tapauksessa dokumentin rakenteen ei tarvitse olla käyttäjän tiedossa lainkaan. Käyttäjän on vain tiedettävä tarvittavien elementtien ja attribuuttien nimet, ts. lähtödokumentin tietosisältö, voidakseen muodostaa uusia

dokumenttirakenteita. Muissa tutkielmassa tarkastelluissa kielissä on dokumenttirakenne tunnettava vähintään osittain. XQuery:ssä käyttäjän on myös hallittava iteratiiviset kontrollirakenteet.

Kielellä on korkea deklaratiivisuuden aste

Korkeampi deklaratiivisuuden aste kyselykielellä merkitsee parempaa soveltuvuutta loppukäyttäjälle. Kyselyn määrittelyssä on siis voitava keskittyä tuloksen sisällön kuvaamiseen, tuloksen tuottamisen sijasta. Kyselykieli on sitä deklaratiivisempi, mitä vähemmän käyttäjän tarvitsee miettiä miten sitä tulisi käyttää. Ideaalisessa tapauksessa käyttäjällä ei tarvitsisi olla mitään tietämystä XML-dokumenttien rakenteesta, mallinsovitustekniikoista, muuttujakäsitteistä tai ohjelmointiparadigmoista.

RXQL-kyselykieli tukee korkeasti deklaratiivista kyselyiden muodostamista varsinkin niiden piirteiden osalta, joita muut XML-kyselykielet eivät tue. Loppukäyttäjältä ei esimerkiksi vaadita esimerkiksi proseduraalisen ohjelmoinnin osaamista. Kuitenkin vaaditaan, että käyttäjä havaitsee tietojen konfliktitilanteet ja kykenee poistamaan ne harmonisoidessaan tietoja kyselyjen yhteydessä. Kuvasta 26 voidaan nähdä, että viisi ensimmäistä piirrettä, jotka liittyvät kielen käyttämisen edellytyksiin, ovat kaikki positiivisia piirteitä ainoastaan RXQL-kielessä. Tästä voidaan päätellä se, että RXQL on tarkastelluista kielistä korkeimmalla deklaratiivisuuden asteella muihin tarkasteltuihin XML-kyselykieliin nähden.

XML-GL ja Xing ovat visuaalisuutensa vuoksi myös melko korkealla deklaratiivisuuden asteella muihin tarkasteltuihin kieliin (lukuun ottamatta RXQL:ää) nähden. Kuitenkin XML-kyselykielten tulisi olla niin korkealla abstraktiotasolla, että tiedon uudelleenstrukturointi tulosta muodostettaessa on automaattista eikä näy käyttäjälle. Tämä ei toteudu tarkastelluissa visuaalisissa kyselykielissä. Siksi Xing ja XML-GL eivät sisällä aivan samaa deklaratiivisuuden astetta kuin RXQL.

XML-QL on myös muuten melko deklaratiivinen, mutta sen käyttö kuitenkin vaatii jonkin verran taustatietämystä, kuten aiemmin on esitetty (ks. kuva 26). XPath:n, XQuery:n ja SQL/XML:n deklaratiivisuuden aste on alhainen, sillä ne vaativat hyvin paljon taustatietämystä ja -taitoja. Esimerkiksi XPathin ja XQueryn yhteydessä, käyttäjän on tunnettava XML-dokumentin rakenne yksityiskohtaisesti voidakseen tehdä kyselyitä. Käyttäjän on myös ymmärrettävä polkuorientoitunut mallinsovituksen käsite sekä proseduraalinen muuttujakäsite. XPathissa ei suoraan vaadita XML-merkintätavan hallintaa käyttäjältä, mutta koska XPathia harvoin sellaisenaan käytetään (vaan osana jotain laajempaa kokonaisuutta tai kieltä, kuten XQuery) on XML-merkintätavan ymmärtäminen tarpeen. XPath ja XQuery eivät myöskään tue tiedon harmonisointia, vaikka XQuery:llä laskennallisesti täydellisenä kielenä se voidaan ohjelmoida.

Käyttäjän on ymmärrettävä mallinsovituksen käsite

Yleisimmissä XML-kyselykielissä vaaditaan käyttäjiltä jonkinasteista mallinsovituksen käsitteen ymmärtämistä ja hallintaa. Esimerkiksi visuaalisessa kyselykielessä Xing, kyselyiden muodostaminen perustuu dokumenttimalleihin, mallinsovitukseen ja sääntöihin. Xing tukee mallinsovitustekniikoiden osoita-ja-klikkaa -rajapintaa. RXQL on tarkastelluista kielistä ainoa, joka ei vaadi mallinsovituksen käsitteen ymmärtämistä käyttäjältä kyselyiden muodostamisessa.

Käyttäjän on hallittava muuttujakäsite

Muuttujakäsitteen hallintaa ei vaadita käyttäjältä tarkastelluissa visuaalisissa kyselykielissä XML-GL:ssä ja Xing:issä, eikä myöskään RXQL:ssä, koska niissä ei käytetä varsinaisia muuttujia kyselyiden muodostamisen yhteydessä. Muuttujia ei tarvita XML-GL:ssä ja Xing:issä kyselyn visuaalisuuden takia, koska tarvittavat rakenteet on heti nähtävillä ja valittavissa osaksi kyselyä. Muissa tarkastelluissa kielissä on muuttujan käsite hallittava. Esimerkiksi XML-QL:ssä ja XQuery:ssä on käyttäjän välttämätöntä tuntea proseduraaliseen muuttujakäsitteeseen liittyvä alustaminen.

Käyttäjän on hallittava XML-merkintätapa

Käyttäjän on hallittava XML-merkintätapa XQuery:ssä, SQL/XML:ssä sekä XML-QL:ssä. Visuaalisissa kyselykielissä (Xing ja XML-GL) tai RXQL-kielessä ei käyttäjän tarvitse tuntea XML-merkintätapaa. Vaikka RXQL on tekstuaalinen heterogeenisten XML-dokumenttien käsittelyyn tarkoitettu kieli, se ei pohjaudu miltään osin XML-syntaksiin. Siksi käyttäjän ei myöskään tarvitse osata XML:ää.

Xing ja visuaaliset kielet yleensäkin, on tarkoitettu erityisesti suppean taustatietämyksen omaaville loppukäyttäjille ja siksi ne pyrkivätkin välttämään varsinaista XML-syntaksia. Xing:issä visuaalinen dokumenttimalli tekee mahdolliseksi muodostaa kyselyitä lomakepohjaisen käyttöliittymärajapinnan avulla, jossa ilmaistaan ainoastaan kiinnostuksen kohteena olevia tietoalkioita riippumatta niiden tekstuaalisesta esittämistavasta.

Tekstuaalisissa XML-kyselykielissä, kuten XML-QL:ssä, kyselyistä tulee helposti melko pitkiä ja jotta sitä voitaisiin käyttää tehokkaasti on ymmärrettävä hyvin XML-syntaksia. Jo muuttujien alustaminen edellyttää, että käyttäjä tuntee yksityiskohtaisesti lähtödokumentin XML-esityksen. Sama pätee koskee SQL/XML-kieltä.

kieli / piirre	XPath	XQuery	RXQL	SQL/XML	XML-QL	XML-GL	Xing
käyttäjän tietämystä dokumentin rakenteesta	*Kyllä*	*Kyllä*	**Ei**	*Kyllä*	Jossain määrin	Jossain määrin	Jossain määrin
kielen deklaratiivisuuden aste korkea	*Ei*	*Ei*	**Kyllä**	*Ei*	*Ei*	**Kyllä**	**Kyllä**
käyttäjän ymmärrettävä mallinsovituksen käsite	*Kyllä*	*Kyllä*	**Ei**	*Kyllä*	*Kyllä*	*Kyllä*	*Kyllä*
käyttäjän hallittava muuttujakäsite	*Kyllä*	*Kyllä*	**Ei**	*Kyllä*	*Kyllä*	**Ei**	**Ei**
käyttäjän hallittava XML-merkintätapa	**Ei**	*Kyllä*	**Ei**	*Kyllä*	*Kyllä*	**Ei**	**Ei**
kieli tukee tiedon aggregointia	Jossain määrin	Jossain määrin	**Kyllä**	**Kyllä**	*Ei*	**Kyllä**	**Kyllä**
kieli tukee tiedon koontia/ ryhmittelyä	Jossain määrin	Jossain määrin	**Kyllä**	**Kyllä**	*Jossain määrin*	**Kyllä**	**Kyllä**
kieli tukee tulosten järjestämistä	Jossain määrin	Jossain määrin	**Kyllä**	**Kyllä**	**Kyllä**	**Kyllä**	**Kyllä**
kieli tukee tiedon harmonisointia	*Ei*	*Ei*	**Kyllä**	*Ei*	*Ei*	*Ei*	*Ei*
kielellä on laskennallisesti täydellinen ilmaisuvoima	*Ei*	**Kyllä**	*Ei*	Jossain määrin	*Ei*	*Ei*	*Ei*

26. Tarkasteltujen kyselykielten ominaisuuksien vertailu loppukäyttäjän näkökulmasta

9.2. Ilmaisuvoimakysymykset

Kieli tukee tiedon aggregointia

XML-QL ei tiettävästi tue aggregointifunktioita nykyisessä muodossaan, mutta muut tarkastellut kyselykielet tukevat. Visuaalisissa kyselykielissä, XML-GL ja Xing, aggregointi esitetään graafisesti. XPath ja XQuery eivät alkuperäisessä muodossaan tue tiedon aggregointia, mutta XQuery:llä voidaan tämäkin toiminnallisuus ohjelmoida.

Kieli tukee tiedon koontia/ryhmittelyä

Tiedon koonnilla tarkoitetaan tiedon ryhmittelyä perustuen tiettyjen tietoalkioiden arvioihin. Kyselyn tuloselementit voidaan siis järjestää uudelleen saatavilla olevaa ryhmittelyfunktiota soveltamalla. Yleisin on ryhmittelyfunktio ”*group by*” -ilmaisu, jolla kyselyn tulosjoukko ryhmitellään annetun ehdon perusteella. Tarkastelluissa XML-kyselykielissä, lukuun ottamatta XML-QL -kyselykieltä, oli ainakin jonkinlainen tuki tulostiedon koonnille ja ryhmittelylle. XPathin ja XQueryn yhteydessä tiedon koonnin tuki saadaan tosin vain lisätyillä funktioilla, joita ei alkuperäisissä kielissä ole.

Ilmeisesti XML-QL:n syntaksista tämän kaltainen ryhmittelylause puuttuu. Tulosta voidaan kuitenkin tarkentaa joidenkin elementtiarvojen perusteella. Esimerkiksi XML-QL:ssä voidaan määrittää tietty elementtiesiintymän tai attribuutti-ilmentymän arvo, jonka perusteella muita tähän kyseiseen elementtiesiintymään liittyviä elementti- tai attribuuttiarvoja voidaan hakea.

Esimerkiksi RXQL:ssä ryhmittelyilmaus ”*group by*” ilmaisee ryhmittelyssä käytettävät tietoalkiot eksplisiittisesti. Myös XML-GL:ssä poimitut tietoalkiot voidaan ryhmitellä niiden arvojen ja tietosisältöjen mukaan. Xing:issä tiedon muuntaminen ja uudelleenrakenteistaminen puolestaan tapahtuu sääntöjä käyttämällä.

Kieli tukee tulosten järjestämistä

Kaikki tarkastellut XML-kyselykielet tukevat tiedon järjestämistä ainakin osittain. Esimerkiksi RXQL-kieli käyttää ”*order by*” -funktiota määrittelemään, mitä tietoalkioita kyselyn tuloksessa tulee käyttää perusteena elementtien ilmentymien esiintymisjärjestyksen määräämisessä. XPath ja XQuery tukevat kyselyn tulosten järjestämistä puolestaan vain jos toiminnallisuus ohjelmoidaan XQuery:llä.

Kieli tukee tiedon harmonisointia

Tiedon harmonisoinnilla pyritään eliminoimaan erilaisuudet (heterogeenisyys) lähtödokumenttien organisoinnissa ja esittämisessä poimittaessa tietoja eri tietolähteistä. Samaa tarkoittavat tiedot pyritään siis esittämään yhtenäisellä tavalla. Tiedon harmonisointia tarvitaan, kun käsitellään tuntemattomia tietolähteitä, joiden sisältö ja rakenteet eivät ole käyttäjän tiedossa. Ylipäätänsä tietokeskeiset sovellutukset edellyttävät tietojen harmonisointia.

Vaikka RXQL-kielellä kyselyiden muodostuksessa ei tarvitakaan tietämystä XML-syntaksista, käyttäjän pitää tunnistaa XML-pohjaisissa lähtödokumenteissa olevat tiedon konfliktit. RXQL on kuitenkin tiettävästi ainoa tässä tarkastelluista XML-kyselykielistä, joka tukee tiedon harmonisointia. RXQL:n tietolähteiden harmonisointikyky mahdollistaa myös ryhmittelyn harmonisoidulle tiedolle.

Kielellä on laskennallisesti täydellinen ilmaisuvoima

Ainoastaan XQuery:llä on todistetusti laskennallisesti täydellinen ilmaisuvoima. Muilla tässä Pro gradussa tarkastelluilla XML-kielillä sitä ei ole. XQuery, nimestään huolimatta, ei olekaan puhtaasti XML-kyselykieli, vaan ennemminkin XML-pohjainen ohjelmointikieli. Laskennallisesti täydellisellä kielellä voidaan esittää mikä tahansa ohjelmoitavissa oleva tehtävä. Tällainen kieli on tarkoitettu pääosin ohjelmisto- ja tietokantatoteuttajille. Tällainen kieli on siis tarkoitettu sellaisille ohjelmointiammattilaisille, joilla on hyvät taustatiedot ohjelmointitekniikoista. Ohjelmointitaitoja omaamattomille loppukäyttäjille pyritään kehittämään varsinaisia XML-kyselykieliä. Laskennallisesti täydellisen kielen käyttö vaatii runsaasti tietämystä esimerkiksi kyseisen kielen syntaksista, ohjausrakenteista, muuttujista (niiden alustamisesta) ja käsiteltävän tiedon organisoinnista.

SQL/XML-kieli on myös laskennallisesti täydellinen, mikäli XML-attribuutteja käsitellään XQuery:llä. Mikäli niitä käsitellään XPathilla, ei SQL/XML:llä ole laskennallisesti täydellistä ilmaisuvoimaa.

Luvussa 7 esitellyn logiikkaohjelmointilähestymistavan soveltaminen XML-dokumenteissa ja niihin perustuvan logiikkaohjelmointipohjaisen kyselyn voitaisiin ajatella myös tarjoavan laskennallisesti täydellisen ilmaisuvoiman. Logiikkaohjelmointikieliä ei kuitenkaan kuvan 26 taulukkoon ole liitetty, jotta voitaisiin keskittyä arvioimaan pelkästään varsinaisia XML-kyselykieliä. Ilmaisuvoima logiikkaohjelmoinnilla on kaiken kaikkiaan laskennallisesti täydellinen, kun taas deklaratiivisuus logiikkaohjelmointia osaamattoman loppukäyttäjän näkökulmasta on heikko.

Viiteluettelo

[Almendros-Jiménez, 2008] Jesús M. Almendros-Jiménez, An RDF query language based on logic programming. *Electronic Notes in Theoretical Computer Science* **200** (2008), 67-85.

[Almendros-Jiménez et al., 2009] Jesús M. Almendros-Jiménez, Antonio Becerra-Terón and Francisco J. Enciso-Baños, Integrating XQuery and logic programming. In: Dietmar Seipel, Michael Hanus and Armin Wolf (eds.), *Applications of Declarative Programming and Knowledge Management*, *Lecture Notes in Computer Science* **5437** (2009), Springer, 117-135.

[Almendros-Jiménez et al., 2007] Jesús M. Almendros-Jiménez, Antonio Becerra-Terón and Francisco J. Enciso-Baños, Querying XML documents in logic programming. *Theory and Practice of Logic Programming* **8**, 3 (May 2008), Cambridge University Press, 323-361.

[Arocena and Mendelzon, 1999] Gustavo O. Arocena and Alberto O. Mendelzon, WebOQL: restructuring documents, databases and webs. *Theory and Practice of Object Systems* **5**, 3 (Aug. 1999), 127-141.

[Bonifati and Ceri, 2000] Angela Bonifati and Stefano Ceri, Comparative analysis of five XML query languages. *ACM SIGMOD Record* **29**, 1 (Mar. 2000), 68-79.

[Brabrand et al., 2008] Claus Brabrand, Anders Møller and Michael I. Schwartzbach, Dual syntax for XML languages. *Information Systems* **33** (2008), 385-406.

[Buneman et al., 1996] Peter Buneman, Susan Davidson, Gerd Hillebrand, Dan Suciu, A query language and optimization techniques for unstructured data. In: *Proc. of the 1996 ACM SIGMOD international conference on Management of data*, 505-516.

[Ceri et al., 1999] Stefano Ceri, Sara Comai, Ernesto Damiani, Piero Fraternali, Stefano Paraboschi and Letizia Tanca, XML-GL: a graphical language for querying and restructuring XML documents. *Computer Networks* **31** (1999), 1171-1187.

[Chamberlin et al., 2001] Don Chamberlin, Jonathan Robie, Daniela Florescu, Quilt: an XML query language for heterogenous data sources. In: *The World Wide Web and Databases, Lecture Notes in Computer Science* **1997** (2001), Springer, 1-25.

[Connolly and Begg, 2005] Thomas M. Connolly and Carolyn E. Begg, *Database Systems, A Practical Approach to Design*, *Implementation and Management*. Addison-Wesley, 2005.

[Deutch et al., 1999] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy and Dan Suciu, A query language for XML. *Computer Networks* **31** (1999), 1155-1169.

[Erwig, 2003] Martin Erwig, Xing: a visual XML query language. *Journal of Visual Languages and Computing* **14** (2003), 5-45.

[Gou and Chirkova, 2007] Gang Gou and Rada Chirkova, Efficiently querying large XML data repositories: a survey. *IEEE Transactions on Knowledge and Data Engineering* **19**, 10 (Oct. 2007), 1381-1403.

[Graaumans, 2004] Joris Graaumans, A qualitative study to the usability of three XML query languages. In: *Proc. of the conference on Dutch directions in HCI* (Dutch HCI '04), ACM.

[Holzner, 2003] Steven Holzner, *Real World XML*. Peachpit Press, 2003.

[ISO/IEC 9075-14, 2003] ISO/IEC 9075-14:2003, *Information technology -- Database languages -- SQL -- Part 14: XML-Related Specifications (SQL/XML)*. URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35341

[ISO/IEC 9075, 2008] ISO/IEC 9075-1:2008, *Information technology -- Database languages -- SQL -- Part1: Framework (SQL/Framework)*. URL: http://www.iso.org/iso/catalogue_detail.htm?csnumber=45498

[Ives and Lu, 2000] Zachary G. Ives and Ying Lu, XML query languages in practice: an evaluation. *Web-Age Information Management, Lecture Notes in Computer Science* **1846** (2000), 29-42.

[Konopnicki and Shmueli, 1995] David Konopnicki and Oded Shmueli, W3QS: a query system for the world wide web. In: *Proc. of the 21st International Conference on Very Large Databases* (VLBD '95), Morgan Kaufmann Publishers Inc., 54-65.

[Lampathaki et al., 2009] Fenareti Lampathaki, Spiros Mouzakitis, George Gionis, Yannis Charlabidis and Dimitris Askounis, Business to business interoperability: A current review of XML data integration standards. *Computer Standards & Interfaces* **31**, 6 (Nov. 2009), 1045-1055.

[Niemi et al., 2009] Timo Niemi, Turkka Näppilä and Kalervo Järvelin, A relational data harmonization approach to XML. *Journal of Information Science* **35**, 5 (Oct. 2009), 571-601.

[Niemi and Järvelin, 2006] Timo Niemi and Kalervo Järvelin, *Another Look at XML*. Working paper, University of Tampere, Dept. of Computer Sciences, **A-2006-1**, 2006.

[Näppilä et al., 2010] Turkka Näppilä, Katja Moilanen ja Timo Niemi, *RXQL: An SQL-like Query Language for Selecting, Harmonizing, and Aggregating Data from Heterogeneous XML Data Sources*. University of Tampere, Dept. of Computer Sciences, Report **A-2010-2**, 2010.

[Robie, 1999] Jonathan Robie, *XQL FAQ*. URL: http://www.ibiblio.org/xql/

[Rys et al., 2005] Michael Rys, Don Chamberlin and Daniela Florescu, XML and relational database systems: the inside story. In: Özcan F. (ed.), *Proc. of the 2005 ACM SIGMOD international conference on Management of data*, ACM Press, 945-947.

[Simeoni et al., 2002] Fabio Simeoni, Paolo Manghi, David Lievens, Richard C. H. Connor and Steve Neely, An approach to high-level language bindings to XML. *Information and Software Technology* **44**, 4 (2002), 217-228.

[SQLX.org, 2004] *SQL/XML, SQL & XML Working Together*. URL: http://www.sqlx.org

[SuomiSanakirja.fi] *SuomiSanakirja.fi*. URL: http://suomisanakirja.fi

[Saïd and Evrard, 2001] Tazi Saïd and Fabrice Evrard, Intentional structures of documents. In: *Proc. of the 12th ACM conference on Hypertext and Hypermedia* (HYPERTEXT '01), ACM, 39-40.

[Vadaparty et al., 1993] Kumar V. Vadaparty, Yuksel A. Aslandogan and Gultekin Ozsoyoglu, Towards a unified visual database access. In: *Proc. of the 1993 ACM SIGMOD conference on Management of data* (SIGMOD '93), ACM, 357-366.

[W3C, 1999a] The World Wide Web Consortium (W3C), *HTML 4.01 Specification*. URL: http://www.w3.org/TR/html4/

[W3C, 1999b] The World Wide Web Consortium (W3C), *HTML 4.01 Specification, 3. On SGML and HTML*. URL: http://www.w3.org/TR/html4/intro/sgmltut.html

[W3C, 2001] The World Wide Web Consortium (W3C), *XML Schema: Formal Description*. URL: http://www.w3.org/TR/xmlschema-formal/

[W3C, 2004] The World Wide Web Consortium (W3C), *RDF/XML Syntax Specification*. URL: http://www.w3.org/TR/rdf-syntax-grammar/

[W3C, 2006] The World Wide Web Consortium (W3C), *Extensible Markup Language (XML) 1.1 (Second Edition)*. URL: http://www.w3.org/TR/xml11/

[W3C, 2007a] The World Wide Web Consortium (W3C), *XQuery 1.0: An XML Query Language*. URL: http://www.w3.org/TR/xquery/

[W3C, 2007b] The World Wide Web Consortium (W3C), *XSL Transformations (XSLT) Version 2.0*. URL: http://www.w3.org/TR/xslt20/

[W3C, 2007c] The World Wide Web Consortium (W3C), *XML Path Language (XPath) 2.0*. URL: http://www.w3.org/TR/xpath20/

[Xu and Papakonstantinou, 2005] Yu Xu and Yannis Papakonstantinou, Efficient keyword search for smallest LCAs in XML databases. In: *Proc. of the 2005 ACM SIGMOD international conference on Management of data* (SIGMOD'05), ACM, 527-538.