Tampereen yliopisto

Heini Anttila

RELATIONAALISEN JA XML-DATAN INTEGROINTI

Informaatioteknologian ja viestinnän tiedekunta
Pro gradu -tutkielma
Toukokuu 2020

TIIVISTELMÄ

Heini Anttila: Relationaalisen ja XML-datan integrointi
Pro gradu -tutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Toukokuu 2020

Relaatiotietokannat ovat laajalti käytetty tietokantastandardi tehokkaaseen tiedon talletukseen ja ylläpitoon. Viime vuosikymmeninä XML-muodossa (eXtensible Markup Language) oleva data on tullut suosituksi tiedon siirrossa eri järjestelmien välillä. XML-tieto voidaan kääntää relaatiotietokantaan ja päinvastoin. Näihin tiedon muunnoksiin ja siirtoihin on olemassa eri tyyppisiä metodeja. Metodeja on kehitelty ja kehitellään tehokkuuden ja virheettömyyden parantamiseksi.

Relationaalinen ja XML-data täydentävät toisiaan yritysten tiedonhallinnassa. XML on alustariippumaton, joten XML-tiedostoja voidaan muodostaa erityyppisissä järjestelmissä tai alustoilla. Tietoja voidaan siirtää tai lähettää vaikka toisentyyppisiin järjestelmiin. Samoja tiedostoja voidaan toimittaa eri vastaanottajille. Vastaanottaja voi valita aineistoista kiinnostavat osat ja jättää huomiotta ne tiedot, joita ei tarvita. Tietoa voi saada XML-muodossa monesta tietolähteestä kuten ulkopuolisista järjestelmistä tai oman organisaation eri osastoilta. Nämä tiedot tai kiinnostavat osat niistä muokataan ja talletetaan relaatiokantaan esim. tiedon varastointia varten. Tutkielmassa perehdytään XML:n ja relaatiokantojen kääntämiseen ja yhteiskäyttöön.

Avainsanat: XML, relaatio, integrointi, tietokanta

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla.

Sisällys

1 Johdanto

Informaation integroinnin tarkoituksena on luoda yhtenäinen pääsy moneen itsenäiseen tietolähteeseen [Kozlova et al., 2007]. Datan yhteentoimivuuden ongelma saa alkunsa siitä, että tämän päivän verkostoituneessa maailmassa informaatio on hajallaan monessa tietolähteessä, monena eri kaaviona, ja jopa eri tietomalleina, kuten relationaalisessa ja XML-muodossa. Tällaisen datan integrointi ja muuntaminen on entistä tärkeämpää monissa sovelluksissa. [Yu and Popa, 2004; Kozlova et al., 2006] Tietolähteet ovat yleensä itsenäisiä ja tietolähteiden datan erilaisuus aiheuttaa ongelmia. Tämä ratkaistaan usein sopivilla muunnoksilla sekä siivoamalla tietolähteistä haettua dataa. [Lenzerini, 2002]

XML-dokumenttien tallennuksessa ja kyselyssä on neljä lähestymistapaa [Qtaish and Ahmad 2016]:

1) Tiedostojärjestelmä (a file system): Tässä lähestymistavassa XML talletetaan tekstitiedostona.

2) Oliopohjainen tietokanta (an object-oriented database (OODB)): Tässä lähestymistavassa XML talletetaan oliotietokantaan. XML-elementit ja alielementit voidaan ryhmitellä, mikä saattaa olla hyödyllistä monimutkaisille sovelluksille kuten multimedia ja maantieteellinen informaatio. Kuitenkaan nykyinen OODB sukupolvi ei pysty riittävän hyvin suurten tietokantojen kyselyihin.

3) Natiivi XML-tietokanta (a native XML database (NXD)): Tässä lähestymistavassa XML talletetaan XML-tietokantaan. XML voidaan tallettaa ja kysellä alkuperäisessä muodossa eikä kääntämistä tarvita. Tässä tavassa on vielä vaikeuksia käsitellä isoa määrää XML-dataa. Tapa ei sovi, kun tarvitaan heterogeenisten XML-dokumenttien integrointia.

4) Relaatiotietokanta (a relational database (RDB)): Tässä lähestymistavassa XML talletetaan relaatiotietokantaan. Tämä tapa päihittää muiden lähestymistapojen haitat tallennuksessa, indeksoinnissa, eheydessä, turvallisuudessa ja kyselyissä. Tämä tapa on sopivin XML-datan käsittelyyn.

Lenzerini [2002] esittää teoreettisen näkökulman tiedon integrointiin. Tiedon integrointijärjestelmiä luonnehditaan arkkitehtuurina, joka perustuu globaaliin rakenteeseen ja tie-

tolähdejoukkoon. Tietolähteet sisältävät todellista dataa ja globaali rakenne tarjoaa rekonstruoidun, integroidun ja virtuaalisen näkymän tietolähteisiin. Tietolähteiden ja globaalin rakenteen välisen suhteen mallintaminen on siksi ratkaiseva. Tähän on ehdotettu kahta lähestymistapaa. Ensimmäisessä, globaalissa tavassa globaali rakenne pitää kuvata tietolähteiden termein (global-as-view GAV). Toisessa, paikallisessa tavassa globaali rakenne pitää määritellä tietolähteistä riippumatta, globaalin rakenteen ja tietolähteiden väliset suhteet perustetaan määrittelemällä jokainen tietolähde globaalin rakenteen näkymänä (local-as-view LAV).

- LAV-lähestymistavassa tietolähteet mallinnetaan joukkona globaalin rakenteen näkymiä mieluummin kuin tietokannan datana suoraan. Kyselyn uudelleenkirjoittaminen jaetaan kahteen osaan. Ensimmäisessä osassa kysely kirjoitetaan annetulla kyselykielellä. Toisessa osassa kehitetään kyselyä kirjoittamalla näkymät.

- Useimmilla GAV-integrointijärjestelmillä ei ole eheysrajoituksia globaalissa rakenteessa. Kyselyn prosessointi perustuu yksinkertaiseen kehitykseen.

Bernstein ja Haas [2008] opastaa työkaluihin ja perusteknologioihin, kun yhdistetään informaatiota eri tietolähteistä.

- *Tietovaraston lataus (Data Warehouse Loading)*. Tietovarasto on tietokanta, jossa on dataa monista tietolähteistä. Siinä datan eroja siivotaan ja sovitetaan yhteen. ETL-työkalut (Extract-Transform-Load) helpottavat työtä.

  Jos ohjelmisto on oliopohjainen ja data on relaatiotietokannassa, niin sovelluksen ja relaationaalisen integroinnin käännökset voivat olla mutkikkaita.

  XML:ssä luodaan integroitu näkymä eri tietolähteiden datasta. Usein jokin tietolähde on epätäydellinen tai muista puuttuu mitä jossakin on. Vain kiinnostavat elementit otetaan mukaan. XML:n joustavuus sopii eri esitysmuotoisen datan integrointiin.

- *Kaaviostandardit*. On helpompi integroida eri tietolähteiden dataa, jos ne käyttävät samaa kaaviota. Silloin ei ole tarvetta muokata dataa ennen integrointia. Silloinkin kun tietolähteillä ei ole yhteistä kaaviota, tietolähteellä voi olla yleinen standardi, alakohtainen tai yrityskohtainen. Siten tietolähteet voidaan liittää yhteen muokkaamalla käännökset standardin mukaisiksi. Koska

tämä tapa sallii vain standardin mukaisen tiedon integroinnin, osa tiedosta katoaa.

- *Tiedon siivoaminen*. Yksi tärkeä tiedon siivoustapa on poistaa duplikaatit, kun samaan entiteettiin viitataan eri tietolähteistä. Esim. postituslistoilla voi olla eri tavalla kirjoitettuja osoitteita. Joskus duplikaatit voivat olla säilytettävää tietoa, esim. samannimiset henkilöt.

- *Kaavion kääntäminen*. Perusoperaatio on tunnistaa, kuinka lähdekaaviosta muodostuu integroitu kohdekaavio. Työkalut näyttävät yleensä kolme pystysuoraa ruutua. Vasemmalla ja oikealla ovat kaaviot ja niiden väliin suunnittelija määrittelee kääntämisen viivoja ja muunnoksia piirtämällä.

- *Tiedonpoiminta (Information Extraction).* Tiedonpoiminta on yleistermi tekniikoille, jotka tuottavat rakenteellista tietoa vapaamuotoisesta tekstistä.

Edellä esiteltiin neljä XML:n tallennustapaa, joista relaatiokanta päihittää muut tallennustavat [Qtaish and Ahmad 2016]. Tiedon integrointi on eri tietolähteissä olevan tiedon yhdistämistä ja yhtenäisen näkymän tarjoamista näihin tietoihin. Esimerkkinä oli yleiskatsaus LAV- ja GAV-lähestymistavoista [Lenzerini 2002]. Bernstein ja Haas [2008] opasti työkaluihin ja perusteknologioihin, kun yhdistetään informaatiota eri tietolähteistä.

Tässä työssä käsitellään XML:n ja relaatiokantojen kääntämistä ja yhteiskäyttöä. Luvussa 2 perehdytään relaation ja XML:n perusteisiin, sekä miten ne eroavat toisistaan. Luvussa 3 tutustutaan XML:n ja relaatiokantojen integrointiin. Luku 4 sisältää XML:n tallennusta relaatiotietokantaan. Luvuissa 5 ja 6 on XML:n kääntämisestä relaatiotietokantaan ja päinvastoin. Luku 7 käsittelee käsitteellistä mallintamista tiedon integroinnissa.

2 Relationaalinen ja XML-data

2.1 Relaatio

Tiedot tallennetaan relaatioihin noudattaen relaatiokaavioita (relation schema). Relaatiot ovat relaatiokaavioiden ilmentymiä esim. tietyllä hetkellä. Relaatio voidaan esittää taulukkona, esimerkkinä taulut Henkilö, Koira ja Omistaja.

Henkilö

HenkilöId	Nimi	Puhnro
100	Matti	0401234567
101	Maija	0402345678
110	Kalle	0403456789
111	Kaisa	0404567890

Koira

KoiraId	Nimi	Rotu
200	Murre	Saksanpaimenkoira
201	Sisu	Villakoira
210	Hilla	Beagle

Omistaja

HenkilöId	KoiraId
100	200
100	201
101	201

- relaatiokaavio määrittelee relaation rakenteen
  - mitkä attribuutit
  - attribuuttien arvojoukot
  - miten attribuutit tutkitaan
- relaatio = taulu, taululla on yksikäsitteinen nimi
- monikko = taulun rivi (kaikki rivit ovat erilaisia)
- attribuutti = sarakkeen yksikäsitteinen nimi

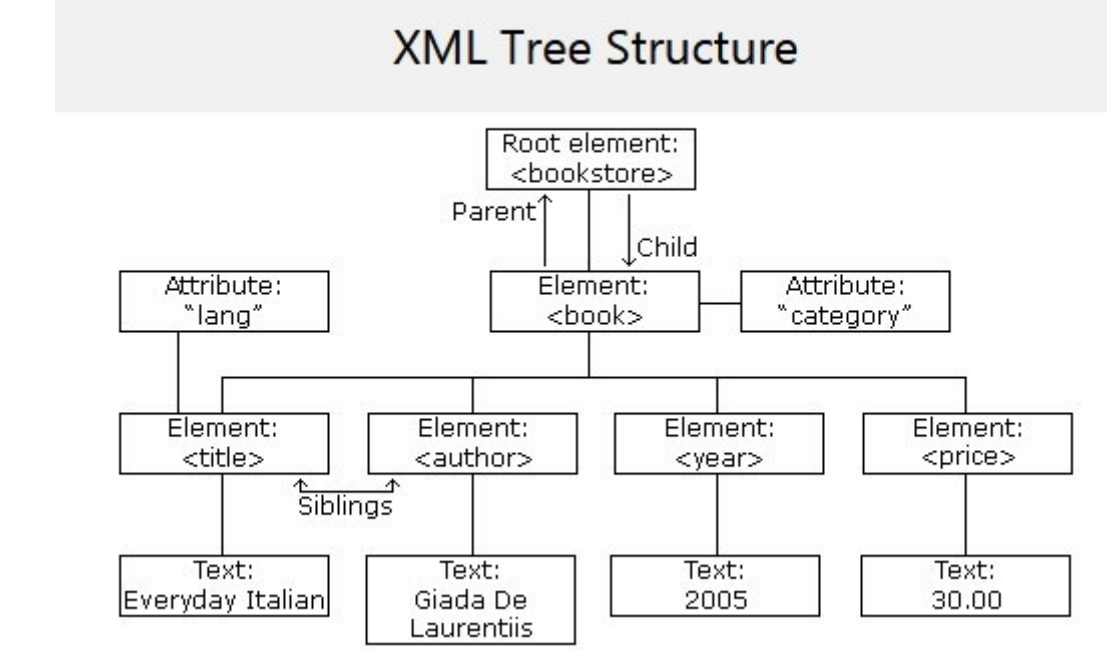
Relaatiotietokanta (relational database) muodostuu useasta relaatiosta.
Relaatiotietokantakaavio (relational database schema) on relaatiokaavioiden kokoelma. [Laine, 2005]

2.2 XML

XML-kaaviomalli sisältää säännöt siitä millainen XML-datatiedosto on. XML-tiedoston on oltava mallin sääntöjen mukainen, jotta dataa voidaan hyödyntää. XML-data voidaan esim. viedä tietokannan tauluihin. [W3c, 2020]

DTD (Document Type Definition) on vanhempi malli, joka sisältää rakenteen ja sanaston. XML-skeema (XML Schema Definition) on uudempi malli ja sisältää tarkemman määrittelyn kuten tietotyypin. XML-skeema alkoi syrjäyttää DTD:n 2000-luvun alkupuolella. [W3c, 2020]

Datatiedoston lyhenne on .xml, DTD-tiedoston lyhenne on .dtd ja XML-skeematiedoston lyhenne on .xsd. [W3c, 2020]



Kuva 1. XML-puun rakenne [W3schools, 2020].

XML-puurakenne voidaan mallintaa viiden tyyppisillä solmuilla: juuri (root), elementti (element), teksti (text) - ei varattuja merkkejä, merkkidata (cdata) - mitä tahansa merkkejä, ja attribuutti (attribute). Lisäksi voidaan käyttää mm. solmutyyppejä nimiavaruus (namespace), prosessointiohje (processing-instruction), dokumentti (document) ja kommentti (comment). [W3c, 2020] Kuvassa 1 on esimerkki XML-puun rakenteesta.

Solmujen väliset suhteet ovat lapsisolmu (child node), vanhempi (parent node), jälkeläinen (descendant node), esi-isä (ancestor node), ja sisar (sibling node). [W3c, 2020]

DTD

DTD-esimerkki [Heikniemi, 2001]

```
<!ELEMENT puhelinluettelo (henkilo)+>
<!ELEMENT henkilo (nimi, osoite, puhnro*, kuvaus?)>

<!ELEMENT nimi (#PCDATA)>
<!ELEMENT osoite (#PCDATA)>
<!ELEMENT puhnro (#PCDATA)>
<!ELEMENT kuvaus (#PCDATA)>

<!ATTLIST puhnro
    tyyppi     CDATA    #REQUIRED>
```

DTD:n ensimmäinen rivi kertoo, että dokumentissa esiintyy puhelinluettelo-elementtejä. Rivin loppuosa (henkilo)+ ilmaisee, että puhelinluettelon sisältönä voi olla yksi tai useampia henkilo-elementtejä. Plus-merkki tarkoittaa siis ilmaisua "yksi tai useampi kappale". [Heikniemi, 2001]

Toinen rivi ilmaisee, että henkilo -nimisen elementin sisältönä on nimi-, osoite-, puhnro- ja kuvaus -elementit. Puhnro-elementin nimen perässä oleva tähti tarkoittaa, että puhelinnumeroita voi olla nolla tai useampia. Kuvaus-sanan perässä oleva kysymysmerkki puolestaan kertoo, että kuvaus-elementtejä esiintyy nolla tai yksi kappaletta, eli kuvaus voi olla olemassa tai sitten ei. Seuraavassa kyseisen DTD:n mukaisessa XML-esimerkissä kuvausta ei käytetty ollenkaan. [Heikniemi, 2001]

Seuraavat neljä riviä määrittelevät, että nimi-, osoite-, puhnro- ja kuvaus -elementit voivat sisältää vain tekstiä (#PCDATA, engl. Parsed Character Data), eivätkä siis lainkaan toisia elementtejä. Loppuosa DTD:stä sisältää ATTLIST-määrittelyn, joka kertoo, että puhnro-nimisellä elementillä voi olla tyyppi-niminen attribuutti, jonka sisältö on tyyppiä CDATA (merkkijono, engl. Character Data). [Heikniemi, 2001]

XML

XML-esimerkki [Heikniemi, 2001]

```
<puhelinluettelo>
   <henkilo>
       <nimi>Jouni Heikniemi</nimi>
       <osoite>Kotikuja 2 C 57</osoite>
       <puhnro tyyppi="koti">07-2345678</puhnro>
   </henkilo>
   <henkilo>
       <nimi>Simo Simpukka</nimi>
       <osoite>Jugendstrasse 8 Z 22</osoite>
       <puhnro tyyppi="gsm">048-3322211</puhnro>
       <puhnro tyyppi="koti">07-8765432</puhnro>
```

```
    </henkilo>
</puhelinluettelo>
```

XML-skeema: XSD

XSD-esimerkki [Singh, 2018]

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://www.beginnersbook.com"
xmlns="https://www.beginnersbook.com"
elementFormDefault="qualified">

<xs:element name="beginnersbook">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="to" type="xs:string"/>
   <xs:element name="from" type="xs:string"/>
   <xs:element name="subject" type="xs:string"/>
   <xs:element name="message" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>

</xs:schema>
```

<xs:element name=”beginnersbook”> - elementin nimi on beginnersbook

<xs:complexType> - beginnersbook -elementin tyyppi

<xs:sequence> - beginnersbook -elementin complexType on sequence

<xs:element name=”to” type=”xs:string”> elementin “to” tyyppi on merkkijono

2.3 Relationaalisen ja XML-datan vertailua

XML-data voidaan luoda yhdessä järjestelmässä ja käsitellä toisessa järjestelmässä riippumatta laiteympäristöstä tai käyttöjärjestelmästä. Siirrettävyyden ja ympäristöriippumattomuuden takia XML soveltuu tietojen vaihtoon eri järjestelmien välillä. [Microsoft, 2020]

Internetissä on saatavilla suuri määrä elektronista dataa, joten informaation esityksestä ja vaihdosta internetissä on tullut tärkeää [Ma and Yan, 2007]. Monet yritykset käyttävät hierarkkisia, relaatio- ja objekti-relaatio -tietokantoja Internetin päivittäisissä vuorovaikutuksissa käyttäjien ja yritysten välillä. Relaatiotietokanta ei kuitenkaan ole tehokas datan määrän kasvaessa tai datan siirrossa. [Fong and Cheung, 2005]

Internetissä XML on yksinkertainen ja tehokas formaatti datan esittämiseen ja tallentamiseen [Fong and Cheung, 2005]. XML ja vastaavat standardit sallivat datan vaihdon avulla sovellusten helpon kehittämisen [Ma and Yan, 2007]. XML-tietokannalla on kui-

tenkin rajoituksensa [Fong and Cheung, 2005]. XML:n käsittely vaatii monenlaisten monimutkaisten tyyppien ja kyselyjen, kuten myös hierarkkisen rakenteen käsittelyä, jota relaatiomallilla ei ole [Yu and Popa, 2004]. Lisäksi XML:ltä puuttuu riittävä kyky datan ja niiden monimutkaisten sisäisten suhteiden mallintamiseen [Ma and Yan, 2007]. XML tuo mukanaan uusia haasteita kyselynprosessointiin, datan muokkaamiseen sekä indeksointiin. On säilytettävä dokumentin järjestys, suorittaa rakenteista navigointia, sekä tukea dynaamisesti rakennettuja XML-solmuja. [Pal et al., 2005]

On olemassa tietokantoja, joissa välitetään vain uusimmasta tiedosta, mutta on myös monia yrityksiä, joissa sisällön muutosten historian säilyttäminen on tärkeää. Tehokas versionhallinta on edelleen haaste useimmissa kaupallisissa tietokannan hallintajärjestelmissä, siitäkin huolimatta, että aiheesta on tehty paljon tutkimusta. Toisaalta, XML on tuonut mukanaan uusia, yksinkertaisempia ja suoritustehokkaampia tekniikoita. [Moro et al., 2007]

XML-datan tallentaminen, kysely ja päivittäminen vaatii XML:n ja tietokantojen integrointia. Monia tietokantoja, kuten relaatio-, olio-, sekä objekti-relaatio -tietokantoja, on käytetty XML-dokumenttien kääntämiseen. Näistä relaatiotietokannat saattavat olla lupaavampi vaihtoehto laajan käyttöasteen ja valmiiden tekniikoiden takia. [Ma and Yan, 2007]

XML:n suurimpia etuja ovat sen alustariippumattomuus sekä joustavuus. Tietomallina XML sopii mille tahansa yhdistelmälle rakenteista, rakenteetonta ja puolirakenteista dataa. XML:ää on helppo laajentaa, sillä uusia tageja voidaan määritellä tarpeen mukaan. XML-dokumentit voidaan myös helposti muuttaa erinäköiseksi XML:ksi ja jopa toiseen formaattiin kuten HTML:ksi. Lisäksi XML-dokumenttien vastaavuus kaavion kanssa voidaan helposti tarkistaa. Kaikki tämä on tullut mahdolliseksi yleisesti saatavilla olevien työkalujen ja standardien, kuten XML-jäsennyksen, XSLT:n, sekä XML-skeeman ansiosta. [Nicola and van der Linden, 2005] XML mahdollistaa myös XML-datan rakenteellisten piirteiden, kuten dokumenttijärjestyksen ja rekursiivisten rakenteiden, säilyttämisen paremmin. [Pal et al., 2005]

XML:ää käytetään yhä etenevässä määrin yrityssovelluksissa puolirakenteisen ja rakenteettoman datan mallinnukseen, sekä datalle, jonka rakenne vaihtelee paljon. [Pal et al., 2004] XML-data on hierarkkista, puolirakenteista, ja järjestettyä, kun taas relationaalinen data on hierarkkitonta, rakenteista, ja järjestämätöntä. [Näppilä et al., 2011] Yritysten ohjelmistot käyttävät XML:ää puolirakenteisen datan mallintamiseen esimerkiksi dokumentinhallinnassa. [Pal et al., 2005]

Yritykset pitävät pysyvästi suuria määriä dataa XML-muodossa. Tälle on monta syytä. Joidenkin yritysten täytyy säilyttää XML-dokumentit alkuperäisessä muodossa säädösten takia. [Nicola and van der Linden, 2005] Tällöin käytetään niiden alkuperäistä XML-skeemaa. [Moro et al., 2007] Tyypillisiä esimerkkejä ovat lakiin ja rahoitukseen liittyvät dokumentit. Toinen syy XML:n käyttämiseen pysyvänä säilytysmuotona on se, että XML on sopivampi tietomalli kuin relaatiokaavio. Esimerkiksi joillain aloilla data on luonteeltaan hyvin monimutkaista ja hierarkkista, ja saattaa silti sisältää suuria määriä rakenteetonta informaatiota. [Nicola and van der Linden, 2005]

XML-skeemat ovat yleensä niin monimutkaisia, että XML-datan purkaminen relaatiotietomalliin johtaa suureen määrään tauluja. Tämä tekee XML-datan purkamisesta relaatiotauluun monimutkaisen, uudelleenkokoamiskustannukset korkeiksi, ja kyselyt hyvin mutkikkaiksi. Lisäksi muutosten tekeminen XML-skeemaan vaatii huomattavaa tietokantakaavion ja sovelluksen ylläpitoa. [Pal et al., 2005] Esimerkiksi terveydenhuoltoalalla XML on laajalti käytetty tietokannoissa terveystietojen metadatan jakamiseen. Metadatakaavio voi sisältää satoja muunnoksia, jotta se tukisi kaikenlaisten lääketieteellisten dokumenttien kyselyä. Näillä sadoilla tyyppien muunnoksilla voi olla yhteinen pohja sekä lisänä erityisiä yksittäisiä laajennoksia. Tällaisen metadatan jättäminen relationaaliseen muotoon johtaa suureen määrään tauluja sekä huonoon tehokkuuteen. Lisäksi, jotta uusi dokumenttityyppi saadaan lisättyä, vie muutosten tekeminen relaatiokaavioon viikkoja. [Moro et al., 2007]

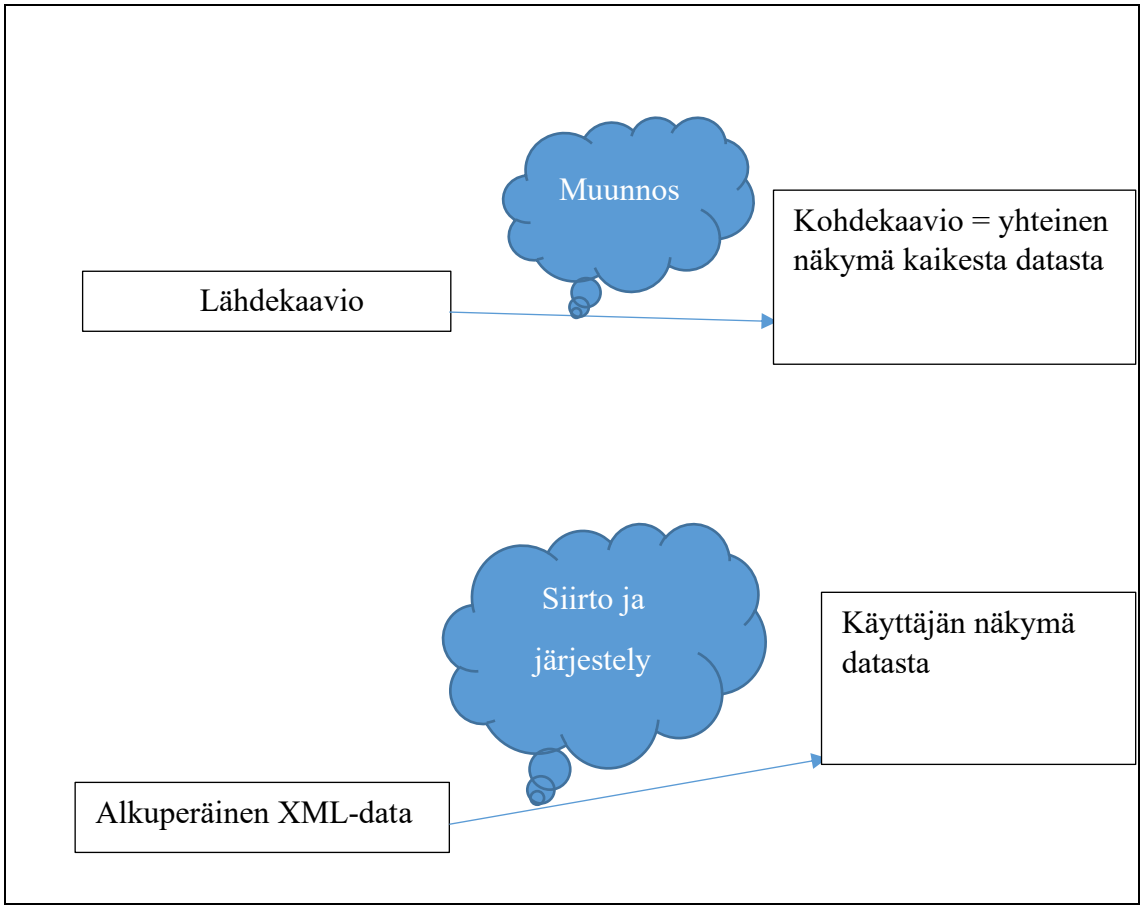
Skeeman evoluutio on sarja muutoksia XML-skeemaan sen eliniän aikana. Tällaiset muutokset yleensä tapahtuvat uusien tai muuttuvien yrityksen tarpeiden takia. Esimerkiksi uusien palvelujen, tuotteiden, yritysprosessien muuttaminen tai lisääminen voivat kaikki johtaa uusiin vaatimuksiin tiedonhallinnassa. Kaikki tämä voi johtaa muutoksiin XML-skeemassa. [Nicola and van der Linden, 2005] Joillakin aloilla tietotyyppeihin tulee viikoittaisia muutoksia, mikä vaatii jatkuvaa muutosten tekemistä tietomalliin ja kaavioon. Tällöin relationaalisen formaatin käyttäminen vaatii pitkäkestoisia tietokantakaavion muutoksia ja datan siirtämistä, millä puolestaan voi olla vaikutuksia yrityksen toimintaan. [Moro et al., 2007] Kaavion muutosten takia relationaalisessa datassa voidaan joutua lisäämään sekä poistamaan tauluja ja attribuutteja, päivittämään tietotyyppejä ja nimiä, sekä päivittämään rajoitteita. Relationaalisen tietokannan kaavion päivitys ei ole helppoa, sillä se yleensä vaatii kallista datan siirtoa. Tietokannan hallintajärjestelmässä, jossa jokainen XML-dokumentti on yhteydessä eri kaavioon tai niillä ei ole kaaviota laisinkaan, XML-skeeman muutokset eivät välttämättä vaadi datan siirtoa. Pikemminkin sovelluksen

datan tarvitsee olla vakaa eri XML-skeema versioiden suhteen. Datan, jolla on usein vaihteleva, tai nopeasti muuttuva kaavio, olisi hyvä olla XML-muodossa. Dataelementillä on usein vaihteleva kaavio, kun kaavion elementin sisältö voi ottaa useita rakenteellisia muotoja. Kaavion vaihtelevuus on avainasia päätettäessä pitäisikö tiedon olla relationaalisessa vai XML-muodossa. [Moro et al., 2007] Kuvassa 2 kerrotaan lyhyesti mitä eroa on relationaalisella ja XML-datalla.

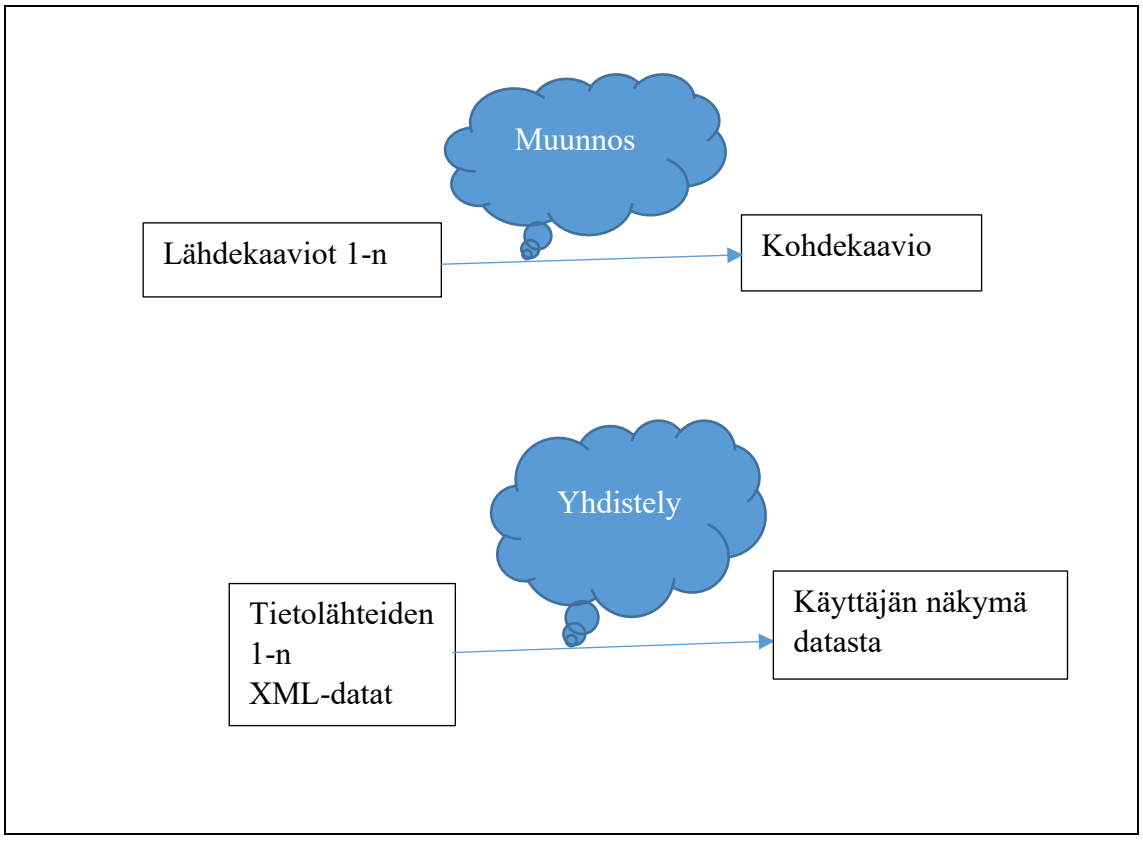
Relational	XML
Regular structure	Heterogeneous structure
Flat data	Nested elements on several levels
The order has no importance	The Order has an importance
Static schemas	Schemas tend to be more extensible
Always have a schema	May or may not have a schema

Kuva 2. Relationaalisen ja XML-datan eroja [Nassiri et al., 2017].

On kaksi datan yhteentoimivuuden perusmuotoa: datan vaihto ja datan integrointi. Datan vaihto, tai datan muuntaminen, on datan siirtämistä ja uudelleenjärjestämistä yhdestä tai useasta lähdekaaviosta kohdekaavioon. [Yu and Popa, 2004] Kohdekaavio on yhteinen näkymä kaikelle datalle, josta käyttäjä voi kysellä tietoa. Lähdekaavio kuvaa alkuperäisen datan rakenteen. [Lenzerini, 2002] Datan vaihto tulee esiin monissa tehtävissä, jotka vaativat datan siirtoa itsenäisten sovellusten välillä, jotka eivät välttämättä käytä samaa datan esitysmuotoa [Yu and Popa, 2004]. Kuvassa 3 on esitetty datan vaihto. Sitä vastoin datan integroinnissa yhdistellään dataa eri tietolähteistä, ja annetaan käyttäjälle tästä datasta yhtenäinen näkymä [Lenzerini, 2002]. Kuvassa 4 on esitetty datan integrointi. Molemmissa tapauksissa suhteet tai käännökset täytyy ensin luoda lähdekaavioiden ja kohdekaavion välille [Yu and Popa, 2004]. Tämän käännöksen tekeminen on yksi tärkeimmistä tehtävistä datan integrointijärjestelmän suunnittelussa [Lenzerini, 2002]. Käännökset määräävät miten toisiinsa liittyvien elementtien ryhmät lähdekaaviossa vastaavat toisiinsa liittyvien elementtien ryhmiä kohdekaaviossa. Käännökset voidaan antaa joko käyttäjän toimesta tai ne voidaan johtaa puoliautomaattisesti kaavioiden yhteensovitusalgoritmien tulosten perusteella. Käännöksiä on käytetty kyselyn uudelleenkirjoittamiseen relationaalisen datan integrointijärjestelmissä. Niitä on käytetty myös relationaalisen datanvaihtojärjestelmien muodolliseen määrittelyyn. [Yu and Popa, 2004]



Kuva 3. Datan vaihto.



Kuva 4. Datan integrointi.

On kaksi vallitsevaa tapaa mallintaa ja esittää dataa [Näppilä et al., 2011]. Ensimmäinen tapa käyttää XML-tietokantoja XML-datan tallentamiseen ja hallintaan suoraan. Toinen tapa on tallentaa XML-data relaatiotietokantoihin, jolloin hyödynnetään relaatiotietokantojen kehittynyttä teknologiaa ja tuotteita. Jälkimmäisessä tapauksessa on suuri tarve tehokkaille menetelmille XML-skeemojen kääntämiseen relaatiokaavioihin. On tärkeää, että tapa, jolla DTD käännetään relaatiokaavioon, pystyy säilyttämään sekä DTD:n rakenteen, että sen sisältämät rajoitteet. [Lv and Yan, 2006] Relaatiotietokannan hallintajärjestelmät ovat pitkän aikaa olleet keskeisessä roolissa yritysten tiedonhallinnassa, sillä niillä voidaan tehokkaasti varastoida ja prosessoida suuria tietomääriä. [Näppilä et al., 2011] Samaan aikaan XML:stä (Extensible Markup Language) on tullut sekä johtava kuvauskieli dokumenttien esitykseen että standardi datan vaihtoon erilaisten järjestelmien, alustojen, sovellusten, ja yritysten välillä [Näppilä et al., 2011; Nicola and van der Linden, 2005]. XML-datan lisääntyessä Internetissä, on tärkeää kehittää tehokkaita menetelmiä XML-datan hallintaan ja tallentamiseen [Lv and Yan, 2006].

Datan tallentaminen XML:nä on entistä tärkeämpää, mutta useimpia olemassa olevia tietovarastoja hallitaan edelleen relaatiotietokantajärjestelmillä [Shui et al., 2005]. Vaikka paljon tärkeää dataa on relationaalisessa muodossa, on käännytty etenevässä määrin XML:n puoleen relationaaliseen malliin sopimattoman datan tallentamisessa [Moro et al., 2007]. Järjestelmille, jotka voivat samaan aikaan prosessoida sekä relationaalista että XML-dataa on kasvava tarve [Näppilä et al., 2011].

Relationaalinen ja XML-data täydentävät toisiaan yritysten tiedonhallinnassa [Moro et al., 2007]. Suuri osa teollisuudesta turvautuu jo olemassa oleviin relaatiotietokantoihin ja sovelluksiin, josta XML-dokumentin informaatio on luotu, tai johon informaatio XML-dokumenteista tallennetaan [Beyer et al., 2005]. Puolirakenteisena tietomallina XML on silta rakenteisten relationaalisten järjestelmien ja vapaamuotoisten tekstidokumenttien välillä. [Beyer et al., 2005] On melko yleistä, että yritysten sisäinen ja yritysten välinen viestintä tehdään XML:n avulla. XML:ää voidaan käyttää sekä rakenteisen säännöllisen informaation, kuten tilisiirtojen, että vähemmän rakenteisten epäsäännöllisten dokumenttien, kuten käyttöohjeiden ja uutisartikkelien, esitykseen. [Näppilä et al., 2011] Vaikka XML:n alkuperäinen tarkoitus oli tiedonvaihto, on se yhä useammin suunniteltu tallennettavaksi pysyvästi. [Beyer et al., 2005] XML-datan tallentaminen ja kyseleminen relaatiotietokannan hallintajärjestelmässä tuo kuitenkin mukanaan ongelman - minkä osan datasta jättää XML-muotoon ja minkä muuttaa relationaaliseksi dataksi [Moro et al., 2007].

3 Erilaisia tapoja integroida XML ja relationaalinen data

XML-tietomallilla on ominaisuuksia, jotka tekevät sen kääntämisen relationaaliseen tietomalliin todella hankalaksi, ellei jopa käytännössä mahdottomaksi. XML-data on hierarkkista ja sillä saattaa olla rekursiivinen rakenne. Relaatiotietokannoissa hierarkiat mallinnetaan viiteavaimien avulla. Dokumenttijärjestys on XML-ilmentymien luontainen ominaisuus ja se täytyy säilyttää kyselytuloksissa. Tämä on päinvastoin kuin relationaalisen datan kanssa, joka on järjestämätöntä, ja järjestys täytyy määrittää ylimääräisillä järjestyssarakkeilla. XML-osapuun uudelleenrakentaminen voi olla kohtuuttoman kallista. [Pal et al., 2004]

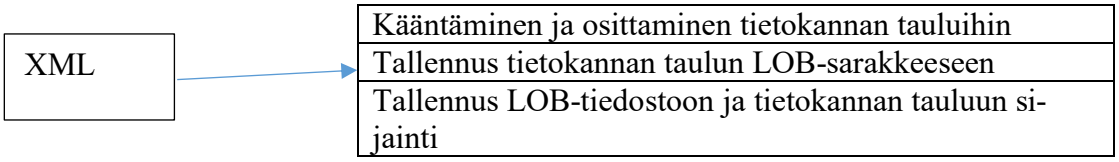
Suuret kaupalliset tietokannan hallintajärjestelmät (kuten IBM DB2, Oracle, ja Microsoft SQL Server) tukevat XML-datan tallentamista ja kyselyä relationaalisen datan lisäksi. Relaatiotietokantojen mallinnus on tutkitumpi aihe kuin XML-tietokantojen mallinnus. Useimmat lähestymistavat keskittyvät laajentamaan perinteisiä mallinnustekniikoita tukemaan myös XML-dataa. [Moro et al., 2007] XML tallennetaan useimmiten relaatiotietokantoihin, jotta pystytään hyödyntämään nopeita relationaalisia kyselyjä ja kehittyneitä tallentamismahdollisuuksia [Kolahi and Libkin, 2007].

Relaatiomalli on ollut hyvä käsittelemään yksinkertaista rakenteellista dataa. Suuret oliot voidaan tallentaa tietokantaan, tai tiedostoihin tietokannan ulkopuolelle niin että tietokantaan tallennetaan vain tieto siitä missä tiedosto sijaitsee. Suuret oliot LOBit (Large Object) on suunniteltu isolle rakenteettomalle datalle. LOBeja käytetään pääasiassa ison rakenteellisen datan kuten merkkijojon tai rakenteettoman datan kuten ison tekstidokumentin tallentamiseen ja manipulointiin. Esim. valokuvat ovat rakenteetonta dataa. Henkilön tiedoissa voi olla vaikkapa rakenteellista dataa ja rakenteeton valokuva. Silloin valokuvat voidaan tallettaa tietokantaan tai tiedostoihin sen ulkopuolelle. LOBit voivat olla tietotyyppiä BLOB (Binary Large Object), CLOB (Character Larger Object), riippuen tiedon pituudesta ja tyypistä. [Oracle, 2020]

Relaatiotietokannat voivat tukea XML-datan tallentamista, manipulointia, kyselyä, sekä palauttamista. Tämä on yleensä tehty tallentamalla XML-dokumentit suuriin objekteihin (LOB) tai kääntämällä ja osittamalla XML relaatiokaavioon. Näillä ratkaisuilla on toimintaan ja suorituskykyyn vaikuttavia rajoituksia. LOB-pohjainen säilytys mahdollistaa nopean kokonaisten dokumenttien lisäyksen sekä haun, mutta kärsii huonosta suorituskyvystä palauttamisessa. Tätä voidaan jonkin verran parantaa, jos dokumentin indeksointi hoidetaan lisäyksen aikana. Se saattaa nopeuttaa kyselyjä, jotka etsivät annettujen ha-

kuehtojen täyttäviä dokumentteja. Dokumenttien osien palauttaminen ja osadokumenttien päivittäminen vaatii kuitenkin edelleen kallista XML-jäsennystä. [Nicola and van der Linden, 2005]

XML relationaaliseen dataan:



Kun XML on kerran hajotettu relaatioiksi, on pelkällä SQL:llä suoritetut kyselyt ja päivitykset tehokkaita. Tällä lähestymistavalla on silti haittapuolensa. XML-skeemoissa voi olla monia sisäkkäisiä ja toistuvia elementtejä, jolloin vastaava relaatiokaavio koostuisi kymmenistä tai jopa sadoista tauluista. Tällaisen käännöksen määritteleminen XML:stä relaatiokaavioksi on monimutkainen tehtävä. Kun data on kerran lisätty, muutosten tekeminen relaatiokaavioon, XML-skeeman muuttumisen takia, on melkein aina mahdotonta. Tämä rajoittaa kovasti joustavuutta, mihin XML:ää usein käytetään. Myös XML-dokumenttien uudelleenrakentamisen vaatimat monimutkaiset liitokset voivat olla kalliita, kun käsitellään suuria määriä dataa. Tämän lisäksi voi olla, että monimutkaisia XQuery-kyselyitä ei edes pystytä kääntämään SQL:ksi. [Nicola and van der Linden, 2005]

Joissain tapauksissa on järkevämpää osittaa XML-dokumentit relaatiotaulun riveiksi ja sarakkeiksi, siitäkin huolimatta, että ne voitaisiin säilyttää XML-muodossa. Tietyissä käyttötilanteissa XML:ää käytetään vain datan kuljettamisessa tietokantaan. XML-rakenteesta tulee epäoleellista, kun data on kerran integroitu olemassa olevan relationaalisen datan kanssa. Esimerkiksi, jos sovellus saa kaiken oleellisen datan web-palvelujen viestistä ja purkaa datan tauluihin, silloin alkuperäistä XML-viestiä ei välttämättä enää tarvita. Osittamista saatetaan myös tarvita, sillä monet olemassa olevat tiedonlouhinnan työkalut toimivat vain datan ollessa relationaalisessa muodossa. Relationaalisen datan kyselyjen tehokkuus voi myös olla parempi kuin vastaavan XML:n jos kaavio on tarpeeksi yksinkertainen. [Nicola and van der Linden, 2005]

SQL:ää ja XQueryä voi käyttää erillään toisistaan. Tietokantasovellukset voivat kuitenkin hyötyä merkittävästi näiden kielten integroinnista. Koska monet sovellukset käsittelevät relationaalista ja XML-dataa samanaikaisesti, tarvitsee kyselyjen silloin yhdistää nämä kaksi datatyyppiä. [Nicola and van der Linden, 2005] XQueryn ja SQL:n semanttisen

erilaisuuden takia kaikkia XQuery-lauseita ei pysty kääntämään SQL:ksi, tai ne kääntyvät huonoiksi SQL-lauseiksi [Beyer et al., 2005].

Esimerkki kääntämisestä, kun kaikkea ei tallenneta tauluihin:

```
.xsd:
.
.

<xs:element name="koira">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="koiraid" type="xs:string"/>
   <xs:element name="nimi" type="xs:string"/>
   <xs:element name="rotu" type="xs:string"/>
   <xs:sequence>
    <xs:element name="henkiloid" type="xs:string"/>
   </xs:sequence>
   </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name="henkilo">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="henkiloid" type="xs:string"/>
   <xs:element name="nimi" type="xs:string"/>
   <xs:element name="puhnro" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name="koirarotu">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="maa" type="xs:string"/>
   <xs:element name="karvanpituus" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>

</xs:schema>
```

Koira-tauluksi

Omistaja-tauluksi

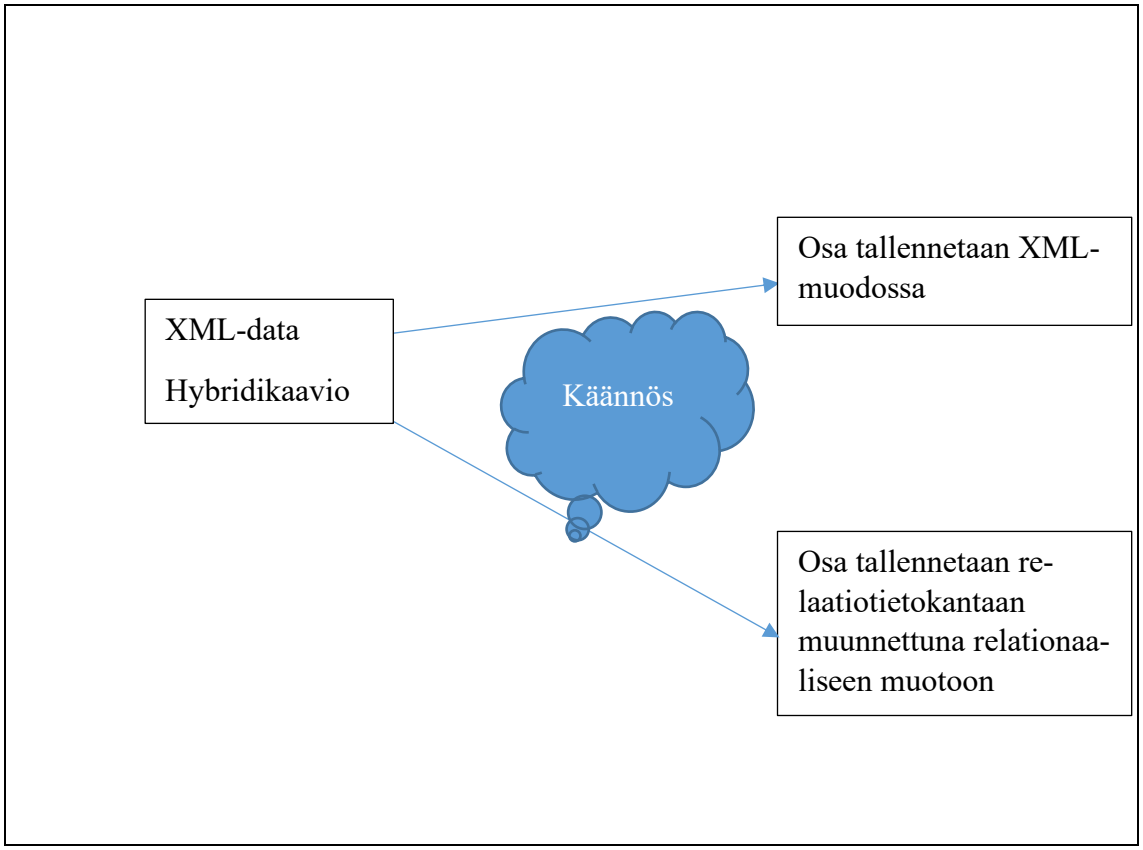
Henkilö-tauluksi

Ei tauluksi, koska näistä tiedoista ei kiinnostuttu

3.1 Hybridi relaatio-XML-tietokanta

Monet yritykset ovat siirtyneet XML:ään, sen paremman joustavuuden ja helpomman ylläpidon takia. Toisaalta on paljon dataa, joka pysyy lähes muuttumattomana ja siten sopii hyvin relationaaliseen malliin. On selvää, ettei relationaalinen eikä XML-datan hallintajärjestelmä yksinään riitä, vaan tarvitaan järjestelmä, joka pystyy käsittelemään kumpaakin tietotyyppiä. [Moro et al., 2007]

Hybridijärjestelmillä voidaan hallita sekä relationaalista että XML-dataa. Hybridikaavion suunnittelussa on tärkeää päättää mikä osa datasta talletetaan relationaalisessa ja mikä XML-muodossa [Moro et al., 2007]. Kuvassa 5 on esitetty hybridijärjestelmä. Vaikka XML-data voitaisiin muuttaa relationaaliseen muotoon, voi olla, että se ei muuten ole soveliasta. Syntyy tarve XML:n hakemiseen relationaalisen datan rinnalla. Relationaalinen ja XML-data täydentävät toisiaan. [Beyer et al., 2005]



Kuva 5. Hybridijärjestelmä.

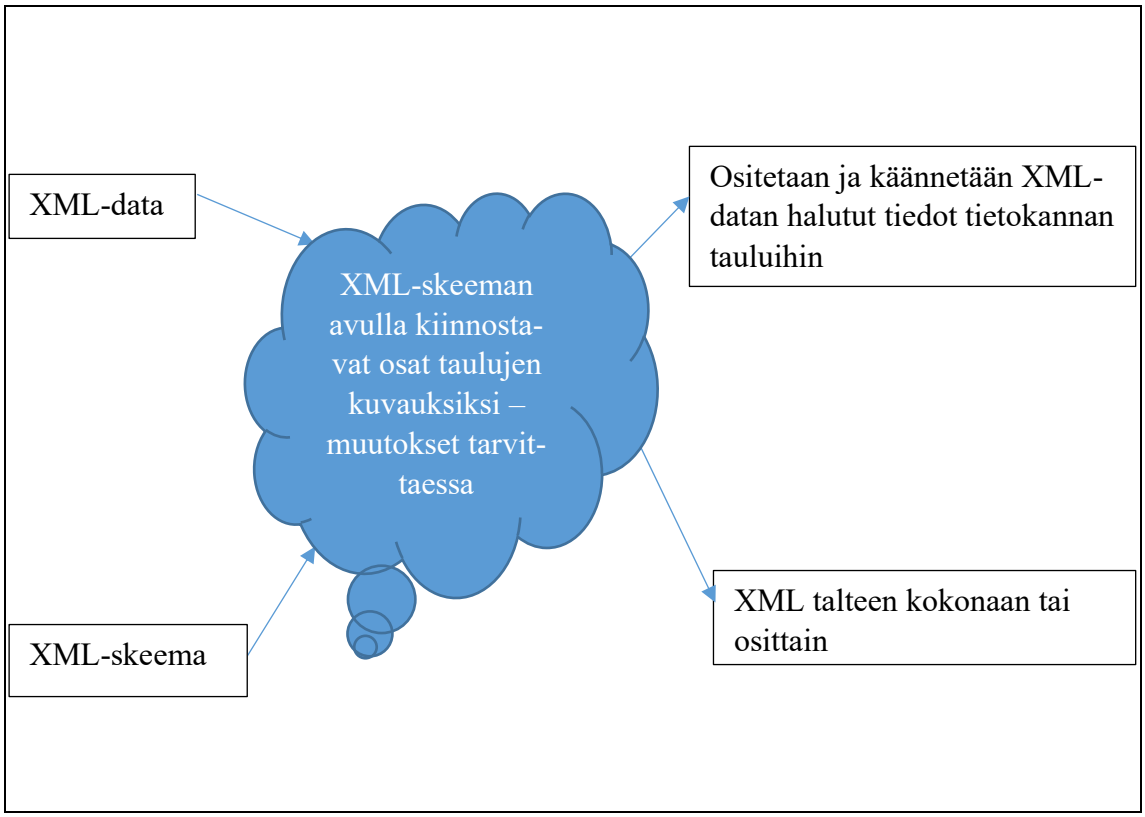
Suuret relaatiotietokantajärjestelmät ovat tukeneet XML:ää useita vuosia, erityisesti kääntämällä XML LOB:iin tai (objekti-)relaatiotauluihin. Näillä on kuitenkin rajoituksensa. Ratkaisuna voi olla XML-dokumenttien tallennus puumuodossa, vastaten XML-tietomallia. Tällä vältetään kääntäminen XML:n ja relaatiorakenteiden välillä, sekä vastaavat rajoitukset. [Nicola and van der Linden, 2005]

Hybridijärjestelmiä ovat esimerkiksi System RX ja IBM DB2 9. System RX on hybridi relationaalisen ja XML-datan hallintajärjestelmä. Se tukee sekä relationaalista että XML-dataa. System RX prosessoi tehokkaasti suuria määriä XML-dataa. [Beyer et al., 2005] IBM DB2 9:ssä XML-dokumentit tallennetaan puumuodossa [Moro et al., 2007]. DB2 on täydennetty tukemaan XML-indeksejä, XQueryä, SQL/XML:ää, sekä XML-skeemaa. Tämä pitää XML:n ja relationaalisen datan hallinnan tasavertaisessa asemassa. [Nicola

and van der Linden, 2005] DB2:n kyselyprosessori käsittelee sekä SQL:ää että XQueryä samassa kehyksessä (framework), ilman että XQuery käännetään SQL:ksi. Tauluun vietyyn dokumenttiin voi olla liitetty kaavio. Samassa sarakkeessa voi olla monta XML-dokumenttia, joihin on yhdistetty erilaisia kaavioita. [Moro et al., 2007]

3.2 XML-tuki relaatiotietokannoissa

XML-tuesta on tullut perusominaisuus kaikissa suurissa kaupallisissa tietokannan hallintajärjestelmissä. Eräs avainongelmista on minkä osan datasta tallettaa relationaalisessa muodossa, ja minkä osan XML-muodossa. Katso mallia kuvasta 6. Muita haasteita ovat miten yhdistää XML-data relaatiotauluihin ja miten säilyttää suhteiden rajoitteet, kun määritellään kaaviota hierarkioille. [Moro et al., 2007]



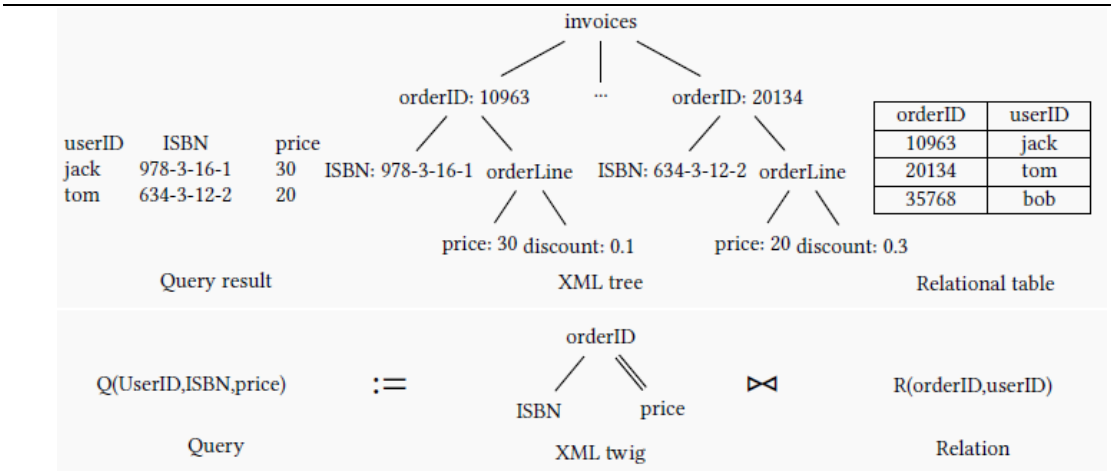
Kuva 6. Kiinnostava osa datasta talletetaan tietokantaan tauluiksi.

Yksi ratkaisu XML-tukeen relaatiotietokannassa on XML:n generointi joukosta tauluja XSD:iin (XML schema definition) perustuen, ja XML-ilmentymien osittaminen kyseisiksi tauluiksi. [Pal et al., 2004]

Kun XML on kerran ositettu tauluiksi, voidaan relaation koko voimaa käyttää datan hallintaan ja kyselyyn. Osittaminen sopii XML-datalle, jolla on hyvin määritelty rakenne. Se riippuu XML-datan kuvaavan kaavion sekä relationaalisen ja XML-muotojen välisen XML-datan käännöksen olemassaolosta. [Pal et al., 2004]

Kun XML:n käyttäminen lisääntyy sekä datakeskeisissä että dokumenttikeskeisissä sovelluksissa, on XML-tuen lisäämisestä relaatiotietokantoihin hyötyä. Se tarjoaa kypsemmän alustan XML-tietomallille ja toimii yhteentoimivuuden perustana relationaalisen ja XML-datan välillä. XML-data ositettuna yhteen tai useampaan relaatiotauluun tukee rajallisesti XML-tietomallia. XML-data voidaan säilyttää BLOB-tietotyyppinä taulun sarakkeissa, jotta tuetaan XML-tietomallia paremmin. Tästä tulee uusia haasteita kyselyn prosessointiin, kuten indeksittömän XML:n BLOB:in indeksointi hyvän kyselytehokkuuden aikaansaamiseksi. [Pal et al., 2004]

Chen [2018] käsittelee relaatiotietokannan ja XML-tietokannan liittämistä. Vaatimukset kasvavat, kun käsitellään isoa määrää eri tyyppistä dataa. Data voi olla eri tyyppistä ja muotoista: rakenteellista, puolirakenteellista ja rakenteetonta. Eri tietomallien käsittelyyn kehitetään monimuotoista tietokantaa. Tässä tutkitaan relaatiokannan ja XML-tietokannan liittämistä. XML-oksat (twig) muutetaan relationaalisiksi rakenteiksi niin että voidaan laskea yhdistelmän koko pahimmassa tapauksessa. Kuvassa 7 on esitetty esimerkki relationaalisen ja XML-datan liittämisestä.



Kuva 7. Esimerkki relationaalisen ja XML-datan liittämisestä [Chen, 2018].

Optimaalisen algoritmin saavuttamiseksi yhteiset arvot ja relaatiot laajennetaan kaikista tietokannoista samaan aikaan. XML:n oksien vanhempi-lapsi-suhteet käsitellään relaatiotauluina kokoa varten, mutta niitä ei fyysisesti siirretä relaatiotauluiksi. Välitulokset ovat ratkaisu jo laajennettujen attribuuttien osajoukolle ja kuvaavat pahimman tapauksen optimaalista algoritmia. Pahimman tapauksen algoritmia aiotaan parantaa suodattamalla toteuttamiskelvottomia välituloksia ja osittain validoimalla oksarakennetta liittämisen aikana. [Chen, 2018]

3.3 Middleware (XML- ja relaatiotietokannan yläpuolelle)

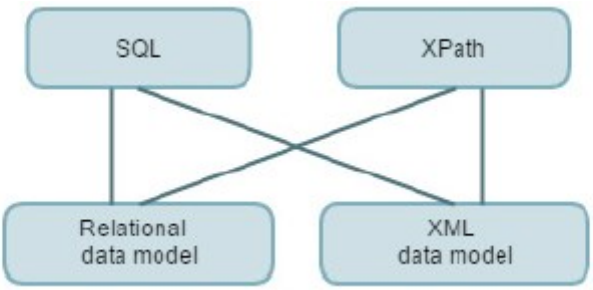
On monia tutkimus- sekä teollisuusaloja (digitaaliset kirjastot, tiedonhallintajärjestelmät yms.) missä ohjelmistojen täytyy päästä käsiksi sekä XML- että relaatiotietokantoihin. Middleware on ohjelmisto tietokantojen ja ohjelmiston välillä. Se tarjoaa yhteneväisen pääsyn sekä relationaaliseen että XML-dataan. Tarkoitus on, että integroitua dataa voidaan hakea sekä XQueryllä että SQL:llä, riippumatta siitä onko tietolähde XML vai relationaalisessa muodossa. [Kozlova et al., 2006]

Relationaalisen ja XML-datan tehokas integrointi voidaan saavuttaa middleware kerroksella, joka antaa käyttäjille integroidun, kirjoitussuojatun, ja ajonaikaisen pääsyn sekä relationaaliseen että XML-dataan monesta eri tietolähteestä. [Kozlova et al., 2006]

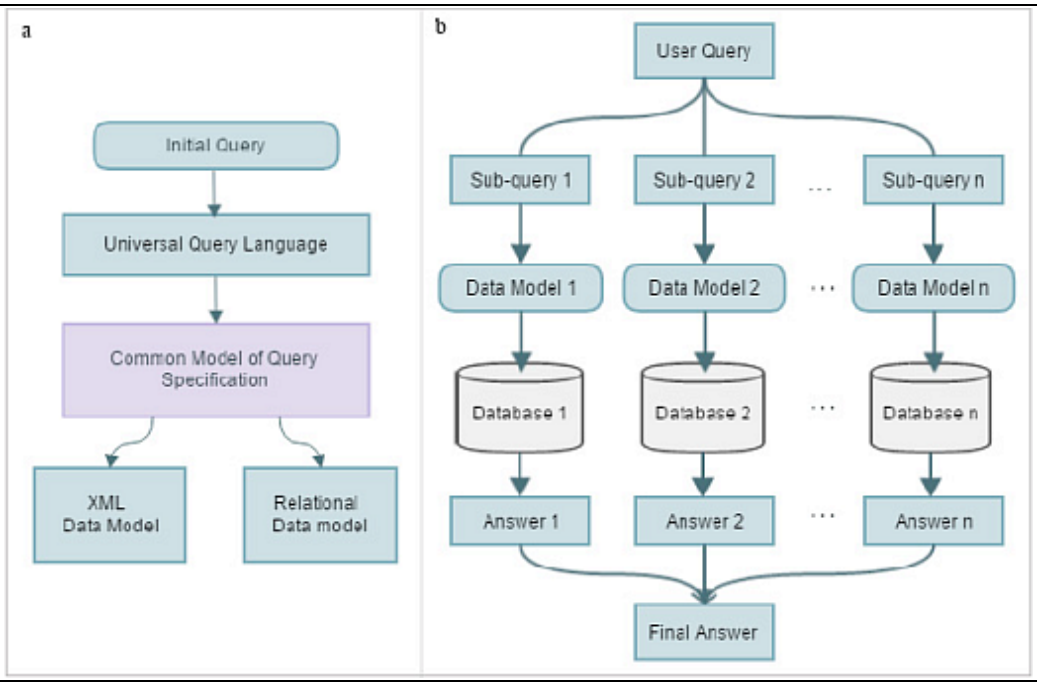
Nassiri et al. [2017] käsittelee relationaalisen ja XML-tietomallien integrointia. Relaatiomalli on vuosia ollut käytetyin malli tiedon käsittelyssä. XML on nopeasti tulossa suositummaksi tiedon vaihdon vakiomuodoksi. Jonkinlainen yhteyssilta näiden mallien välille tarvitaan. Nassiri et al. [2017] määrittelevät miten tieto otetaan talteen mallista riippumatta ja yksi kysely riittää tiedon saamiseen eri malleista, jotka ovat tässä tapauksessa XML ja relationaalinen. Käyttäjien ei tarvitse tuntea monia kyselykieliä kerralla, vaan he voivat kysellä millä tahansa kyselykielellä molemmilla malleilla esitettyä dataa, jopa silloin kun kieli ei ole kyseisen mallin kyselykieli.

Dataa kysellään eri suunnista: relationaalinen XML:nä ja XML relationaalisena, käännöstyökalulla, joka muuttaa XML-kyselyt SQL:ksi ja päinvastoin. Kuvan 8 esimerkissä haetaan dataa sekä SQL:llä että XPath:lla. [Nassiri et al., 2017]

Päämäärä on, että yksi kysely hakee dataa XML:stä ja relationaalisesta tietomallista:



Kuva 8. Yksi kysely hakee dataa XML:stä ja relationaalisesta tietokannasta [Nassiri et al., 2017].



Kuva 9. Kaksivaiheinen lähestymistapa a) Järjestelmän looginen rakenne b) Datan haku heterogeenisillä malleilla [Nassiri et al., 2017].

Kuvan 9 ensimmäisessä vaiheessa generoidaan Universal Query Language (UQL), jossa syötteenä on käyttäjän kysely. Toisessa vaiheessa viedään kysely läpi Common Model of Query Spesification (CMQS):in avulla. Alkuperäinen kysely voi olla joko SQL tai XPath. [Nassiri et al., 2017]

3.4 Käyttöoikeus

Alwehaibi ja Atay [2017] ehdottavat sääntöpohjaista pääsyn valvontamallia (access control) relationaalisille XML-tietokannoille. Ehdotetussa käyttöoikeusmallissa XML-lupasäännöt muunnetaan relaatiotauluun talletetuiksi. Suojauksen tarkistus suoritetaan relaatiotietokannan sisällä. Lupakonfliktit hoidetaan tehokkaasti käyttäen keskitettyä lupataulua tietokannassa. Hienojakoinen pääsyn valvontamalli sallii valtuutettujen käyttäjien päästä jokaiseen yksittäiseen sallittuun tietojoukon osaan. Valvontamalli estää valtuutettujen käyttäjien pääsyn mihinkään tietojoukon osaan, joka ei ole sallittu. Kontrolli toimii taulu- ja saraketasolla.

XML-lupadokumentti käännetään relationaaliseksi lupatauluksi, jota käytetään lupien kyselyssä. Hienojakoisessa pääsyn valvontamallissa määritellään myös *absoluuttisen ristiriidan* ja *osittaisen ristiriidan* käsitteet ristiriitaisten valtuuskäytäntöjen käsittelyyn. [Alwehaibi and Atay, 2017]

Lupakonflikteja tulee eri tapauksissa. Jos valtuuskäytäntöä ei tehdä keskitetysti, saadaan firman eri osastoilta tarpeettomia tai ristiriitaisia sääntöjä. Vaikka valtuuskäytäntö olisi

keskitettyä, säännöt eivät ole staattisia. Olemassa olevia sääntöjä voi olla tarve muuttaa, kun liiketoimintasäännöt muuttuvat tai yritys kehittyy. Siten olemassa olevat turvallisuus-säännöt ja äskettäin ehdotetut saattavat aiheuttaa luparistiriitoja. [Alwehaibi and Atay, 2017]

Valtuuskäytäntö XML:ssä on joko ulkoista tai sisäistä. Ulkoisessa käytäntö määritellään ulkoisessa tiedostossa, joka perustuu XML-dokumenttiin, jolla on DTD tai XML-skeema. Sisäisessä lähestymistavassa määritellään XML-dokumenttiin lisätunnisteita sisällyttämään turvallisuusinformaatiota. Kirjoitus käsittelee ulkoista lähestymistapaa. Säännöt ja käyttäjäinformaatio käännetään XML:stä relaatiotauluiksi. [Alwehaibi and Atay, 2017]

Lupamalli sisältää joukon sääntöjä muodossa (*subjekti, objekti, ehto, toiminta, tyyppi, moodi*):

- *subjekti* on joukko käyttäjiä
- *objekti* viittaa subjektia koskevaan tietoryhmään
- *ehto* kuvaa valinnaista tekemistä, jota sovelletaan objektiin
- *toiminta* viittaa toimintatyyppiin (select, update, delete jne.) joka subjektilta kielletään tai subjektille sallitaan. Tämä esitys käsittelee yksinkertaisuuden vuoksi vain select-toimintoa.
- *tyyppi* osoittaa vaikuttaako sääntö vain objektiin vai myös jälkeläisiin
- *moodi* kertoo, onko toiminta sallittu vai kielletty

Yksinkertainen valtuusmalli

```
<rule>
  <subject>staff</subject>
  <object>//zip[.>48000]</object>
  <action>select</action>
  <type>L</type>
  <mode>grant</mode>
</rule>
```

XML-dokumentti käyttäjistä voi sisältää *userID, password, first* ja *last name* ja jokaisen käyttäjän roolin *role*. Käyttöoikeudet voidaan niputtaa rooliin.

Tietokannoissa tiedot voidaan suojata taulu- ja rivikohtaisesti ja näkymillä voi suojata tietoja käyttäjäryhmittäin, mutta hierarkkisen XML-datan solmutason rajoitus puuttuu. Alwehaibi ja Atayn [2017] esityksessä kontrolli toimii taulu- ja saraketasolla. Eri käyttö-

oikeusmallit saattavat vaikuttaa oikeuksiin. Tarvitaan tuki konfliktin ratkaisuun. Alwehaibi ja Atayn [2017] systeemissä on algoritmi, joka ratkaisee olosuhteista johtuvia konflikteja. Algoritmi vaatii vielä kehittelyä.

4 Relaatiotietokantaan tallennettu XML

XML-datan lisääntyessä on suuri tarve tehokkaille menetelmille XML-datan tallentamiseen ja hallintaan. Relaatiotietokannat ovat ensisijainen valinta tähän tarkoitukseen, niiden tiedonhallintakyvyn takia. [Lv and Yan, 2006] Jossain tapauksissa osa datasta voi olla relationaalisessa muodossa, esimerkiksi tuotteen nimi ja hinta, ja osa XML:nä, esimerkiksi tuotteen kuvaus. Tällöin relationaalinen malli usein laajennetaan sallimaan tietotyypit, joiden arvot ovat XML-dokumentteja. [Näppilä et al., 2011]

DTD (Document Type Definition) on yksi käytetyimmistä kaavioista XML-dokumenteille [Lv and Yan, 2006]. DTD on perinteisesti ollut useimmin käytetty menetelmä XML-dokumenttien rakenteen kuvaamiseen. DTD ei kuitenkaan ole tarpeeksi ilmaisukykyinen erittäin rakenteisen datan kuvaamiseen. XML-skeema tarjoaa paljon rikkaamman joukon rakenteita, tyyppejä ja rajoitteita datan kuvaamiseen. [Ma and Yan, 2007] DTD ja XML-skeeman rakenne ovat hyvin samankaltaisia. Molempien rakenne voidaan esittää puuna. [Lv and Yan, 2006] Yksi ero DTD:n ja XML-skeeman välillä on se, että DTD asettaa tyypit elementeille perustuen ainoastaan niiden nimiin, kun taas XML-skeema ottaa huomioon myös kontekstin, jossa elementti esiintyy. [Barbosa et al., 2005]

4.1 XML:n tallennus ja ylläpito relaatiotietokannassa

XML voidaan tallentaa relaatiotietokantaan joko kokonaisena (BLOB) XML-dokumenttina tai osittaa XML-dokumentti relaatiotauluiksi [Beyer et al., 2005]. Taulussa voi olla yksi tai useampi sarake tietotyyppiä XML, relationaalisten sarakkeiden rinnalla [Pal et al., 2004]. BLOB-perustaiset lähestymistavat käyttävät ulkoista XPath/XQuery-prosessoria. Tämän suurin etu on se, että koko XML-dokumentin pystyy hakemaan nopeasti, ja minkä tahansa XML-dokumentin voi tallentaa kaaviosta riippumatta. Yleensä kuitenkin koko dokumentti täytyy tuoda muistiin ennen prosessointia, jolloin XML-osien hakeminen dokumentista on hitaampaa. Lisäksi dokumentin osittainen päivitys ei onnistu, jolloin koko dokumentti on korvattava uudella. [Beyer et al., 2005]

Sen sijaan että koko data tallennettaisiin yksittäisenä isona XML-ilmentymänä, on relaatiotietokannassa luonnollisempaa tallentaa data tauluihin, jotka edustavat tietomallin eri entiteettejä [Pal et al., 2004]. XML-dokumenttien osittaminen riveiksi ja niiden tallentaminen särmätauluihin on laajalti omaksuttu tapa tallentaa XML-data relaatiotietokantoihin. Usein toistuvat solmujen lisäykset ja poistot saattavat kuitenkin vaatia merkittävää taulun elementtien uudelleennimeämistä, millä voi olla haitallinen vaikutus tietokannan suorituskykyyn. [Shui et al., 2005] Eräs huono puoli XML-dokumentin osittamisessa relaatiotauluiksi on se, että koko tai osadokumentin haku ei ole tehokasta. [Beyer et al.,

2005] Suurien XML-ilmentymien ajoaikainen osittaminen voi olla kallista [Pal et al., 2004].

SQL-sarakkeen tietotyypiksi voidaan antaa XML. Koska XML on vain tietotyyppi muiden joukossa, voivat taulut sisältää minkä tahansa yhdistelmän XML-sarakkeita ja relationaalisia sarakkeita. Taulun kaikki sarakkeet voivat olla tietotyyppiä XML. Sarake tyyppiä XML voi pitää sisällään XML-dokumentin jokaista riviä kohden. NULL-arvoa käytetään ilmaisemaan XML-dokumentin puuttumista. XML-skeemaa ei vaadita XML-sarakkeen määrittämiseen tai XML-datan lisäämiseen tai kyselyyn. XML-sarake voi yhtä hyvin pitää sisällään kaaviottomia dokumentteja kuin myös dokumentteja, joilla on monta erilaista tai muuttuvaa XML-skeemaa. Kaavion validointi on vapaaehtoista ja dokumenttikohtaista, mikä takaa parhaimman joustavuuden. XML-tietotyypin kokoa ei ole rajoitettu, toisin kuin esimerkiksi CLOB-tietotyypillä. [Nicola and van der Linden, 2005] XML-arvojen muoto ja vastaavuus XML-tietomalliin tarkistetaan XML-sarakkeeseen tallentamista varten. Tietokantamoottori validoi ilmentymän XML-skeeman mukaan ennen kuin se tallennetaan XML-sarakkeeseen. [Pal et al., 2004]

XML sarakkeisiin:

Sarake	LOB-sarake	XML-sarake
XML käännetty ja hajotettu relaationaaliseksi	XML CLOB max 4 GB	Kaaviollinen tai kaavioton XML Rajaton koko

Tehokkaimmat informaatiojärjestelmät perustuvat relaatiomalliin. XML-rakenteet ovat koneen ja ihmisen luettavissa. XML-dataa voidaan helposti analysoida ja vaihtaa eri järjestelmien välillä. Vaikka XML on helposti ymmärrettävissä, se on hierarkkinen malli. Nykyään järjestelmät usein tarjoavat eri metodeja XML-relaatiotietokanta muunnoksiin. Metodit eivät aina ole tehokkaita. Esitys määrittelee uusia sääntöjä XML-relaatio-käännöksiin, mikä tarjoaa tehokkaamman tiedon esityksen molemmissa malleissa ja perustuu sääntöjen ohjaamiin periaatteisiin, tekoälyyn ja tuotantojärjestelmiin. [Lvamin and Cherepovskaya, 2017]

XML:n hyviä puolia tiedon vaihdossa on:

- joustavuus, saatavuus, hierarkkisten rakenteiden tallettaminen ilman että osaa yhtään ohjelmointikieltä

- rakenteita on mahdollista standardoida DTD:n tai XML-skeeman mukaan, jolloin voidaan tehdä yhteisiä muotoja tiedon vaihtoon sekä samassa organisaatiossa että alueellisella, kansallisella ja kansainvälisellä tasolla.
- yhtenäinen tietostandardi sallii lisätyn, lähetetyn tai saadun rakenteen tarkistamisen

Tiettyä tapaa XML:n tallettamiseen relaatiokantaan ei ole. Päätös voidaan varmistaa eri metodeilla:

- XML-dokumentin minimimäärä suhteita perustuu toiminnallisten riippuvuuksien analysointiin, pää- ja viiteavaimiin
- kiinteät taulut ja muunnoksen XPath-kyselystä SQL-kyselyksi kuvaava algoritmi
- itseoppiva algoritmi XML:n muuntamisesta relationaaliseksi

Olemassa olevat metodit XML-datan tallettamiseen luokitellaan.

- talletetaan XML isoina tekstidatoina CLOB alkuperäisessä muodossaan. Tietoa ei voi tehokkaasti tutkia, valita tai päivittää, vaan dokumentti pitää ladata, muuttaa, ja ladata takaisin. Metodi on tehoton.
- relaatiotietokanta tukee XML-tietotyyppiä. Tämä ei vaadi XML-dokumenttien alustavia muunnoksia. Sallii tehokkaiden metodien käyttämisen XML-datan käsittelyssä. Datan päivitykseen käytettävät toiminnot eivät ole tehokkaita.
- XML käännetään relaatiotietokannan taulun riveiksi ja sarakkeiksi. Tämä tapa sallii tehokkaiden sisäänrakennettujen tapojen käyttämisen etsimiseen, valitsemiseen, päivittämiseen ja poistamiseen.

4.2 Relaatiotietokantaan tallennetun XML:n kysely

XML-datan manipulointi on tuettu ominaisuus SQL-standardissa. Tosin tämän lähestymistavan haittapuoli on se, että se vaatii yhteensopimattomien tupleorientoituneen (relationaalisen) sekä polkuorientoituneen (XML) prosessoinnin synkronointia. [Näppilä et al., 2011] On usein toivottavaa käyttää ja laajentaa SQL-lauseita hakemaan XML-dataa. Tietokannan käyttäjät tuntevat SQL:n mikä tekee siitä hyvän lähtöpisteen XML:n hallintaan. Luontainen lähestymistapa on laajentaa olemassa olevia SQL-sovelluksia ja jopa olemassa olevia SQL-lauseita XML-ominaisuuksilla. Kun XML on tavallinen SQL-tietotyyppi, kokonaisten dokumenttien palautus XML-sarakkeesta on yksinkertaista. [Nicola and van der Linden, 2005] XML-tietotyyppiseen sarakkeeseen tallennetut XML-arvot voivat olla kokonaisia dokumentteja tai niiden osia [Pal et al., 2004]. Haaroittuvissa polkukyselyissä kuten //a[.//b]//c, SQL-moottori ei kuitenkaan pysty hyödyntämään osakyselyjen välitulosjoukkoja ilman erillisen näkymän tai taulun luontia, tai käyttämättä

mahdollisesti rekursiivista osakyselyä. Suuri sisäkkäisiä osakyselyjä sisältävä SQL-kysely voi joskus sekoittaa kyselyn optimoijan ja johtaa huonoihin kyselysuunnitelmiin. [Shui et al., 2005]

XML-tietotyypin arvojen kyselyyn käytetään XQueryä osana SQL-lausetta. Kun kyselyn suoritus prosessoi jokaisen XML-ilmentymän ajoaikana tulee siitä kallista aina kun ilmentymän koko on suuri tai kun kysely koskee suurta määrää taulun rivejä. Indeksointimekanismi nopeuttaa indeksittömän (BLOB) XML:n kyselyä. [Pal et al., 2004]

SQL:n laajennos SQL/XML tukee sekä relationaalisen että XML-datan manipulointia relaatiotietokannan hallintajärjestelmissä. Se sisältää pienen joukon ominaisuuksia XML:n manipulointiin, kuten XML:n julkaisufunktioita, XML-tietotyypin, sekä joitain kääntämissääntöjä. Relationaalinen ja XML-data pysyy erillään ja ne prosessoidaan täten erikseen. [Näppilä et al., 2011] SQL/XML laajentaa relationaalista mallia XML-tietotyypillä, mahdollistaa XML-tietotyypin kyselyn relationaalisen datan rinnalla, sekä käännöksen relationaalisen ja XML-datan välillä [Beyer et al., 2005]. Relationaalisen datan voi julkaista XML-muodossa ja tuottaa sarakkeen tyyppiä XML, joka toimii syötteenä XQuerylle. Lisäksi SQL:n avulla voi palauttaa dokumentteja XML-varastosta. SQL/XML funktiot yhdistävät XPathin tai XQueryn SQL-lauseisiin, jolloin osadokumenttien hakeminen ja palauttaminen on mahdollista. XQuery tyypillisesti pääsee käsiksi XML-tietokantaan. XQuery voi vaihtoehtoisesti sisältää SQL-lauseita liittääkseen XML:n relationaaliseen dataan. [Nicola and van der Linden, 2005]

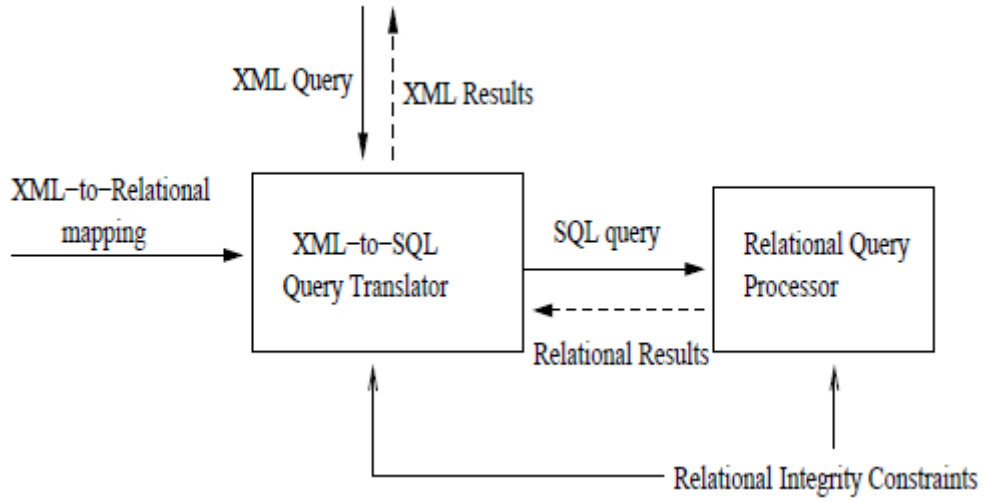
Pal et al. [2004] tutustuttaa XML-datan tukeen relaatiotietokannassa, koska XML:n käyttö kasvaa sekä tietokeskeisissä että dokumenttikeskeisissä sovelluksissa. Vaikkakin kyselyt yhteen tai useaan relaatiotauluun silputtuun XML-dataan ymmärretään, tuki XML-tietomalliin on rajoitettu. XML-kaaviomallin tukena XML-dataa voidaan jatkaa bittijonona (BLOB) taulujen sarakkeissa. Kyselyjen hyvän suorituskyvyn takaamiseksi BLOB:ja voidaan indeksoida.

Osittaminen sopii XML-dataan, jolla on hyvin määritelty rakenne. Onnistuminen riippuu XML-dataa kuvaavan kaavion ja XML-datan ja relationaalisen välisten käännösten olemassaolosta. XML-data on hierarkkinen ja sillä voi olla rekursiivinen rakenne. Relaatiotietokanta tarjoaa heikosti tukea hierarkkiselle datalle. Ne mallinnetaan viiteavaimilla. Asiakirjan järjestys on XML:n luontainen ominaisuus ja sen täytyy näkyä kyselyissä. Tämä on vastakohta relationaaliselle datalle, joka on järjestämätöntä ja jonka järjestys täytyy tehdä lisäsarakkeilla. Kyselyssä vaaditaan iso määrä liitoksia kaavion kokoami-

seen. Käsitellään taulua, jossa on relaatiosarakkeiden lisäksi yksi tai useampia XML-sarakkeita BLOB-sarakkeissa. Kyselyjen nopeuttamiseen tarvitaan indeksejä, kun käsitellään isoa järjestelmää tai paljon rivejä. Indeksointi nopeuttaa XML-BLOB:ien kyselyä. Kirjoitus käsittelee tekniikoita indeksoida XML, joka on talletettu relaatiotietokantaan hajottamattomassa muodossa. [Pal et al., 2004]

B+puu indeksiä on käytetty laajalti relaatiotietokannoissa ja se on luontainen valinta indeksittömän XML-datan indeksointiin. XML-indeksien avulla voidaan tehokkaasti kysellä XML-dataa. Kyselyn suoritus saattaa vaatia XML-tuloksen uudelleenkokoamista B+puun indeksistä säilyttäen samalla dokumenttijärjestyksen ja dokumenttirakenteen. Kyselyn prosessointia voi nopeuttaa tallentamalla XML-ilmentymien ositetun muodon B+puuhun, joka säilyttää XML-ilmentymien rakenteen taulun XML-sarakkeessa. [Pal et al., 2004]

Krishnamurthy et al. [2005] käsittelee sitä kuinka yksinkertaiset käännökset XML:stä muuttuvat monimutkaisiksi SQL-kyselyiksi. Kun käännetään XML-kyselyjä SQL-kyselyiksi, liittyy siihen hierarkkisten kaavioiden kääntäminen litteiksi relaatiokaavioiksi. On kaksi tapaa käsitellä ongelmaa. Yksi tapa on generoida SQL-kyselyt käyttäen melko yksinkertaista käännösalgoritmia ja sitten yrittää optimoida kyselyä. Toinen tapa olisi yrittää käyttää älykkäämpää käännösalgoritmia toiveissa generoida tehokas SQL suoraan. Esityksessä paneudutaan jälkimmäiseen lähestymistapaan.



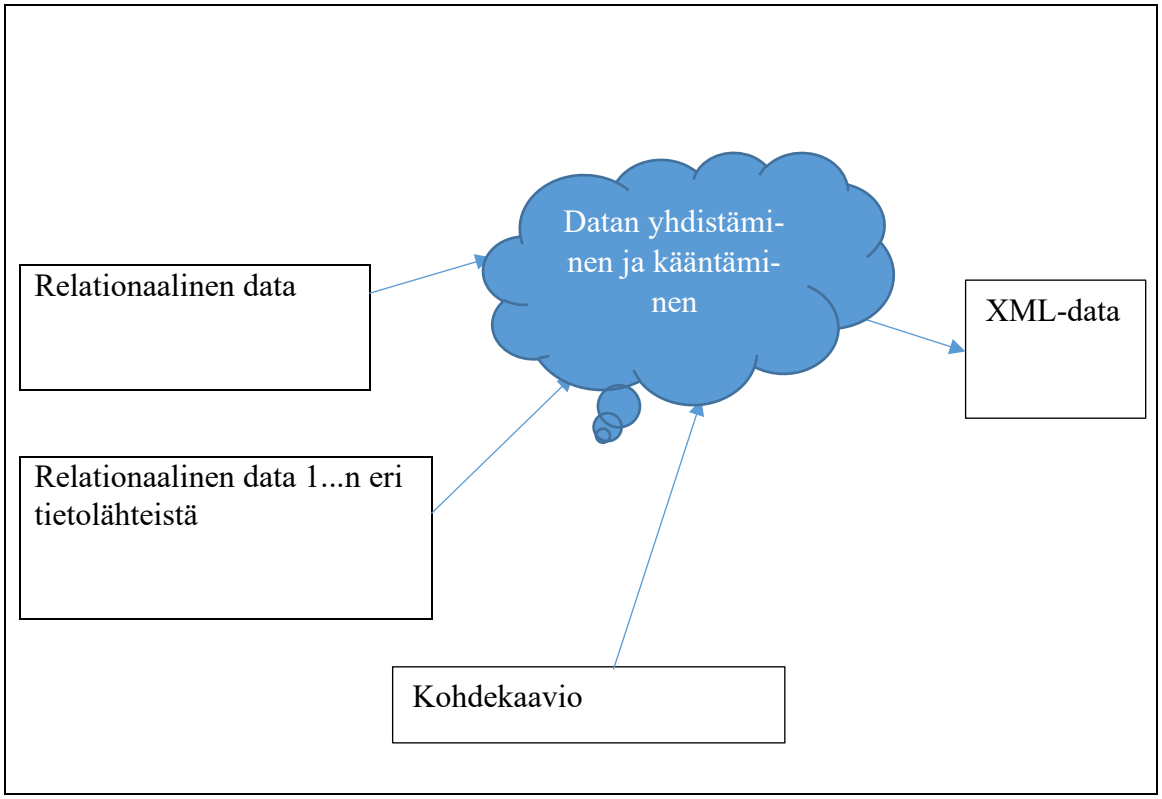
Kuva 10. Vaiheet kun käytetään RDBMS XML-kyselyn kehittämiseen [Krishnamurthy et al., 2005].

Kuvan 10 oikeanpuoleisessa laatikossa käytetään RDBMS sijaan termiä Relational Query Processor, koska ei oleteta sitä, onko XML:stä SQL:ään kyselyn kääntäjä RDBMS:ssä vai middlewaressa. [Krishnamurthy et al., 2005]

Ensimmäisenä tunnistetaan bijektiiviset käännökset (bijective mappings). Ne ovat sellaisia, jotka voidaan optimoida käyttäen eristämistä ja vastaavuutta. Toisin sanoen kohde-relationaalinen data viedään tasan kerran XML-näkymään eli on yksi-yhteen vastaavuus XML-näkymän datan ja relationaalisen datan välillä. Toiseksi löydetään kyselyjä, joita ei tarvitse minimoida. Kun kääntäminen ja eheysrajoitukset eivät riipu kyselyn optimoinnista, niitä voidaan käyttää esilaskemaan hyödyllistä informaatiota ja käyttää sitä informaatiota kyselyssä. [Krishnamurthy et al., 2005]

5 Relationaalinen data XML:ksi

Kääntäminen relationaalisesta datasta rakenteellisesti yhtäläiseen XML-dataan on suorasukaisempi prosessi kuin toisinpäin [Näppilä et al., 2011]. Joukko rajoitteita kohdekaavioissa on osa kääntämisskenaariota. Kohderajoitteita voidaan käyttää datan yhdistämissääntöjen ilmaisuun, mitkä syntyvät, kun integroidaan dataa monista tietolähteistä, jossa on päällekkäistä informaatiota. Datan yhdistäminen on tunnetusti vaikeata datan integroinnissa. Kuvassa 11 on esitetty, miten relationaalinen data muuttuu XML:ksi. Esimerkiksi tieteellisen datan integroinnissa tulee vastaan monia monimutkaisia tapauksia, jossa tarvitaan datan yhdistelyä. Käännökset voivat olla epätäydellisiä ja saattavat määritellä vain kohde-elementtien osajoukon. Kohderajoitteiden yhdistäminen takaa sen, että saman dataentiteetin eri osat, vaikka ne ovat eri tietolähteissä, yhdistetään ja esitetään käyttäjälle kokonaisena. Muunnossäännöt kuvaavat kyselyn muunnoksen askeleet, kun kyselyn ilmaisu korvataan toisella ilmaisulla. [Yu and Popa, 2004]



Kuva 11. Tauluista XML.

5.1 Kaavion muunnos

Relaatio-XML-kaavioiden luonti tietomallista on haastavaa. Samalle joukolle parametreja ja vaatimuksia voidaan käyttää montaa erilaista relaatio-XML-kaaviota. [Moro et al., 2007]

Kun on kohderajoitteita, ei riitä, että kääntämistä harkitaan vain kyselyn vastaamiseen. Kaikkien käännösten tietolähteistä ja kaikkien kohderajoitteiden välisen vuorovaikutuksen täytyy määritellä kohdeilmentymä jota kyselyn pitäisi tarkkailla. Tämä vuorovaikutus voi olla melko monimutkaista, kun kaaviot ja käännökset ovat sisäkkäisiä ja kun datan yhdistämissäännöt voivat käyttää toisiaan, mahdollisesti rekursiivisella tavalla. [Yu and Popa, 2004]

5.2 Informaation ja eheysrajoitteet säilyttäminen

Kohderajoitteet määritellään perustuen ainoastaan kohdekaavioon, jonka oletetaan olevan suunniteltu lähdekaavioista riippumatta. Itse asiassa kohderajoitteet ovat usein osa kohdekaaviota, esimerkkinä avainrajoitteet XML-skeemassa. Joukko kohderajoitteita, jotka määrittävät lisäominaisuuksia kohdekaavioon ovat erityisen tärkeitä, kun dataa integroidaan useista tietolähteistä, jossa on päällekkäistä dataa, ja kun kyseiset rajoitteet voivat kohteessa ilmaista datan yhdistämissääntöjä. [Yu and Popa, 2004] 1:1, 1:n ja m:n suhteet täytyy säilyttää ja muuntaa, jotta ei kadoteta informaatiota [Fong and Cheung, 2005].

6 XML:n muunnos relaatiotietokannan tauluiksi

Jotta relationaalisia tietokantoja voidaan käyttää XML-datan hallintaan, tarvitaan käännös, joka kertoo menetelmän, miten dokumentit ositetaan relaatiotietokantoihin, menetelmän tietokantojen julkaisuun takaisin dokumenteiksi, ja joukon rajoitteita, jotka näiden tietokantojen täytyy täyttää. Eräs tärkeä seikka XML-dokumentin osittamisessa on, vaatiiko dokumenttien tallentaminen DTD:n tai XML-skeeman. [Beyer et al., 2005] Kuten minkä tahansa muun kääntämisstrategian kanssa, on tärkeätä tutkia kääntämisen informaationsäilytysominaisuuksia, jotta ymmärretään niiden soveltuvuus tietylle sovellukselle. [Barbosa et al., 2005] Kääntämisessä informaation säilyttäminen on tärkeää. Dokumenttijärjestyksen säilyttäminen nostaa kääntämisprosessin kustannuksia. [AmerYahia et al., 2004]

Relationaalisen ja XML-tietomallin yhteensopimattomuuden takia, on tarpeellista ensin osittaa ja viedä XML-data relaatiotauluihin, ja sitten kääntää alkuperäisen datan XML-kyselyt vastaaviksi käännettyjen taulujen SQL-kyselyiksi. XML:n tallentamisessa relaatiotietokannanhallintajärjestelmiin on rajoituksensa. Järjestelmän tarjoamat tallentamisratkaisut ovat sidottuja tiettyyn tietokantaan ja käyttävät omia kääntämiskieliä, joiden oppiminen voi olla vaativaa, eivätkä ne usein pysty ilmaisemaan haluttuja käännöksiä. [Amer-Yahia et al., 2004] Tärkeä vaatimus kääntämisjärjestelmälle on tuki kaikille kääntämistehtäville kaavion suunnittelusta lähtien. Jotta XML-dokumentti voitaisiin tallentaa relaatiotietokantaan, pitää XML-dokumentin puurakenne ensin kääntää vastaavaan relaatiokaavioon. Seuraavaksi XML-dokumentit ositetaan ja viedään käännettyihin tauluihin. Lopuksi XML-kyselyt käännetään SQL:ksi, viedään relaatiotietokannanhallintajärjestelmään, ja tulokset käännetään XML:ksi. [Amer-Yahia et al., 2004]

Jotta kääntäminen on häviötöntä, jokainen dokumentti, tai sen osa, pitää voida uudelleenrakentaa tietokannasta alkuperäiseen muotoonsa. Käännöksen validoinnilla taas tarkoitetaan sitä, että jokainen laillinen instanssi tietokannassa vastaa validia dokumenttia. Häviöttömyys ja validointi ovat toisistaan riippumattomia ja toinen ei edellytä toista. Häviöttömyys on riittävä sovelluksille, jotka koskevat vain dokumenttien kyselyitä. Sekä häviöttömyyttä että validointia tarvitaan, jos dokumenttien pitää vastata DTD:tä tai XML-skeemaa ja sovellus koskee dokumenttien kyselyjä sekä päivityksiä. [Barbosa et al., 2005] XML-skeeman käyttö tekee käännösten määrittelystä helppoa. Toinen XML-skeeman käytön etu on se, että kääntämismäärittelyt voidaan validoida automaattisesti. [Amer-Yahia et al., 2004] Elementin validisuusrajoite on identtinen DTD:ssä ja XML-skeemassa. Kun kääntämisessä käytetään häviöttömyyttä ja validointia, dokumenttien kyselyt ja kaikki päivitykset voidaan tehdä käyttämällä SQL-kyselyjä. [Barbosa et al., 2005]

Käännös kaavioiden välillä on häviötön, kun se koostuu lauseista, joissa joku osa XML-datasta vastaa jotain relaatiodatan osaa. Siksi kyselyn uudelleenkirjoittaminen on vastaavuuden säilyttävä. Käännös on häviöllinen tai epätäydellinen kun joku osa lähdedatasta on jonkun kohdedatan osan osajoukko. Täten tietolähteet ja niiden käännökset antavat epätäydellisen, osittaisen, kuvan maailmasta. Kun käytetään epätäydellisiä käännöksiä, jotka ovat riippumattomia toisistaan, kattamaan eri osat kaavion muunnoksesta, voidaan järjestelmä määritellä vaiheittain. Tästä tulee tärkeämpää, kun käännöksiin lisätään kohderajoitteita. [Yu and Popa, 2004]

Kyselyn uudelleenkirjoittaminen voidaan tehdä kääntämällä XML-rakenne ensin relaatiokaavioksi, ja käyttämällä sitten näkymiä kyselyjen vastaamiseen. Käännökset itse vaaditaan määriteltävän mieluummin relaatiokaavion termeinä kuin XML-kohteen termeinä. Tämän takia kyselyt ja käännökset voivat olla melko monimutkaisia, jolloin niiden ymmärtäminen tai määrittely on vaikeata käyttäjän näkökulmasta. [Yu and Popa, 2004]

Sekä häviöttömyys että validointi voidaan taata relaatiokaavion rajoitteilla. Häviöttömässä ja validoivassa käännöksessä relaatiokaavio vastaa dokumenttikaaviota. Rajoitteet relaatiokaaviossa estää poikkeavuudet hylkäämällä päivitykset, jotka johtaisivat kelvottomiin dokumentteihin tietokannoissa. Relationaalisten rajoitteiden käytöllä validisuuden varmistamiseksi on useita etuja verrattuna vaihtoehtoiseen tapaan, jossa dokumentin päivitetty osa validoidaan uudelleen. Vaihtoehto informaation säilyttävälle käännökselle on validoida joka päivityksestä saatu dokumentti ennen muutosten tekemistä. Vaikka päivityksen laillisuus voitaisiin testata dokumentin tai sen osan uudelleenrakentamisella, tekemällä päivityksen ja tarkistamalla tuloksen validisuus, on sillä kuitenkin huonot puolensa. Relaatiotietokannan ilmentymä on laillinen, jos se ei riko ainuttakaan rajoitetta relaatiokaaviossa. Validointi on laskennallisesti kallista ja osittaisen validoinnin tekniikat vaativat paljon lisäinformaatiota. Lisäksi tämä lähestymistapa vaatii erityiseen tarkoitukseen tehtyjä työkaluja, joiden toiminnallisuus menee päällekkäin tietokannanhallintajärjestelmän rajoitteiden tarkistamisen kanssa. [Barbosa et al., 2005]

Skaalautuvan datan integrointijärjestelmän suunnittelussa häviötön käännös on välttämättömyys. Yleensä jokainen käännös kaavioiden välillä on määritelty muista käännöksistä tai tietolähteistä riippumatta ja koskee vain osaa kohdekaaviota. Tämän lähestymistavan etuna on sen modulaarisuus ja skaalautuvuus, eli uudet käännökset, ja niiden mukana kohderajoitteet, voidaan helposti lisätä järjestelmään vaikuttamatta muihin käännöksiin tai rajoitteisiin. [Yu and Popa, 2004]

6.1 Kaavion muunnos

Relationaalisen ja XML-datan erilaisuuden takia on käännös niiden välillä haastavaa. Puolirakenteisen XML-datan muuntaminen rakenteiseksi relationaaliseksi dataksi tarkoittaa aina datan uudelleenjärjestelyä hierarkkisesta hierarkkittomaksi, jolloin on suuri riski, että XML-datan järjestys menetetään. [Näppilä et al., 2011]

XML-datan kääntäminen relationaaliseen malliin on hankalaa ilman että osa informaatiosta menetetään. Hierarkkinen XML-data voidaan osittaa useiksi relaatiotauluiksi, jolloin hierarkkiset suhteet säilytetään käyttämällä join-attribuutteja. Tosin näissä muunnoksissa menetetään helposti XML-datan järjestys, eli dokumenttijärjestys, jolloin alkuperäisen XML-dokumentin uudelleenrakentaminen on vaikeaa. [Näppilä et al., 2011]

On monia eri tapoja kääntää XML-dokumentit relaatiotauluiksi. Eri lähestymistavat käyttävät eri keinoja elementin identiteetin, dokumenttirakenteen ja järjestyksen säilyttämiseen. Lähestymistavat eroavat siten, mitä metadataa ne käyttävät, esimerkiksi kaaviollinen vai kaavioton, miten relationaalinen kokonaisuus luodaan, ja mitä informaatiota säilytetään relationaalisella puolella. Kääntämisstrategiat voidaan karkeasti luokitella kaaviollisiin ja kaaviottomiin. Kaavion informaatiota voidaan käyttää tehokkaiden käännösten luonnissa sekä datan fragmentoitumisen minimointiin. Kaaviottomiksi kutsutaan tekniikoita, jotka tallentavat XML-dokumentit yleisiin ennalta määriteltyihin relaatiotauluihin. [Amer-Yahia et al., 2004]

Kaaviolliset tekniikat ovat keskittyneet rakenteiden ja rajoitteiden kääntämiseen, usein jättäen huomiotta elementtien järjestyksen. Järjestyksen huomiotta jättäminen johtaa häviöllisiin käännöksiin, jolloin dokumenttien uudelleenkokoamista ei pystytä tekemään kokonaan. [AmerYahia et al., 2004] Kaaviopohjaiset lähestymistavat eivät pysty käsittelemään dynaamisia eli muuttuvia kaavioita kustannustehokkaasti ja häiriöittä, sillä jokainen muutos kaavioon vaatii tietokannan uudelleenjärjestelyä. [Beyer et al., 2005]

Tuki XML:n tallentamiselle löytyy useimmista kaupallisista relaatiotietokannanhallintajärjestelmistä. Tietty kääntämisstrategia ei todennäköisesti ole paras valinta kaikille ohjelmistoille. Ihanteellista on räätälöidä käännös perustuen ohjelmiston ominaisuuksiin, esimerkiksi sen dataan. Koska XML on laajalti käytetty datan vaihdossa, on mahdollista, että ohjelmistojen saattaa tarvita tallentaa tietty dokumentti, tai dokumentin eri näkymät, eri tietokantoihin. Eri käännösten määrittely on aikaa vievää ja voi lisätä ohjelmiston kehitys- ja ylläpitokustannuksia huomattavasti. [Amer-Yahia et al., 2004]

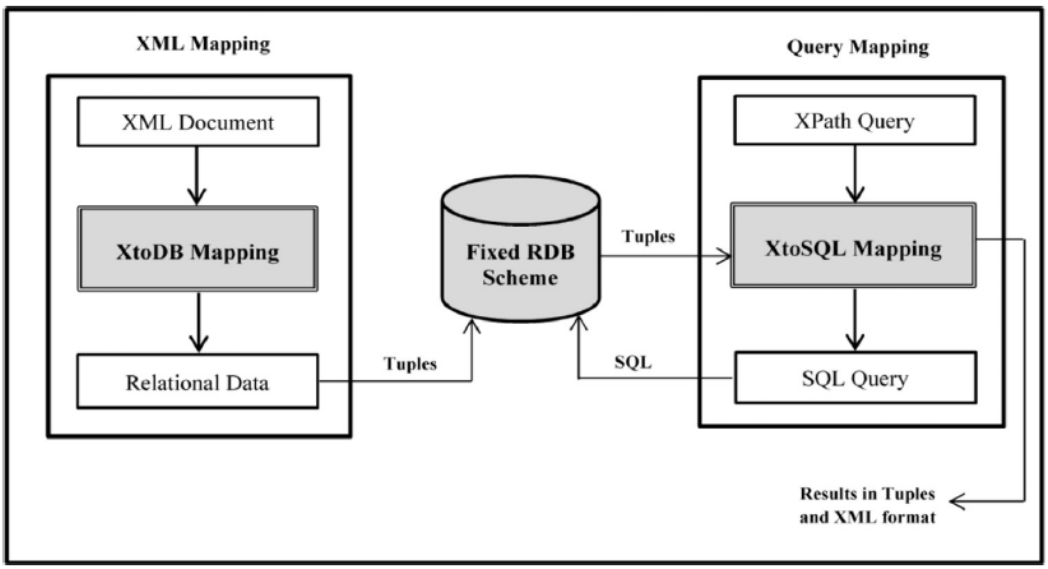
Numerointijärjestelmät, jotka säilyttävät sekä elementtien identiteetin että järjestyksen, voidaan käyttää dokumenttien rakenteen täydelliseen säilyttämiseen. Jotkut käännökset eivät välitä kaavion olemassaolosta, toiset taas hyödyntävät ja vaativat dokumenttikaavion. Jälkimmäinen lähestymistapa johtaa parempaan kyselyn suoritukseen. Kyselyjen arvioitujen ja todellisten suorituskustannusten välillä voi olla huomattavia eroja, jopa yksinkertaisissa SQL-lauseissa. [Barbosa et al., 2005]

Joukolla sääntöjä voidaan kääntää XML-skeeman osat niiden relationaalisiin vastineisiin. Voidaan esimerkiksi määritellä, että elementit, joilla on useampia esiintymiä, täytyy kääntää omiin tauluihin, kun taas elementit, joilla on vain yksi esiintymä pitäisi kääntää sarakkeeksi taulussa, joka vastaa sen vanhempi-elementtiä. Käännökset voidaan joko generoida automaattisesti tai määritellä manuaalisesti. Voi olla monia eri tapoja kääntää tietty dokumentti tai kaavio relaatioiksi, joten hyvän käännöksen aikaansaaminen vaatii kehittäjältä sekä XML että relaatioteknologioiden hyvää ymmärtämistä. Erityisesti suurien kaavioiden tapauksessa voi näiden määrittelyjen kirjoittaminen, sekä tarkistaminen että kaikki elementit ovat käännetty, viedä paljon aikaa. [Amer-Yahia et al., 2004] Relaatiokaaviolla ei ole yksittäistä vastaavaa XML-skeemaa, joten voidaan käyttää datanosittamisalgoritmeja, jotka valitsevat XML-relaatio-käännöksen perustuen johonkin kustannusmalliin [Moro et al., 2007].

Hierarkkinen XML-tietomalli ja hierarkkiton relaatiotietomalli eivät ole täysin yhteensopivia, joten niiden muuntaminen toiseksi ei ole yksinkertainen tehtävä. Epätäydellisen datan takia DTD:hen on lisätty uusia attribuutti- ja elementtityyppejä. Näitä solmuja ei kuitenkaan käytetä relaatiokaavion luomisessa, vaan ne määräävät ainoastaan luotujen relaatiotietokantojen mallin. Relaatiokaavio luodaan DTD-puusta siten että juurisolmu, sekä kaikki solmut, jotka eivät ole lehtisolmuja tai attribuutteja, muodostavat oman taulun. Taulun attribuutit muodostavat solmun lehtisolmut sekä attribuutit. Taulun avaimet saadaan DTD:n ID #REQUIRED ja IDREF #REQUIRED attribuuteista. [Ma and Yan, 2007] Jos DTD:n muuntamisessa relaatiokaavioksi ei oteta huomioon DTD:n semantiikkoja, saadut relaatiokaaviot menettävät osan alkuperäisen DTD:n tärkeästä informaatiosta [Lv and Yan, 2006].

Qtaish ja Ahmad [2016] tarjoavat uuden XAncestor-nimisen lähestymistavan kääntämiseen. Menetelmä perustuu kahteen algoritmiin. XML-käännösalgoritmi (XtoDB) ja kyselyn käännösalgoritmi (XtoSQL). XtoDB kääntää XML-dokumentteja kiinteään relaatiotietokantaan käyttämällä vähemmän talletustilaa. XtoSQL kääntää XPath-kyselyt vastaaviksi SQL-kyselyiksi perustuen esimääriteltyyn relaatiotietokantaan tarkoitukse-

naan saada kyselyaika pienemmäksi. Tarkoitus on kääntää XML-puun eri kantaisien polkuja, kaikkien XML-puun lehtisolmujen informaatio, sen sijaan että käännettäisiin koko dokumentti, sisä- ja lehtisolmut, polkuineen juuresta alkaen (solmu- ja lehtipolut). Kuvassa 12 on esitetty Xancestor-käännös.



Kuva 12. Xancestor-käännös [Qtaish and Ahmad 2016].

6.2 Informaation ja eheysrajoitusten säilyttäminen

Soveltamalla vastaavuuden säilyttäviä muunnoksia, joiden tiedetään olevan informaation säilyttäviä, saadaan käännös, jonka relaatiokaavio vastaa alkuperäistä XML-skeemaa. On useita kääntämistekniikoita tiettyjen XML-skeeman rajoitteiden, kuten avainten, vierasavainten, kardinaalisuusrajoitteiden ja ID/IDREF attribuuttien, muuntamiseen relationaaliseksi. Jotkut strategiat ovat itsenäisiä, ja uusia käännöksiä voidaan tehdä yhdistämällä niitä. [Barbosa et al., 2005]

Yksinkertainen tapa säilyttää XML-dokumentin vanhempi/lapsi suhteet, on antaa yksilöllinen tunniste jokaiselle elementille, ja lapselle vierasavain, joka osoittaa sen vanhemman tunnisteeseen. [Amer-Yahia et al., 2004]

Kun elementeille annetaan yksilöllinen tunniste, on erilaiset numeroinnit mahdollisia. Esimerkiksi Dewey tallentaa jokaisessa solmussa, polun solmusta dokumentin juureen liittämällä ketjuksi polun solmujen yksilölliset tunnisteet. Täten solmun tunniste sisältää sen vanhemman solmun tunnisteen ja dokumenttipuun tason, josta solmu löytyy. [Amer-Yahia et al., 2004]

Kun XML-dokumentit tallennetaan relaatioina, rajoitteet eivät välttämättä säily samassa muodossa, esimerkiksi XML-avaimet saattavat muuttua rajoitteiksi attribuuttien välillä relaatiokaaviossa. [Kolahi and Libkin, 2007]

7 Käsitteellinen mallintaminen tiedon integroinnissa

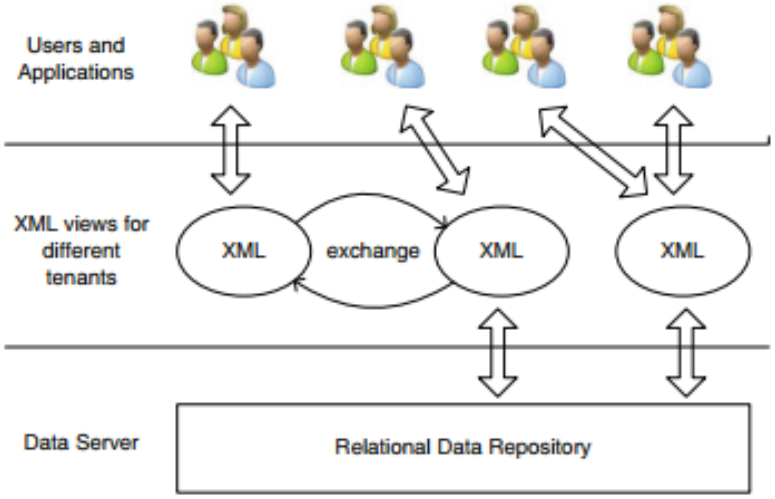
Informaatiota integroidessa eri tietokannoista, tai jos sama informaatio esiintyy usein samassa tietokannassa, voi datassa tulla vastaan ristiriitaisuuksia. Esimerkiksi saman henkilön iäksi voi yhdessä tietolähteessä olla merkitty 34, toisessa taas 37. Attribuutin arvo voi olla epätäsmällistä tai epämääräistä. Esimerkiksi henkilön ikä voi olla merkitty ikähaitarilla {18,19,20,21}, tai sanallisesti ”nuori”, tai ”noin 23”. Attribuutin arvon totuudesta voidaan olla myös epävarmoja. Esimerkiksi henkilön iäksi on merkitty 35, mutta tämä on totta 98 % varmuudella. Data voi olla myös moniselitteistä, eli sama asia voidaan tulkita monella eri tavalla. Sama datan pala voi olla epätäydellistä monellakin eri tavalla. [Ma and Yan, 2007]

XML rajoittaa attribuutit yksittäiseen yksikäsitteiseen arvoon. Tämä rajoitus ei kuitenkaan aina pidä paikkaansa. Usein jollain tietoalkioilla tiedetään olevan useita arvoja – nämä arvot saattavat olla kokonaan tuntemattomia ja ne voidaan määrittää todennäköisyysjakaumalla. Esimerkiksi henkilöllä saattaa olla useita sähköpostiosoitteita. Tapauksessa, jolloin sähköpostiosoitetta ei tiedetä tarkkaan, saatetaan eri osoitevaihtoehdoille antaa todennäköisyys. Sitä vastoin joillakin tietoalkioilla tiedetään olevan vain yksittäinen yksikäsitteinen arvo. Esimerkkinä henkilön ikä vuosina. Jos arvo on kuitenkin toistaiseksi tuntematon, voidaan käyttää ikäjakaumaa todennäköisyyksineen. [Ma and Yan, 2007]

Sumeita datajoukkoja ja todennäköisyysjakaumia on käytetty laajalti informaation epätäsmällisyyden ja epävarmuuden käsittelyyn sovelluksissa. Sovelluksissa informaatio on usein epämääräistä ja moniselitteistä. Siksi monen laatuista epätäydellistä informaatiota on laajalti tuotu tietokantoihin. [Ma and Yan, 2007] Relationaalisissa tauluissa tyhjä attribuutti merkitään NULL arvolla. XML:ssä taas tyhjät arvot puuttuvat kokonaan. [Moro et al., 2007]

Kohderajoitteet voivat johtaa ristiriitoihin. Ristiriita on tapaus, jossa kahden tai useamman lähdearvon on tarkoitus olla, mutta eivät ole, samanarvoisia kohderajoitteen takia. Ristiriitojen käsittely on yksi datan integroinnin ongelma. [Yu and Popa, 2004]

Wu et al. [2015] käyttää käsitteellistä mallintamista relaatiotietokannan ja XML-näkymien suunnitteluun ja algoritmien kehittämiseen tiedon siirtämiseksi mallista toiseen.



Kuva 13. Tiedon ylläpito ja kulku relaationaalisten ja XML-kaaviomallien välillä [Wu et al., 2015].

Kuvan 13 esimerkeissä käytetään ER-mallilla suunniteltua tietokantaa, joka julkaistaan XML-näkyminä. ER-kaavio käännetään ORA-SS-kaavioksi DTD:n ja XML-skeeman sijaan. XML-kysely käännetään ORA-SS:ään eri käsitteiksi. Sitten nämä käsitteet käännetään ER-kaavioon. Kysely tulkitaan ER-kaavion kautta SQL:llä. [Wu et al., 2015]

8 Yhteenveto

Tässä työssä käsiteltiin XML:n ja relaatiokantojen kääntämistä ja yhteiskäyttöä. Tutustuttiin XML:n ja relaatiokantojen integrointiin, XML:n tallennukseen relaatiotietokantaan, XML:n kääntämiseen relaatiotietokantaan ja päinvastoin sekä käsitteelliseen mallintamiseen tiedon integroinnissa.

XML-datan tallentaminen, kysely ja päivittäminen vaatii XML:n ja tietokantojen integrointia. Relaatiotietokannat saattavat olla lupaavampi vaihtoehto kuin muut tietokantatyypit, koska ne ovat laajalti käytössä ja sisältävät valmiita tekniikoita. Tässä työssä ei ole käsitelty muita tietokantoja kuin relaatiokantaa. XML-data voidaan tallettaa relationaaliseen dataan osittamalla se tauluihin, tallentamalla LOB-sarakkeisiin, LOB-tiedostoon tai XML-sarakkeisiin (luvut 3 ja 4).

Datan yhteentoimivuuden perusmuodot voidaan jakaa kahteen osaan: datan vaihto ja datan integrointi (kohta 2.3 kuvat 3 ja 4). Datan vaihdossa dataa siirretään yhteiseksi näkymäksi, datan integroinnissa eri tietolähteet yhdistetään ja tälle datalle tehdään yhteinen näkymä.

On erilaisia tapoja integroida relationaalinen ja XML-data. Hybridijärjestelmät (kohta 3.1) pystyvät käsittelemään sekä relationaalista että XML-dataa. Siinä on tärkeää päättää mikä osa datasta talletetaan relationaalisessa ja mikä XML-muodossa. Vaikka XML-data voitaisiin muuttaa relationaaliseen muotoon, voi olla, että se ei muuten ole soveliasta. Hybridijärjestelmissä relationaalinen ja XML-data täydentävät toisiaan.

Yksi ratkaisu XML-tukeen relaatiotietokannassa on XML:n generointi joukosta tauluja XML-skeemaan perustuen ja XML-ilmentymien osittaminen kyseisiksi tauluiksi (kohta 3.2). Kun XML on kerran ositettu tauluiksi, voidaan relaation koko voimaa käyttää datan hallintaan ja kyselyyn. Osittaminen sopii XML-datalle, jolla on hyvin määritelty rakenne. Se riippuu XML-dataa kuvaavan kaavion sekä relationaalisen ja XML:n välisen käännöksen olemassaolosta. Relationaalisen ja XML-datan tehokas integrointi voidaan saavuttaa middleware-kerroksella, joka antaa käyttäjille integroidun, kirjoitussuojatun, ja ajonaikaisen pääsyn sekä XML- että relaatiodataan monesta eri tietolähteestä (kohta 3.3). Yksi kysely riittää tiedon saamiseen eri malleista, jotka ovat tässä tapauksessa XML ja relaatio. Käyttäjien ei tarvitse tuntea monia kyselykieliä kerralla, vaan he voivat kysellä millä tahansa kyselykielellä molemmilla malleilla esitettyä dataa, jopa silloin kun kieli ei ole kyseisen mallin kyselykieli.

On monia eri tapoja kääntää XML-dokumentit relaatiotauluiksi (kohta 6.1). Eri lähestymistavat käyttävät eri keinoja elementin identiteetin, dokumenttirakenteen ja järjestyksen säilyttämiseen. Kääntämisstrategiat voidaan karkeasti luokitella kaaviollisiin ja kaaviottomiin. Kaaviottomiksi kutsutaan tekniikoita, jotka tallentavat XML-dokumentit yleisiin ennalta määriteltyihin relaatiotauluihin. Kaaviolliset tekniikat ovat keskittyneet rakenteiden ja rajoitteiden kääntämiseen, usein jättäen huomiotta elementtien järjestyksen. Tuki XML:n tallentamiselle löytyy useimmista kaupallisista relaatiotietokannanhallintajärjestelmistä. Tietty kääntämisstrategia ei todennäköisesti ole paras valinta kaikille ohjelmistoille. Ihanteellista on räätälöidä käännös perustuen ohjelmiston ominaisuuksiin, esimerkiksi sen dataan. Koska XML on laajalti käytetty datan vaihdossa, on mahdollista, että ohjelmistojen tarvitsee tallentaa tietty dokumentti, tai dokumentin eri näkymät, eri tietokantoihin.

Viiteluettelo

[Alwehaibi and Atay, 2017] A. Alwehaibi and M. Atay. A Rule-based Relational XML Access Control Model in the Presence of Authorization Conflicts. In: *Proc. of 14th International Conference on Information Technology - New Generations*, 311-319.

[Amer-Yahia et al., 2004] S. Amer-Yahia, F. Du, and J. Freire. A comprehensive solution to the XML-to-relational mapping problem. In: *Proc. of 6th annual ACM international workshop on Web information and data management*, 31-38.

[Barbosa et al., 2005] D. Barbosa, J. Freire, and A.O. Mendelzon. Designing information-preserving mapping schemes for XML. In: *Proc. of 31st international conference on Very large data bases*, 109-120.

[Bernstein and Haas, 2008] P.A. Bernstein and L.M. Haas. A guide to the tools and core technologies for merging information from disparate sources. *Communications of the ACM* 51, 9 (Sep. 2008), 72-79.

[Beyer et al., 2005] K. Beyer, R.J. Cochrane, V. Josifovski, J. Kleewein, G. Lapis, G. Lohman, B. Lyle, F. Özcan, H. Pirahesh, N. Seemann, T. Truong, B. van der Linden, B. Vickery, and C. Zhang. System RX: one part relational, one part XML. In: *Proc. of 2005 ACM SIGMOD international conference on Management of data*, 347-358.

[Chen, 2018] Y. Chen. Worst case optimal joins on relational and XML data. In: *Proc. of 2018 SIGMOD international conference on Management of data, 1833-1835*.

[Fong and Cheung, 2005] J. Fong and S.K. Cheung. Translating relational schema into XML schema definition with data semantic preservation and XSD graph. *Information and Software Technology* 47, 7 (May. 2005), 437-462.

[Heikniemi, 2001] J. Heikniemi. Mikä on XML?. *http://www.heikniemi.fi/kirj/moxml.html*

[Kolahi and Libkin, 2007] S. Kolahi and L. Libkin. XML design for relational storage. In: *Proc. of 16th international conference on World Wide Web*, 1083-1092.

[Kozlova et al., 2007] I. Kozlova, N. Ritter, and M. Husemann. Providing semantically equivalent, complete views for multilingual access to integrated data. In: *Proc. of*

ER '07 Tutorials, posters, panels and industrial contributions at the 26th international conference on Conceptual modeling, 191-196.

[Kozlova et al., 2006] I. Kozlova, N. Ritter, and O. Reimer. Towards integrated query processing for object-relational and XML data sources. In: *Proc. of 10th International Database Engineering and Applications Symposium, 2006. IDEAS '06*, 295-300.

[Krishnamurthy et al., 2005] R. Krishnamurthy, R. Kaushik and J.F. Naughton. Efficient XML-to-SQL query translation: where to add the intelligence?. In: *Proc. of the 30th VLDB Conference*, Toronto, Canada, 2004, 144-155.

[Laine, 2005] H. Laine. Relaatiotietokannan peruskäsitteet. *https://www.cs.helsinki.fi/u/laine/tkp/relaatiomalli/rakenne.html*

[Lenzerini, 2002] M. Lenzerini. Data integration: a theoretical perspective. In: *Proc. of 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 233-246.

[Lv and Yan, 2006] T. Lv and P. Yan. Mapping DTDs to relational schemas with semantic constraints. *Information and Software Technology* 48, 4 (Apr. 2006), 245-252.

[Lvamin and Cherepovskaya, 2017] A.V. Lyamin and E.N. Cherepovkaya. XML-relational mapping using production rule system. In: *Proc. of 2017 Intelligent Systems Conference*, (Sep 2017), 422-429.

[Ma and Yan, 2007] Z.M. Ma and L. Yan. Fuzzy XML data modeling with the UML and relational data models. *Data & Knowledge Engineering* 63, 3 (Dec. 2007), 972-996.

[Microsoft, 2020] XML-perustietoja aloittelijoille. *https://support.office.com/fi-fi/article/xml-perustietoja-aloittelijoille-a87d234d-4c2e-4409-9cbc-45e4eb857d44*

[Moro et al., 2007] M.M. Moro, L. Lim, and Y. Chang. Schema advisor for hybrid relational-XML DBMS. In: *Proc. of 2007 ACM SIGMOD international conference on Management of data*, 959-970.

[Nassiri et al., 2017] H. Nassiri, M. Machkour, M. Hachimi. Integrating XML and relational data. *Procedia Computer Science* 110 (2017), 422-427

[Nicola and van der Linden, 2005] M. Nicola and B. van der Linden. Native XML support in DB2 universal database. In: *Proc. of 31st international conference on Very large data bases*, 1164-1174.

[Näppilä et al., 2011] T. Näppilä, K. Moilanen, and T. Niemi. A query language for selecting, harmonizing, and aggregating heterogeneous XML data. *International Journal of Web Information Systems* 7, 1 (2011), 62-99.

[Oracle, 2020] Introduction to LOBs. *https://docs.oracle.com/cd/B10501_01/appdev.920/a96591/adl01int.htm*

[Pal et al., 2005] S. Pal, I. Cseri, O. Seeliger, M. Rys, G. Schaller, W. Yu, D. Tomic, A. Baras, B. Berg, D. Churin, and E. Kogan. XQuery implementation in a relational database system. In: *Proc. of 31st international conference on Very large data bases*, 1175-1186.

[Pal et al., 2004] S. Pal, I. Cseri, O. Seeliger, G. Schaller, L. Giakoumakis, and V. Zolotov. Indexing XML data stored in a relational database. In: *Proc. of 30th international conference on Very large data bases*, 1146-1157.

[Qtaish and Ahmad 2016] A. Qtaish, K. Ahmad. XAncestor: An efficient mapping approach for storing and querying XML documents in relational database using path-based technique. *Knowledge-Based Systems* 114 (Dec. 2016), 167-192.

[Shui et al., 2005] W.M. Shui, F. Lam, D.K. Fisher, and R.K. Wong. Querying and maintaining ordered XML data using relational databases. In: *Proc. of 16th Australasian database conference*, 85-94.

[Singh, 2018] C. Singh. XML Schema – XSD (XML Schema Definition). *https://beginnersbook.com/2018/11/xml-schema/*

[W3c, 2020] *https://www.w3.org/*

[W3schools, 2020] *https://www.w3schools.com/*

[Wu et al., 2015] H. Wu, T.W. Ling, and W.S Ng. Flexible data management across XML and relational models: a semantic approach. In: *Proc. of international conference on Conceptual modeling, Lecture Notes in Computer Science* 9381 (2015), 302-316.

[Yu and Popa, 2004] C. Yu and L. Popa. Constraint-based XML query rewriting for data integration. In: *Proc. of 2004 ACM SIGMOD international conference on Management of data*, 371-382.