RDF-muotoisen tiedon hallinta

Topi Sarkkinen

Pro gradu -tutkielma

Helsinki 7.4.2016

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta – Fakultet –Faculty	Laitos – Institution – Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä – Författare – Author

Topi Sarkkinen

Työn nimi – Arbetets titel – Title

RDF-muotoisen tiedon hallinta

Oppiaine – Läroämne – Subject

Tietojenkäsittelytiede

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
Pro gradu -tutkielma	7.4.2016	66 sivua

Tiivistelmä – Referat – Abstract

Klassinen web muodostuu linkitetyistä dokumenteista. Sen rinnalle on muodostunut linkitetyistä tietojoukoista koostuva semanttinen web, jonka tarkoituksena on tehdä verkon sisällöstä koneellisesti ymmärrettävää semanttisen metatiedon avulla. Verkon yhteenlinkittyneitä tietojoukkoja kutsutaan myös linkitetyksi dataksi.

RDF (*Resource Description Framework*) muodostaa pohjan semanttiselle webille ja linkitetylle datalle. RDF on formaattiriippumaton tietomalli semanttisen metatiedon liittämiseksi web-resursseihin. Sen tietomallissa resurssien ominaisuuksia ja yhteyksiä mallinnetaan subjekti-predikaatti-objekti -kolmikoilla. Voimakkaamman päättelyn ja sovellusten yhteentoimivuuden mahdollistamiseksi RDF-dataa voi formaalisti kuvailla ja luokitella ontologioiden avulla. OWL (*Web Ontology Language*) on hallitseva kieliperhe web-ontologioiden määrittelyyn. Lisäksi RDF-muotoiseen tietoon voi tehdä kyselyjä SPARQL-kyselykielellä (*SPARQL Protocol and RDF Query Language*).

Tutkielmassa perehdytään RDF-muotoisen tiedon hallintaan ja siihen liittyviin teknologioihin. Erityisesti selvitetään hajallaan olevan RDF-muotoisen tiedon hallintaan liittyviä erityispiirteitä, kuten ontologioiden yhteensovittamista ja useisiin lähteisiin tehtyjä kyselyitä. Suoritamme myös kaksi koekyselyä hajallaan olevien lähteiden kyselyn havainnollistamiseksi.

ACM Computing Classification System (CCS):
Information systems ~ Semantic web description languages
Computing methodologies ~ Knowledge representation and reasoning

Avainsanat – Nyckelord – Keywords
semanttinen web, linkitetty data, rdf, sparql, owl

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

Sisältö

1 Johdanto

Perinteisen linkitetyistä dokumenteista rakentuvan webin rinnalle on kehittynyt linkitetystä semanttisesta datasta koostuva semanttinen web. Sen tavoitteena on tietoverkko, jossa tiedolla on eksakti merkitys ja jossa tietokoneet voivat prosessoida ja integroida verkon tietoa tehokkaasti [BHL01]. Alan pioneereina ovat toimineet muun muassa Tim Berners-Lee, James Hendler ja Ora Lassila, jotka vuonna 2001 esittelivät artikkelissaan [BHL01] visionsa semanttisesta webistä. Käytännön tasolla semanttista webiä toteutetaan W3C:n (*World Wide Web Consortium*) määrittelemien standardien pohjalta [SHB06]. Tieteellisen mielenkiinnon lisäksi semanttinen web on herättänyt myös huomattavaa kaupallista kiinnostusta [LaH07].

Tärkein semanttisen webin pohjan muodostavista teknologioista on RDF (*Resource Description Framework)* [Hen09]. Se on erityisesti web-maailmaan tarkoitettu W3C:n standardoima tietomalli metadatan välittämiseen erilaisista resursseista. Sen ensimmäisen versio standardoitiin vuonna 1999 ja versio 1.1 julkaistiin vuonna 2014.

Perinteinen web koostuu ihmisten luettavaksi tarkoitetuista dokumenteista. Tietokoneen näkökulmasta ne ovat vain kokoelma merkkijonoja, jotka eivät sisällä sen ymmärtämää semantiikkaa, jolloin tietokone ei pysty ymmärtämään sivuja ja niillä olevaa tietoa sen syvällisemmin. Tämän vuoksi webiä käyttävien älykkäiden sovellusten rakentaminen ja webissä olevan tiedon automatisoitu käsittely on hankalaa [Las98]. RDF pyrkii ratkaisemaan ongelman lisäämällä sivujen ja niillä olevien resurssien semantiikka metadatan avulla, jolloin webistä ja sen sisältämästä tiedosta tulee paremmin tietokoneiden käsiteltävä pelkän ihmisten luettavan dokumenttiverkon sijaan [DMM00]. RDF-standardia käsitellään tarkemmin tutkielman kappaleessa 2.

RDF-muotoisen tiedon hallintaan liittyy RDF:n itsensä lisäksi myös muita teknologioita. RDF-muotoisen tiedon hakemisessa käytetään SPARQL-kyselykieltä (*SPARQL Protocol and RDF Query Language*) [HaS13]. SPARQL on SQL-tyylinen W3C:n standardoima semanttinen kyselykieli, jolla voidaan tiedon hakemisen lisäksi myös versiosta 1.1 lähtien muokata tietoa RDF-muotoisista tietokokoelmista. SPARQL-spesifikaatio sisältää myös protokollan kyselyiden lähettämiseen HTTP:n yli [FWC13]. SPARQL-kieltä ja RDF-muotoisen tiedon kyselyä käsitellään kappaleessa 3.

Semanttisen tiedon käsitteiden formaaliin määrittelyyn ja käsitteiden välisten suhteiden kuvaamiseen käytetään ontologioita [GrM12]. Niiden avulla älykäs sovellus voi tehdä tiedosta paremmin tarkkoja päättelyjä ja tulkintoja. Nykyään tärkein web-ontologiastandardi on OWL (*Web Ontology Language*), joka on W3C:n standardoima kieliperhe ontologioiden mallintamiseen [DeS04]. OWL pohjautuu aikaisempiin ontologiakieliin ja on suunniteltu nimensä mukaisesti täyttämään juuri webin ontologiatarpeet [HPV03]. OWL on laajennos RDF-tietomalliin ja OWL-ontologia on itsessään RDF-muotoista tietoa [DeS04]. OWL mahdollistaa laajemman semanttisen kuvailun kuin pelkkä RDF, minkä vuoksi OWL-kieltä tarvitaan sovellusten yhteentoimivuuden ja monimutkaisen päättelylogiikan rakentamiseksi RDF-tiedon päälle. Ontologioita ja OWL-standardia käsitellään kappaleessa 4.

Julkisesti saatavilla olevan RDF-muotoisen tiedon määrä kasvaa jatkuvasti [SHB06]. Yksi tunnetuin avoimen semanttisen datan lähde on DBpedia, jonka päämääränä on kerätä tietoa Wikipediasta ja mahdollistaa sen semanttinen kysely julkaisemalla tieto jäsenneltynä RDF-datana [BLK09]. Myös useat julkiset tahot, kuten valtiot ja kaupungit, julkaisevat avointa dataa, josta osa on RDF-muodossa [FIDAT, UKDAT, USDAT, PKDAT]. Avoimia tietolähteitä käsitellään kappaleessa 5 ja tietolähteiden käyttämiä sanastoja kappaleessa 6.

Semanttisen datan tietolähteet ovat usein linkittyneet toisiinsa. Näistä yhteyksistä muodostuu linkitetty data (*linked data*) [BBH09]. Semanttinen web ja linkitetty data mielletään joskus samaksi asiaksi, mutta yksi tulkinta niiden erolle on, että semanttinen web tarkoittaa laajempaa visiota älykkäästä webistä, kun taas linkitetty data on yksi keino toteuttaa se [BBH09]. Myös tässä tutkielmassa semanttisella webillä tarkoitetaan abstraktimpaa visiota aiheesta ja linkitetyllä datalla konkreettista keinoa toteuttaa visiota linkittämällä eri lähteiden semanttista tietoa yhteen standardien ja käytänteiden mukaisesti. Linkitettyä dataa käsitellään avoimien tietolähteiden kanssa kappaleessa 5.

Linkitetyn datan kasvava määrä ja hajanainen luonne tuottavat haasteita RDF-muotoisen tiedon hallintaan [FCO12]. Koska ei ole olemassa yhtä keskitettyä kaikenkattavaa ontologiaa, niin kaikki verkon eri lähteistä haettu data ei sovi saumattomasti yhteen. Tiedon yhdistämiseen tarvitaan tietolähteiden välille yhteisiä ontologioita tai ontologioiden yhteensovittamista (*ontology mapping, ontology alignment*), jossa eri ontologioiden käsitteiden välille määritellään suhteita [BMM08].

Ontologioiden yhteensovittamista käsitellään kappaleessa 7.1.

Myös hajallaan olevien tietolähteiden kysely itsessään on haaste RDF-muotoisen tiedon hallinnassa. Ongelmaa on yritetty ratkaista SPARQL-kyselykielen versiossa 1.1, joka toi kieleen tiedonhaun useasta lähteestä [PrB13]. Hajallaan oleviin lähteisiin voidaan myös tehdä kyselyjä etsimällä tietoa datan linkitysten kautta määrittämättä tietolähteitä ennalta [HaF12]. Hajallaan olevan tiedon kyselyä käsitellään kappaleessa 7.2.

Tutkielma pyrkii selvittämään, miten RDF-muotoista tietoa hallitaan ja miten se onnistuu erityisesti ympäristössä, jossa tieto on jakautunut useisiin itsenäisiin hajallaan oleviin lähteisiin. Tätä varten teemme myös koe- ja esimerkkikyselyt, joissa tehdään kyselyjä hajallaan oleviin tietolähteisiin ja yritetään yhdistää niistä saatu tieto. Ensimmäinen kysely on federoitu SPARQ-kysely, jolla haluamme saada tietoa lääkkeistä kahdesta eri lähteestä. Toisessa kyselyssä havainnollistamme tiedon kyselyä linkkejä seuraamalla ennalta tuntemattomista lähteistä. Kyselyt ovat tutkielman kappaleessa 7.3.

2 RDF

W3C:n kehittämä RDF luo perustan semanttiselle webille tarjoamalla yksinkertaisen mutta voimakkaan tavan välittää semanttista tietoa web-resursseista [CWL14]. RDF perustuu abstraktiin tietomalliin, eikä sillä ole tiettyä määrättyä esitys- tai serialisointiformaattia [Kle01].

RDF-metadatan avulla voidaan yksinkertaisimmillaan esimerkiksi kertoa web-sivun kirjoittaja tai kieli. Sillä voidaan myös kertoa tietojoukossa olevasta resurssista lisätietoa, esimerkiksi mainita Helsinkiä käsittelevällä sivulla sen väkiluku ja sijainti. Metadatalla voidaan siis kuvata asioiden ominaisuuksia ja yhteyksiä tietokoneelle ymmärrettävästi, mikä helpottaa esimerkiksi hakukoneiden työtä ja auttaa kehittämään webiä ymmärtäviä älykkäitä järjestelmiä [Las98].

Vuonna 2014 standardoitu versio 1.1 toi mukanaan joitain uudistuksia, mutta säilytti yhteensopivuuden edelliseen 1.0-versioon [Woo14]. Uudessa versiossa muun muassa vaaditaan, että kaikki literaalit sisältävät tiedon niiden tyypistä.

2.1 Tietomalli

RDF:n tarkoituksena on välittää metatietoa resursseista [DMM00]. Tätä varten RDF määrittelee tietomallin, joka täsmentää, miten resursseista tehdään formaaleja toteamuksia. Tietomalli perustuu lausumiin, joissa kuvataan resurssia subjekti-predikaatti-objekti -kolmikoilla (*triples*) [CWL14]. Subjekti on kuvattava resurssi ja objekti on joko jokin toinen resurssi tai vaihtoehtoisesti literaalilla kuvattava ominaisuus. Predikaatti kertoo näiden kahden välisen suhteen tyypin. Esimerkiksi toteamus ”Helsinki sijaitsee Suomessa” voidaan ilmaista kolmikkona: subjekti (”Helsinki”), predikaatti (”sijaitsee”) ja objekti (”Suomessa”).

2.1.1 Tietotyypit

RDF-tietomallissa tärkein tietotyyppi on resurssi, jolla tarkoitetaan globaalilla URI-tunnisteella identifioitavaa kohdetta [CWL14]. Resursseja voivat olla esimerkiksi kokonaiset web-sivut tai jotkin entiteetit, kuten kaupungit tai henkilöt. Globaalisti yksilöivän URI-tunnisteen ansiosta resurssin identiteetti on eksakti ja myös muut tahot voivat käyttää samaa tunnistetta resurssin yksilöimiseen.

Toinen tietomallin tietotyyppi on literaali [CWL14]. Literaali on jokin arvo, joka ei resursseista poiketen viittaa mihinkään yksilöityyn käsitteeseen. Literaaliin liittyy RDF-versiossa 1.1 tieto sen tyypistä, esimerkiksi merkkijono tai kokonaisluku, joka ilmaistaan tietotyyppiin viittaavalla URI-tunnisteella. Merkkijonotyyppinen literaali voi myös sisältää tiedon merkkijonon kielestä. Pelkät merkkijonot eivät sisällä tietokoneen ymmärtämää semantiikkaa, joten niitä voidaan käyttää vain ominaisuuksien arvoina eli objekteina [DMM00].

RDF-tietomalli sallii myös tyhjän solmun (*blank node*) käytön RDF-lauseen tietyissä osissa [CWL14]. Ne eivät viittaa mihinkään resurssiin, eivätkä sisällä literaalien tapaan tietoa. Tyhjät solmut kertovat jonkin asian olemassaolosta, mutta asiaa ei syystä tai toisesta vain ole ilmaistu resurssilla tai literaalilla.

2.1.2 Subjekti, predikaatti ja objekti

Kolmikon subjektilla tarkoitetaan resurssia, jota ollaan kuvailemassa. Subjekti ei voi olla pelkkä literaali, vaan sen täytyy olla joko yksilöitävissä oleva resurssi tai tyhjä solmu [CWL14]. Se voi olla esimerkiksi verkkosivu, josta halutaan kertoa jotain lisätietoa.

Predikaatti kuvaa subjektin ominaisuutta tai suhdetta objektiin. RDF-standardin mukaan predikaatin tulee aina olla yksilöivällä URI-tunnisteella varustettu resurssi, koska ei olisi semantiikan kannalta järkevää, että kahden käsitteen välistä yhteyttä ei kuvattaisi koneellisesti ymmärrettävällä tavalla [DMM00].

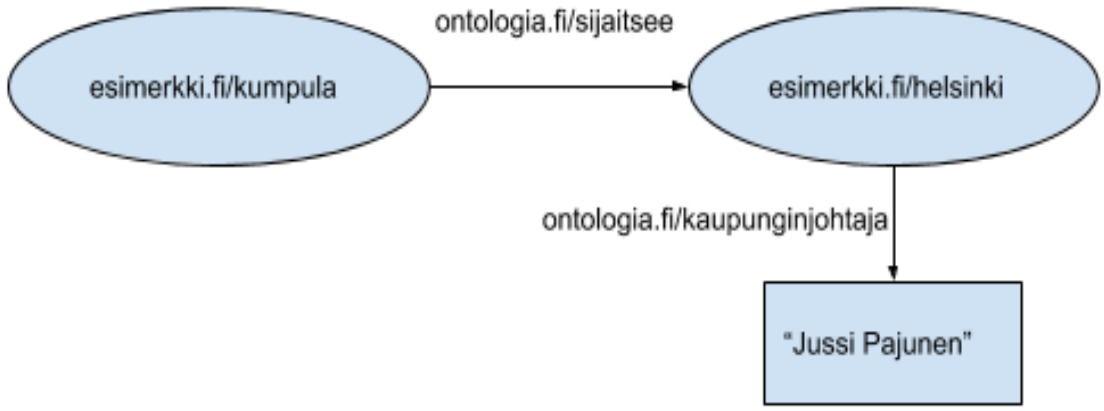
Objekti kertoo kuvailtavissa olevan ominaisuuden arvon [CWL14]. Kolmikon muista jäsenistä poiketen se voi olla resurssin tai tyhjän solmun lisäksi myös literaali. Jos esimerkiksi halutaan määritellä henkilöllä olevan sähköpostiosoite, niin objekti luultavasti olisi merkkijono eli henkilön sähköpostiosoite. Jos taas haluttaisiin kertoa Kumpulan sijaitsevan Helsingissä, niin olisi semanttisen ymmärrettävyyden kannalta järkevämpää, että myös objekti (Helsinki) olisi yksilöity resurssi. Silloin se olisi koneellisesti ymmärrettävissä ja resurssin URI-tunnistetta seuraamalla voi löytää lisää relevanttia tietoa.

2.1.3 Lausumat ja kolmikot

RDF:n tietomalli rakentuu lausumista, joissa subjekti-predikaatti-objekti -kolmikoilla

kuvataan jotain resurssin ominaisuutta tai suhdetta [DMM00]. Esimerkiksi tämän dokumentin kirjoittajaa voisi kuvata lausumalla ”Dokumentin (subjekti) kirjoittaja on (predikaatti) Topi (objekti)”.

RDF-tieto ei yleensä koostu pelkästään irrallisista lausumista. Jokaiseen lausuman resurssiin voi liittyä monia muita lausumia, joissa kerrotaan niistä lisää. Näin tietoa yhteen linkittämällä syntyy verkkomainen tietorakenne [DMM00]. Esimerkiksi kuvassa 1 on kahden RDF-lausekkeen muodostama verkko: ”Kumpula sijaitsee Helsingissä” ja ”Helsingin kaupunginjohtaja on Jussi Pajunen”. Esimerkissä objekti ”Jussi Pajunen” ei ole resurssi, vaan pelkkä merkkijonoliteraali.



Kuva 1: Esimerkki RDF-lausekkeiden muodostamista yhteyksistä

Jos RDF-kolmikko ei noudata tietomallin rajoitteita subjektin, predikaatin ja objektin tyypeistä, niin kolmikkoa kutsutaan yleisluontoiseksi RDF-kolmikoksi (*generalized RDF triple*) [CWL14]. Tällaisissa kolmikoissa mikä vain osa voi olla URI-tunnisteen osoittama resurssi, literaali tai tyhjä solmu, eivätkä näin ole virallisen spesifikaation mukaisia.

2.2 Formaatit

RDF-tietomalli on abstrakti malli, jolle ei ole tiettyä määrättyä esitysformaattia [CWL14]. Sen sijaan sille on olemassa useita vaihtoehtoisia esitystapoja. RDF:n alkuperäinen W3C:n standardoima esitysformaatti on XML-pohjainen RDF/XML [GaS14]. RDF/XML:n lisäksi nykyään on käytettävissä useita muitakin formaatteja. Suosittuja ovat muun muassa kompaktin syntaksin sisältävä Turtle (*Terse RDF Triple*

Language) [BBP14] ja JSON-pohjainen JSON-LD (*JavaScript Object Notation for Linked Data*) [SKL14]. RDF-tietoa voi myös upottaa suoraan HTML-dokumentteihin RDFa:n (*Resource Description Framework in Attributes*) avulla [ABM15].

2.2.1 RDF/XML

RDF/XML on W3C:n alun perin RDF:n kanssa standardoima laajasti tuettu esitysformaatti, joka perustuu nimensä mukaisesti XML-metakieleen [GaS14]. RDF-kolmikoiden subjektit, predikaatit ja objektit esitetään XML-elementteinä tai elementtien attribuutteina.

Vaikka RDF/XML on nykyäänkin yleisesti käytössä, on sitä kohtaan esitetty myös kritiikkiä [DMV00]. Erityisesti XML:ään liittyvä runsasmerkkisyys voi olla ongelma, koska se johtaa suurempiin tiedostokokoihin ja vaatii enemmän siirrettävää dataa. Myös paremmin ihmisen käsiteltävissä olevat esitysformaatit ovat haastajia RDF/XML-syntaksille.

RDF/XML-dokumentti alkaa XML-deklaraatiolla, jonka jälkeen tulee varsinainen RDF-juurisolmu <*rdf:RDF*> [GaS14]. Juurisolmu identifioi kyseisen dokumentin RDF-dokumentiksi. Sen sisään määritellään myös käytössä oleva nimiavaruus. Eri nimiavaruuksille voi lisäksi antaa juurisolmussa lyhenteet, jotta pitkää nimiavaruuspolkua ei tarvitsisi käyttää jokaisen solmun kohdalla uudestaan. Voimme esimerkiksi määritellä lyhenteen *ont* tarkoittavan URI:a http://*esimerkki.fi/ontologia*, jonka jälkeen nimiavaruuteen voi viitata kirjoittamalla esimerkiksi *esim:väkiluku* kertoaksemme, että kyseessä on edellä määritellystä nimiavaruudesta löytyvä määritelmä väkiluvulle. RDF ei siis itsessään määrittele mitään resursseja formaalisti, vaan sitä käytetään vain tiedon ja suhteiden mallintamiseen [CWL14].

Resurssia kuvaileva RDF-lauseke määritellään <*rdf:Description*>-elementillä, jonka sisälle lausekkeen eri osat määritellään [CWL14]. Subjekti eli kuvailtava kohde identifioidaan elementin sisällä olevalla *rdf:about*-attribuutilla, eli esimerkiksi <*rdf:Description rdf:about=”esimerkki.fi/resurssit/Kumpula”*>.

Lausekkeen predikaatti ja objekti määritellään joko <*rdf:Description*>-elementin sisään omaksi elementikseen tai <*rdf:Description*>-elementin attribuutiksi. Elementin tai attribuutin nimi kertoo predikaatin ja sen sisällä oleva arvo objektin. Esimerkiksi <*esim:väkiluku*>*3924*</*esim:väkiluku*> määrittelisi kuvailtavissa olevan resurssin

väkiluvuksi 3924. Lyhenne *esim* viittaa aiemmin juurisolmussa määriteltyyn nimiavaruuteen, josta pitäisi löytyä määritelmä termille.

Elementin sisällä oleva arvo määrittää objektin, mutta pelkän merkkijonon tai luvun sijaan objekti voi olla myös toinen resurssi, jolloin objektista saadaan paremmin semanttista tietoa. Kumpulasta voitaisiin myös kertoa sen kuuluvan Helsinkiin: *<cd:artist rdf:Resource=”esimerkki.fi/resurssit/Helsinki”/>*. Tässä attribuutti rdf:Resource kertoo objektin olevan resurssi.

Resurssia kuvailevan *<rdf:Description>*-elementin sisään voi liittää useampia kuvauksia, jolloin muodostuu useita RDF-kolmikoita ja verkko semanttista tietoa. Esimerkissä 1 on määritelty Kumpulalle sijainti ja väkiluku, joista muodostuu luonnollisella kielellä sanottuna RDF-kolmikot ”Kumpula sijaitsee Helsingissä” ja ”Kumpulan väkiluku on 3924”.

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ont="esimerkki.fi/sanasto">

<rdf:Description
  rdf:about="esimerkki.fi/resurssit/Kumpula">
  <ont:sijaitsee> rdf:resource=”esimerkki.fi/resurssit/Helsinki”</ont:sijaitsee>
  <ont:väkiluku>3924</ont:väkiluku>
</rdf:Description>

</rdf:RDF>
```

Esimerkki 1: Esimerkki Kumpulaa kuvailevasta RDF/XML-dokumentista.

RDF/XML mahdollistaa myös erilaisten kokoelmien määrittelyn [CWL14]. Kokoelmat ovat *<rdf:Bag>*, *<rdf:Seq>* ja *<rdf:Alt>*. Kokoelmaelementillä *<rdf:Bag>* voidaan määritellä ryhmä arvoja, joiden ei tarvitse olla määrätyssä järjestyksessä. Arvot tulevat elementin alle *<rdf:li>*-alielementteinä. Kokoelmalla *<rdf:Alt>* taas määritellään jollekin arvolle vaihtoehtoisia arvoja. Elementillä *<rdf:Seq>* määritellään ryhmä arvoja määrättyyn järjestykseen. Järjestettyä kokoelmaa käytetään, jos halutaan säilyttää arvot esimerkiksi aakkosjärjestyksessä.

2.2.2 Turtle

Turtle (*Terse RDF Triple Language*) on W3C:n standardoima vaihtoehtoinen formaatti RDF-datan esittämiseen ja serialisoimiseen [BBP14]. Turtle ei pohjaudu XML:ään ja onkin RDF/XML-formaattiin verrattuna kompaktimpi ja kenties helpommin ihmisen manuaalisesti käsiteltävissä yksinkertaisuutensa ansiosta.

Yksinkertaisimmillaan Turtlessa esitetään RDF-kolmikkoja kolmena toisistaan välilyönnillä erotettuina URI-tunnisteina [BBP14]. Kolmikko päätetään pisteeseen. Esimerkissä 2 on yksinkertainen Turtle-syntaksilla määritelty RDF-kolmikko, joka kertoo subjekti-predikaatti-objekti -kaavalla Kumpulan kaupunginosan sijaitsevan Helsingissä.

```
<esimerkki.fi/resurssit/kumpula>
<esimerkki.fi/sanasto/sijaitsee> <esimerkki.fi/resurssit/helsinki> .
```

Esimerkki 2: Esimerkki yksinkertaisesta Turtle-kolmikosta.

Usein samasta subjektista halutaan kertoa useita lausumia. Tätä varten Turtlessa voi ketjuttaa useita lausumia yhdelle subjektille puolipisteen avulla [BBP14]. Esimerkissä 3 olevat kaksi lausumaa kertovat yhteisen subjektin avulla Kumpulan sijaitsevan Helsingissä, ja että sen ruotsinkielinen nimi on Gumtäkt. Puolipiste siis kertoo seuraavan predikaatti-objekti-parin kuvaavan edellä määriteltyä subjektia ja mahdollistaa tiiviimmän syntaksin.

```
<esimerkki.fi/resurssit/kumpula>
<esimerkki.fi/sanasto/sijaitsee> <esimerkki.fi/resurssit/helsinki> ;
<esimerkki.fi/sanasto/nimi-ruotsiksi> ”Gumtäkt” .
```

Esimerkki 3: Esimerkki kahdesta lausumasta samalla subjektilla.

Turtlella on mahdollista myös ketjuttaa lausumia samalla subjektilla ja predikaatilla, eli kertoa samasta resurssista sama tieto useilla eri vaihtoehtoisilla arvoilla [BBP14]. Arvot eli objektit erotetaan toisistaan pilkulla, jolloin syntyy objektilista. Esimerkissä 4 on lueteltu Kumpulan postinumerot käyttämällä objektiketjutusta.

```
<esimerkki.fi/resurssit/kumpula>
<esimerkki.fi/sanasto/postinumero> ”00520”, ”00550”, ”00560”, ”00610” .
```

Esimerkki 4: Esimerkki objektilistauksesta.

Turtlessa voi antaa nimiavaruudelle lyhennelmän *@prefix*-määrityksellä, jotta samaa pitkää tunnistetta ei tarvitsisi kirjoittaa aina uudestaan. Esimerkissä 5 on kuvattu lyhennelmän käyttöä Turtlella.

```
@prefix res: <www.esimerkki.fi/resurssit> .
@prefix ont: <www.esimerkki.fi/sanasto> .
res:helsinki ont:sijaitsee res:suomi .
```

Esimerkki 5: Esimerkki URI-tunnisteen lyhentämisestä.

2.2.3 JSON-LD

JSON-LD (*JavaScript Object Notation for Linked Data*) [SKL14] on W3C:n suosittelema standardi linkitetyn datan esittämiseen ja välittämiseen JSON-pohjaisesti [JSON]. Se on suunniteltu pääasiassa yhteentoimivien web-palveluiden avuksi ja linkitetyn datan tallentamiseksi JSON-pohjaisiin tietovarastoihin [SKL14].

Vaikka JSON-LD on tarkoitettu yleisesti linkitetyn datan esittämiseen, on se samalla myös konkreettinen RDF-esitysformaatti, koska sillä pystyy toteuttamaan RDF-spesifikaation määrittelyt [SKL14]. JSON-LD-dokumentti voi siis samalla olla myös validi RDF-dokumentti ja validi JSON-dokumentti. JSON-LD kuitenkin laajentaa RDF-tietomallia sallimalla RDF-kolmikon predikaatin olevan tyhjä solmu, eli sillä voi mallintaa yleisluontoisia RDF-kolmikoita, jotka eivät ole validia RDF-tietoa. JSON-LD myös tukee JSON:n natiiveja tietotyyppejä, joita kaikkia RDF ei tunne. Formaatin spesifikaatio kuitenkin määrittelee muunnossäännöt niiden välille. JSON-LD:n etuna on JSON-formaatin suosio ja se, että kaikki valmiit JSON-kirjastot tukevat sen käsittelyä natiivisti [SKL14]. Esimerkki 6 on yksinkertainen esimerkki RDF-tiedosta JSON-LD-muodossa.

```
{
 "@context": {
  "ont": "http://www.esimerkki.fi/sanasto/",
 },
 "@id": "term:Kumpula",
 "ont:sijaitsee": {
  "@id": "term:Helsinki"
 },
 "ont:väkiluku": 3924
}
```

Esimerkki 6: Esimerkki yksinkertaisesta JSON-LD-dokumentista.

2.2.4 RDFa

RDFa (*Resource Description Framework in Attributes*) on W3C:n suosittelema standardi semanttisen metadatan upottamiseksi erityisesti HTML-dokumentteihin, mutta yleisesti myös muihin XML-pohjaisiin asiakirjoihin [ABM15]. Upottamista varten RDFa sisältää joukon merkintäattribuutteja, joilla metadata liitetään dokumentin elementteihin.

RDFa:n vahvuutena on helppokäyttöisyys, jonka ansiosta sen mahdollinen käyttäjäkunta on laaja. RDFa:ta käyttääkseen ei tarvitse ymmärtää RDF-tietomallia tai muita RDF-syntakseja, sillä metatiedon lisäämiseksi tarvitsee vain valita käyttöön jokin sanasto ja lisätä HTML-elementtiin uusi attribuutti viittaamaan sanaston termiin [ABM15]. Esimerkiksi Dublin Core -sanastoon [DBLCR] viittaamalla voisi annotoida blogikirjoituksen otsikon, tekijän ja päivämäärän koneellisesti ymmärrettäväksi.

Esimerkissä 7 on määritelty HTML-dokumentin *body*-osio käyttämään Dublin Core -sanastoa *vocab*-attribuutilla. Lisäksi kolmeen elementtiin on lisätty tieto niiden merkityksestä käyttäen *property*-attribuutteja. *Vocab*-attribuutin ansiosta *property*-attribuuteissa ei tarvitse enää kirjoittaa sanaston koko osoitetta, vaan pelkän termin käyttäminen riittää. RDF/XML:n ja Turtlen tapaan *prefix*-attribuutilla voisi määritellä useita sanastoja käytettäväksi lyhenteen kautta.

Esimerkin jokaisesta kolmesta *property*-määrityksestä muodostuu RDF-kolmikko [ABM15]. Kolmikoiden subjektina on tässä tapauksessa kyseinen web-sivu, eli *esimerkki.fi/blogit/bob/post23*. Subjektiksi voisi määritellä myös vain jonkin osan sivua *resource*-attribuutilla, jos sivulla on useita eri resursseja, kuten esimerkiksi useita

blogikirjoituksia yhdellä sivulla. Kolmikoiden predikaatit ovat *property*-attribuutin osoittamat termit, esimerkiksi *creator*. Objektit eli arvot taas ovat annotoitujen elementtien sisällä olevat arvot.

esimerkki.fi/blogit/bob/post23

```
<html>
<body vocab="http://purl.org/dc/terms/">
  <h2 property="title">Blogiotsikko</h2>
  <p>Kirjoitettu: <span property="created">2016-02-02</span></p>
  <p>Blogiteksti. Kirjoittanut <span property=”creator”>Bob </span></p>
</body>
</html>
```

Esimerkki 7: Esimerkki RDFa-tiedosta upotettuna HTML-dokumenttiin.

3 Kyselykieli – SPARQL

SPARQL (*SPARQL Protocol and RDF Query Language*) on W3C:n standardoima SQL-tyylinen semanttinen kyselykieli tiedon hakemiseen ja muokkaamiseen RDF-muotoisissa tietokokoelmissa [HaS13]. SPARQL on RDF:n lisäksi yksi semanttisen webin pohjan muodostavista teknologioista, sillä se on hallitseva standardi semanttisen tiedon kyselyyn [Hen09].

SPARQL-kielen versio 1.0 standardoitiin vuonna 2008 ja versio 1.1 vuonna 2013 [HaS13]. Versiossa 1.1 kieleen lisättiin muun muassa alikyselyt, koostefunktiot ja tiedonhaku useasta etänä sijaitsevasta SPARQL-päätepisteestä alikyselyiden avulla. Useaan päätepisteeseen kohdistuvia kyselyitä käsitellään erikseen tutkielman kappaleessa 6.

RDF-muotoisen tiedon kyselyyn on olemassa myös muita kyselykieliä [Kar02, BrA03], mutta SPARQL on ainoa W3C:n suosittelema standardi ja se onkin noussut erittäin suosituksi julkaisunsa jälkeen [Hen09]. SPARQL:n useille ohjelmistokehittäjille tuttu SQL-tyylinen syntaksi myös helpottaa sen omaksumista.

3.1 Käyttö

SPARQL-kieltä käytetään tiedon hakuun joko tavallisesta RDF-muotoisesta tiedosta tai etätietovarastosta, jolla on SPARQL-päätepiste (*SPARQL endpoint*) [HaS13]. SPARQL-päätepiste on osoite, jossa SPARQL-protokollapalvelu kuuntelee pyyntöjä asiakasohjelmilta [FWC13]. Päätepisteen kautta käyttäjät voivat suorittaa kyselyjä sen taustalla olevaan tietovarastoon joko päätepisteen mahdollisesti tarjoaman käyttöliittymän kautta tai erillistä asiakasohjelmaa käyttäen.

Esimerkiksi Apachen semanttisen webin ohjelmistokehys Jena sisältää suositun SPARQL-asiakasohjelman nimeltä ARQ [JENA]. ARQ tukee kaikkia SPARQL:n version 1.1 uusia ominaisuuksia, kuten useasta päätepisteestä tiedon hakemista. SPARQL-kielelle on myös saatavilla implementaatioita useiden suosittujen ohjelmointikielten kanssa käytettäväksi [DNRDF, SESAM, EARDF, RASQL].

Kyselykielen lisäksi SPARQL-standardi sisältää myös protokollan kyselyiden ja niiden vastausten lähettämiseen webissä asiakasohjelman ja SPARQL-etäpalvelun välillä

[FWC13]. Protokolla toimii HTTP-protokollan päällä. Protokolla määrittelee, miten asiakasohjelma ja etäpalvelu kommunikoivat, ja mahdollistaa näin SPARQL-päätepisteiden julkaisemisen verkossa.

3.2 Kyselyt

SPARQL-kyselyt toimivat vertaamalla kyselyssä annettuja kolmikoita kyseltävän tiedon RDF-kolmikoihin [ArP11]. Kyselyn kolmikot ovat kuin normaaleja kolmikoita, mutta niissä mikä vain osa voi olla kyselyssä käytössä oleva muuttuja. Kolmikot täsmäävät, jos kyselyn kolmikon määritellyt osat sopivat datan RDF-kolmikon osiin. Jos kyselyn kolmikossa on muuttuja, niin siihen asetetaan täsmänneestä kolmikosta vastaavalta paikalta saatu arvo.

SPARQL-kysely koostuu useasta osasta. Kyselyn alussa voidaan määritellä lyhenteitä URI-tunnisteilla kuvattaville nimiavaruuksille *PREFIX*-komennolla, jotta pitkiä tunnisteita ei tarvitsisi kirjoittaa kyselyssä aina uudestaan [HaS13]. Voidaan esimerkiksi määritellä kyselyn alkuun *PREFIX esim: <http://esimerkki.fi/>* ja viitata myöhemmin kyseiseen nimiavaruuteen pelkällä etuliitteellä *esim*.

Mahdollisten nimiavaruusmäärittelyjen jälkeen alkaa varsinainen kysely yhdellä neljästä kyselymuodosta: *SELECT, CONSTURCT, ASK* tai *DESCRIBE* [HaS13]. Kyselymuodon jälkeen määritellään halutut tulosmuuttujat kysymysmerkillä ja muuttujan nimellä, esimerkiksi *?sukunimi*. Kyselyssä saadut tulokset asetetaan tulosmuuttujiin.

Komennolla *FROM* voidaan määritellä tietokokoelma, johon kysely kohdistetaan [HaS13]. Ehtoja kyselylle on mahdollista määritellä *WHERE*-komennolla. Kyselyn tuloksia on myös mahdollista rajata tai järjestää koostefunktioilla ja lisämäärityksillä.

Alikyselyitä voi määritellä SPARQL-kielessä yksinkertaisesti tekemällä alkuperäisen kyselyn sisään toisen kyselyn [HaS13]. Niiden avulla voi jakaa monimutkaisen kyselyn pienempiin selkeisiin kokonaisuuksiin. Kyselyä suorittaessa alikyselyt ajetaan ensin, jolloin niiden tulokset ovat käytettävissä ulommissa kyselyissä.

3.2.1 SELECT

Kyselymuodoista *SELECT* eniten SQL-kyselyjä. Se palauttaa kyselyn kolmikoiden ja kyseltävän tiedon RDF-kolmikoiden täsmäytyksen tuloksena tulosmuuttujiin asetetut

arvot [HaS13]. Sen avulla voi myös käsitellä kyselyn tuloksena saatuja arvoja, esimerkiksi laskea yhteen kaksi kokonaislukutyyppistä muuttujaa ja asettaa saatu arvo uuteen muuttujaan.

Esimerkkinä *SELECT*-kyselystä taulukossa 1 on yksinkertaista henkilöistä kertovaa RDF-dataa Turtle-muodossa ja kyseiseen tietoon kohdistuva kysely, sekä kyselyn palauttama tulosjoukko. Kyselyllä halutaan selvittää, kuka tuntee kenet. Kyselyssä määritellään aluksi kaksi lyhennettä, *henk* (http://esimerkki.fi/henkilot/) ja *ont* (http://esimerkki.fi/sanasto/). Tulosmuuttujiksi asetetaan *?kuka* ja *?tunteeKenet* ilmaisemaan kenen kahden välillä suhde on. Kyselyn ensimmäinen kolmikko määrittelee, että *a*:n pitää tuntea *b*. Toinen kolmikko taas asettaa *a*:n nimen tulosmuuttujaan *?kuka* ja kolmas kolmikko *b*:n nimen tulosmuuttujaan *?tunteeKenet*. Kyselyn perusteella saadaan kolme tulosta: Topi tuntee Timin, Topi tuntee Tarun ja Timi tuntee Tarun.

```
@prefix : <http://esimerkki.fi/>

:topi :nimi "Topi" .
:topi :tuntee :timi .
:topi :tuntee :taru .

:timi :nimi "Timi" .
:timi :tuntee :taru .

:taru :nimi "Taru" .
```

```
PREFIX : <http://esimerkki.fi/>

SELECT ?kuka ?tunteeKenet WHERE {
  a? :tuntee ?b .
  a? :nimi ?kuka .
  b? :nimi ?tunteeKenet .
}
```

kuka	**tunteeKenet**
”Topi”	”Timi”
”Topi”	”Taru”
”Timi”	”Taru”

Taulukko 1: Esimerkkidata Turtle-muodossa, SPARQL-kysely ja tulosjoukko.

3.2.2 CONSTRUCT

Tavanomaisen taulukoidun datan sijaan SPARQL:n komento *CONSTRUCT* palauttaa joukon RDF-kolmikoita [HaS13]. Pelkän tiedon kyselyn lisäksi sillä voi myös muokata olemassa olevaa dataa tai luoda hyödyllisiä uusia kolmikoita. *CONSTRUCT*-kyselyssä määritellään tulosmuuttujien lisäksi malli rakennettaville kolmikoille. Kolmikot luodaan ottamalla kyselyn palauttamat kolmikot ja korvaamalla mallin määräämät kohdat sen arvoilla.

Taulukossa 2 on esimerkkidataa henkilöiden sukulaissuhteista Turtle-muodossa, *CONSTRUCT*-kysely uusien RDF-kolmikoiden rakentamiseksi lapsenlapsien kuvaamiseksi sekä kyselyn palauttama kolmikkojoukko. Kyselyssä määritellään aluksi malli rakennettaville kolmikoille, eli esimerkissä *?henkilö :onLapsenlapsi ?isovanhempi*. Muuttujat *?henkilö* ja *?isovanhempi* korvataan normaalisti kyselyn palauttamilla tuloksilla, mutta mallin määrittelemä *:onLapsenlapsi* jätetään paikoilleen. Näin saadaan kaksi RDF-kolmikoilla kuvattavaa toteamusta: Taru on Timin lapsenlapsi ja Timo on Timin lapsenlapsi.

@prefix : <http://esimerkki.fi/> :taru :onLapsi :topi :timo :onLapsi :topi :topi :onLapsi :timi

PREFIX : <http://esimerkki.fi/>
 	
CONSTRUCT { ?henkilö :onLapsenlapsi ?isovanhempi . } WHERE {
?henkilö :onLapsi ?vanhempi .
?vanhempi :onLapsi ?isovanhempi .
}

:taru :onLapsenlapsi :timi . :timo :onLapsenlapsi :timi .

Taulukko 2: Esimerkkidata Turtle-muodossa, CONSTRUCT-kysely ja muodostetut kolmikot.

Jos kyselyn rakentamat kolmikot sisältävät epästandardia tietoa, kuten merkkijonon subjektin tai predikaatin paikalla, niin kyseistä kolmikkoa ei lisätä palautettavien kolmikoiden joukkoon [HaS13]. Valmiin kyselyn jälkeen palautetut kolmikot voidaan liittää suoraan tietojoukkoon, johon kysely tehtiin. Uusista kolmikoista voi myös tarvittaessa luoda uuden tietojoukon.

3.2.3 ASK

SPARQL:n *ASK*-kyselyä voidaan käyttää selvittämään, löytyykö tietojoukosta vastausta johonkin tiettyyn kyselyyn [HaS13]. Kysely palauttaa totuusarvon, mutta ei mitään muuta tietoa mahdollisista tuloksista. *ASK*-kysely on hyödyllinen jonkin yksinkertaisen totuusarvolla kuvattavan tiedon hakemiseen tai etänä sijaitsevien tietojoukkojen sisällön selvittämiseen etukäteen ennen varsinaisten kyselyiden tekemistä.

Esimerkiksi taulukossa 3 tehdään henkilöiden nimistä kertovaan dataan *ASK*-kysely selvittämään, onko tietojoukossa jonkun nimi ”Timi”. Kysely palauttaa totuusarvon *false*, koska nimeä ei löydy.

@prefix : <http://esimerkki.fi/> :topi :nimi "Topi" :taru :nimi "Taru"

PREFIX : <http://esimerkki.fi/>
 	
ASK {
  ?henkilö :nimi "Timi" .
}

false

Taulukko 3: Esimerkkidata Turtle-muodossa, ASK-kysely ja palautettu totuusarvo.

3.2.4 DESCRIBE

SPARQL:n *DESCRIBE*-kysely muistuttaa *SELECT*-kyselyä, mutta tulosjoukon sijaan se palauttaa vapaamuotoisen kuvauksen löydetystä aineistosta [HaS13]. Kuvaus on joukko RDF-kolmikkoja, mutta kyselyn kohteena oleva päätepiste määrittelee itse palautettavan sisällön. *DESCRIBE*-kyselyä voi esimerkiksi käyttää lisätiedon selvittämiseen ennen varsinaisen kyselyn tekoa, mutta SPARQL-standardi ei takaa saatujen vastausten olevan kyselylle relevantteja.

3.2.5 Koostefunktiot ja lisämääritykset

SPARQL sisältää useita koostefunktioita ja kyselyn lisämäärityksiä [HaS13]. Koostefunktioilla käsitellään kyselyiden palauttamaa tietokokoelmaa jälkikäteen ja lisämäärityksillä voidaan rajata jo ennalta kyselyn tuloksia. Koostefunktiot tulivat osaksi SPARQL-standardia versiossa 1.1.

Oletuksena kaikki SPARQL-kyselyssä kolmikoilla määritellyt ehdot on täytyttävä, jotta tulos otetaan mukaan tulosjoukkoon [HaS13]. Ehtoja voi kuitenkin myös asettaa valinnaisiksi *OPTIONAL*-määritystä käyttämällä. Tällöin tulosjoukkoon lisätään valinnaiseksi määritelty tieto, jos se on saatavilla, mutta tulosta ei myöskään kokonaan hylätä jos sitä ei löydy. Jos valinnaista tietoa ei löytynyt, sen kohta tulosjoukossa jätetään tyhjäksi. Valinnaiset määrittelyt ovat hyödyllisiä, jos esimerkiksi halutaan tehdä kysely henkilötietoja sisältävään tietojoukkoon, jossa kaikille ei ole määritelty puhelinnumeroa. Määrittelemällä kyselyssä puhelinnumero valinnaiseksi se saadaan löydyttäessä mukaan, mutta henkilöitä ei myöskään jää tuloksista pois puuttuvan puhelinnumeron takia.

Kyselyn ehdot määritteleviä kolmikoita voi yhdistää *UNION*-määrityksen avulla [HaS13]. Tällöin tietojoukosta voidaan hakea vaihtoehtoisia ominaisuuksia. Löydetyt arvot yhdistetään yhdeksi joukoksi.

ORDER BY -määritys järjestää tulokset jonkin tietyn tai usean tulosmuuttujan mukaan [HaS13]. Järjestys on oletuksena nouseva, mutta sen voi määritellä laskevaksi *DESC*-määrityksellä tai todeta eksplisiittisesti nousevaksi *ASC*-määrityksellä. URI-tunnisteita käsitellään järjestäessä merkkijonoina.

DISTINCT-määritys poistaa tulosjoukosta duplikaatit [HaS13]. *LIMIT* taas rajoittaa tulosjoukon koon tiettyyn määrään. Jos tulosjoukko on *LIMIT*-määritystä käytettäessä pienempi kuin määritelty koko, niin palautetaan koko joukko. *OFFSET* palauttaa tulokset vasta jostain tietystä tuloksesta alkaen.

Tulosjoukon rajaaminen tietyn ehdon mukaan onnistuu *FILTER*-määrityksellä [HaS13]. Rajaus kohdistetaan johonkin muuttujaan, jolloin mukaan otetaan vain rajauksen läpäisseet tulokset. Merkkijonoja voi rajata esimerkiksi säännöllisten lausekkeiden kautta ja kokonaislukuja tai päivämääriä vertailuoperaattorien avulla.

Tulosten suodattamiseen voidaan käyttää myös *MINUS*-määritystä [HaS13]. Sen avulla voidaan etsiä tulokset, jotka eivät täytä jotain tiettyä ehtoa. Esimerkiksi henkilötietoja sisältävästä tietojoukosta voitaisiin sen avulla etsiä kaikki, joilla ei ole määritelty puhelinnumeroa.

Kyselyn tuloksia voi SPARQL-kielessä muokata koostefunktioiden avulla [HaS13]. *COUNT* laskee palautettujen tulosten määrän, *SUM* laskee yhteen palautettujen tulosten

yhteissumman, *MIN* palauttaa tulosten pienimmän arvon, *MAX* palauttaa tulosten suurimman arvon ja *AVG* laskee tulosten keskiarvon.

Jos koostefunktion tuloksia halutaan ryhmitellä jonkin muuttujan mukaan, niin käytettävissä on *GROUP BY* -määritys [HaS13]. Lisäksi koostefunktioiden tuottamia tuloksia voi rajata *HAVING*-määrityksellä. Rajaus toimii kuten *FILTER*-operaatio, mutta sen voi kohdistaa koostefunktioiden tuloksiin.

4 Ontologiat – OWL

Filosofiassa ontologia tutkii olemisen ja olemassaolon konsepteja. Tietojenkäsittelytieteessä ontologioita käytetään kuvamaan koneellisesti ymmärrettävällä tavalla käsitteitä tai jotain tiettyä sovellusaluetta, jossa se mallintaa kyseisen alueen entiteettejä ja niiden välisiä suhteita [CJB99]. Ontologioilla voidaan määritellä käsitteitä eksaktisti ja luokitella käsitteet hierarkioihin. Niillä voidaan myös määritellä päättelysääntöjä, kuten että naispuolinen vanhempi on äiti tai että vanhemman isä on isoisä. Ontologioita voidaan käyttää myös apuna tiedon yhteensovittamiseen eri sovellusten välillä tai saman tiedon uudelleenkäyttöön uusissa sovelluksissa [GoC02].

OWL (*Web Ontology Language*) on W3C:n standardi web-pohjaisten ontologioiden mallintamiseen [DeS04]. OWL koostuu useista lajeiksi kutsutuista alikielistä. Alkuperäinen OWL standardoitiin vuonna 2004 ja uusi versio OWL 2 vuonna 2012 [HKP12]. OWL 2 toi mukanaan muun muassa uusia alikieliä ja syntaksimuutoksia, mutta säilytti yhteensopivuuden aiemman version kanssa.

OWL perustuu RDF-tietomalliin [DeS04]. OWL kuitenkin laajentaa RDF:ää sanastolla käsitteiden välisten monimutkaisten suhteiden ja rajoitteiden kuvailemiseen, mikä tekee edistyneemmästä koneellisesta päättelystä mahdollista [HPV03]. OWL laajentaa myös RDFS:ää (*RDF Schema*), joka tarjoaa yksinkertaisia luokkia ja attribuutteja RDF-tiedon luokitteluun [BrG14].

4.1 Edeltäjät

Web-pohjaisten ontologioiden rakentamiseen on ollut saatavilla kieliä 2000-luvun alusta lähtien [HeM00, HeH00]. Myös 1990-luvun lopulla oli jo julkaistu RDFS, joka tarjoaa yksinkertaisia luokkia RDF-tiedon luokitteluun [BrG14]. Vaikka RDFS ei mahdollista kovin ilmaisuvoimaisien ontologioiden määrittelyä, se on kuitenkin vaikuttanut voimakkaasti myöhempien ontologiakielten syntyyn [HPV03].

Ensimmäisiä julkaistuja varsinaisia ontologiakieliä oli SHOE (*Simple HTML Ontology Extensions*) [HeH00]. SHOE on XML-pohjainen merkintäkieli, joka laajentaa HTML:ää joukolla merkintäattribuutteja semanttisen tiedon välittämiseksi. SHOE

käyttää käsitteiden nimeämiseen URI-tunnisteita, mikä osoittautui merkittäväksi innovaatioksi modernimpien kielien kannalta [HPV03].

Toinen varhainen ontologiakieli oli Yhdysvaltain asevoimien tutkimuslaitoksen DARPA:n kehittämä DAML (*DARPA Agent Markup Language*) [HeM00]. DAML:n suunnittelijat ottivat lähtokohdakseen RDFS:n, mutta tavoitteena oli laajentaa siitä voimakkaampi kieli. Samoihin aikoihin kehitettiin myös OIL-projektia (*Ontology Interface Layer*) [FVH01]. DAML:n ja OIL:n kehittäjät huomasivat pian päämääriensä olevan vastaavanlaisia, joten projektit päätettiin yhdistää. Yhdistymisestä syntyi DAML+OIL [MFH02]. DAML+OIL on tiukemmin sidottu RDF:ään ja se mahdollistaa suoran formaalin semantiikan määrittelemisen.

Suoraan varhaisempaan DAML+OIL:iin perustuen W3C standardoi vuonna 2004 oman ontologiakielensä OWL:n [DeS04]. Nykyään OWL on webin hallitseva ontologiakielistandardi. Vuonna 2012 siitä julkaistiin uusi versio OWL 2 [HKP12].

4.2 Toimintaperiaatteet

OWL on kieli tiedon mallintamiseen [HKP12]. Tietoa mallinnetaan käsitteillä ja lausumilla, joissa yksinkertaistettuna jostain käsitteestä kerrotaan jotain. Esimerkiksi lausuman ”vanhemman isä on isoisä” voisi mallintaa OWL:lla ja sisällyttää ontologiaan. Tällaisia ontologian määrittelemiä faktoja kutsutaan aksioomiksi. OWL:ssä voi myös yhdistää käsitteitä uusien ilmaisujen luomiseen. Esimerkiksi käsitteet ”nainen” ja ”professori” voisi yhdistää uudeksi naispuolista professoria tarkoittavaksi ilmaisuksi, jolloin siitä tulee oma käsitteensä, mutta myös alkuperäiset käsitteet sisältyvät siihen.

OWL-ontologiat rakentuvat luokista, luokkien ilmentymistä eli yksilöistä ja niiden ominaisuuksista [HKP12]. Esimerkiksi käsite ihminen voisi olla luokka ja henkilö Timi sen ilmentymä. Yksilöt eivät myöskään välttämättä kuulu vain yhteen luokkaan, vaan Timi voisi kuulua myös luokkaan mies. Luokka mies voidaan lisäksi määritellä olevan ihmisen aliluokka. Käsitteiden ominaisuuksia ja suhteita toisiin käsitteisiin mallinnetaan asettamalla luokalle tai yksilölle tietty ominaisuus. Henkilöille asetettavia ominaisuuksia voisivat olla esimerkiksi ikä, pituus tai tieto äidistä, joka viittaisi toiseen henkilöön.

OWL perustuu RDF-tietomalliin, joten myös sen pohjana ovat subjekti-predikaatti-objekti-kolmikot [HKP12]. Näistä kolmikoista rakentuu faktoja kertovia lausumia, joita

kutsutaan aksioomiksi. Kuten RDF:ssä, kolmikon subjektin ja predikaatin täytyy myös OWL:ssä olla resurssiin viittaava URI-tunniste, mutta objekti voi olla pelkkä literaali. Subjektin ja objektin osoittamat resurssit voivat olla joko OWL:n luokkia tai yksilöitä, kun taas predikaatti on niiden yhteydestä kertova ominaisuus.

OWL noudattaa avoimen maailman oletusta [HKP12]. Sen mukaan jonkin faktan puuttuminen tietojoukosta ei välttämättä tarkoita faktan olevan epätosi. Toisin kuin useissa perinteisissä tietokantajärjestelmissä, OWL:ssä siis oletetaan kaiken olevan mahdollista, ellei toisin sanota. Tähän liittyen OWL:ssä voi myös asettaa kielteisiä ominaisuuksia, eli eksplisiittisesti kertoa jonkin asian olevan epätosi.

Vaikka OWL noudattaakin avoimen maailman oletusta, ei kaikkea tietoa tarvitse sanoa eksplisiittisesti, sillä faktoja on mahdollista myös päätellä tiedon perusteella implisiittisesti [ZhM03]. Jos tietojoukossa on esimerkiksi todettu sisaruksen olevan oman vanhemman toinen lapsi, niin sisarussuhteita voidaan päätellä pelkkien vanhempien perusteella toteamatta suoraan henkilöiden olevan sisaruksia. Tällaisia päättelyitä tehdään päättelijöiksi (*reasoner*) kutsuttujen sovellusten avulla. Päättelijöiden toteutus ei ole osa OWL-spesifikaatiota, joten niiden toiminta riippuu kyseessä olevasta implementaatiosta [HKP12]. Esimerkiksi Pellet on tunnettu OWL-päättelijä [SPG07].

4.3 Toiminnot

OWL ei ole sidottu mihinkään tiettyyn esitysformaattiin [HKP12], mutta tässä kappaleessa käytetään Turtle-syntaksia [BBP14] joidenkin OWL:n keskeisten toimintojen havainnollistamiseen. Esimerkit eivät sisällä nimiavaruusmäärityksiä yksinkertaisuuden nimissä. Käytettävissä olevia muita formaatteja käsitellään tarkemmin seuraavassa kappaleessa. Esimerkeistä käy ilmi toimintojen etuliitteestä (esimerkiksi *rdf:type*), että jotkin toiminnot ovat saatu suoraan RDF:stä ja RDFS:tä, joita OWL laajentaa.

OWL:ssä yksilöitä määritellään luokasta *rdf:type*-komennolla [HKP12]. Esimerkiksi *:Timi rdf:type :Mies* määrittelisi yksilön *Timi* olevan luokan *Mies* ilmentymä. Voimme myös määritellä miehen olevan luokan *Ihminen* aliluokka *rdfs:subClassOf*-komennolla: *:Mies rdfs:subClassOf :Ihminen*. OWL:ssä luokkahierarkiat ovat transitiivisia, eli koska Timi on mies, voimme myös päätellä hänen olevan ihminen.

Kuten aikaisemmin todettiin, OWL:ssä yksilö voi kuulua useaan luokkaan [HKP12]. Jos kuitenkin halutaan eksplisiittisesti todeta johonkin luokkaan kuulumisen estävän kuulumisen toiseen luokkaan, sen voi ilmaista *owl:AllDisjointClasses*-komennolla. Sen avulla voi esimerkiksi todeta, että yksilö ei voi samaan aikaan kuulua luokkiin mies ja äiti.

Luokille ja yksilöille asetetaan ominaisuuksia Turtle-syntaksissa yksinkertaisesti sijoittamalla ominaisuus predikaatiksi kahden luokan tai yksilön väliin, esimerkiksi *:Timi :vaimo :Taru* [HKP12]. Myös ominaisuuksilla voi olla luokkien tapaan aliominaisuuksia. Ominaisuuden *vaimo* voisi määritellä ominaisuuden *puoliso* aliominaisuudeksi *rdfs:subPropertyOf*-komennolla: *:vaimo rdfs:subPropertyOf :puoliso*. Myös ominaisuuksien hierarkiat ovat transitiivisia.

Ominaisuuksien määrää voi säädellä kardinaalisuusrajoitteilla [HKP12]. Esimerkiksi *owl:minQualifiedCardinality*-komennolla voi asettaa minimimäärän jollekin ominaisuudelle ja *owl:maxQualifiedCardinality*-komennolla vastaavasti maksimimäärän.

Monimutkaisempien luokkien esittämiseen OWL:ssä voi määritellä useamman luokan risteymästä uuden luokan *owl:intersectionOf*-komennolla [HKP12]. Esimerkiksi äidin voisi määritellä olevan luokkien nainen ja vanhempi risteymä. Myös useamman luokan liitoksesta voi määritellä uuden luokan *owl:unionOf*-komennolla.

OWL:ssä voi määritellä kahden käsitteen tarkoittavan samaan *owl:sameAs*-komennolla [HKP12]. Tämä on erityisen hyödyllistä semanttisen webin näkökulmasta, sillä sen sisältämät itsenäiset ja hajallaan olevat ontologiat voivat käyttää samoille asioille eri tunnisteita. Käsitteet yhdistämällä molempien ontologioiden tieto saadaan helpommin yhdistettyä. Kahden käsitteen voidaan myös todeta olevan erillisiä *owl:differentFrom*-määrityksellä tai käänteisiä *owl:inverseOf*-käskyllä.

4.4 Formaatit

OWL ei ole sidottu yhteen tiettyyn formaattiin, vaan sille on olemassa useita vaihtoehtoisia esitystapoja [HKP12]. OWL-spesifikaatio määrää ainoastaan, että kaikkien työkalujen on tuettava RDF:stä tuttua RDF/XML-syntaksia. Myös muut RDF-syntaksit, kuten Turtle, ovat käytettävissä OWL:ssä.

OWL:n funktionaalinen syntaksi (*Functional Syntax*) on W3C:n standardoima korkean tason syntaksi, joka on suunniteltu enemmänkin spesifikaatioiden ilmaisemiseen ja lähtökohdaksi työkalujen toteuttamiseen kuin varsinaisena serialisaatiomuotona [MPP12]. Siinä aksioomia kuvataan loogisten lausekkeiden tyylisesti. Esimerkissä 8 on näyte OWL:n funktionaalisesta syntaksista.

```
ClassAssertion( :Henkilö :Taru )
SubClassOf( :Nainen :Henkilö )
ObjectPropertyAssertion( :vaimo :Timi :Taru )
SameIndividual( :Timi toinenOntologia:Timi )
```

Esimerkki 8: Esimerkki OWL:n funktionaalisesta syntaksista.

Toinen yleinen korkean tason formaatti on Manchester Syntax [HoP12]. Manchester Syntax on ihmisystävällinen ja mukailee rakenteessaan luonnollista kieltä. Sen avulla on mahdollista kirjoittaa ontologioita intuitiivisen syntaksin avulla ymmärtämättä alla piileviä RDF-rakenteita. Esimerkissä 9 on yksinkertainen näyte Manchester Syntax -formaatista.

```
Individual: Taru
  Types: Henkilö
Class: Nainen
  SubClassOf: Henkilö
Individual: Timi
  Facts: vaimo Taru
  SameAs: toinenOntologia:Timi
```

Esimerkki 9: Esimerkki OWL:n Manchester Syntax -formaatista.

RDF/XML-syntaksi [GaS14] on OWL-spesifikaation mukaan ainoa formaatti, jota kaikkien työkalujen tulisi tukea [HKP12]. Se ei ole aiemmin esiteltyjen formaattien tavoin korkean tason syntaksi, vaan sitä käytetään yleisesti myös varsinaisena serialisointiformaattina. Koska RDF/XML perustuu XML-kieleen, ovat sen dokumentit verrattain runsassanaisia ja vaikeammin ihmisen ymmärrettävissä. Esimerkissä 10 on esimerkki RDF/XML-formaatin käytöstä OWL:n kanssa.

```
<Henkilö rdf:about="Taru"/>
<owl:Class rdf:about="Nainen">
  <rdfs:subClassOf rdf:resource="Henkilö"/>
</owl:Class>
<rdf:Description rdf:about="Timi">
  <vaimo rdf:resource="Taru"/>
</rdf:Description>
<rdf:Description rdf:about="Timi">
  <owl:sameAs rdf:resource="Jim"/>
</rdf:Description>
```

Esimerkki 10: Esimerkki RDF/XML-formaatin käytöstä OWL:n kanssa.

Myös Turtle-syntaksi [BBP14] on käytettävissä OWL:n kanssa [HKP12]. Turtle on kompaktimpaa ja luettavampaa kuin esimerkiksi RDF/XML-formaatti. Esimerkissä 11 on näyte Turtle-formaatilla ilmaistuista OWL-rakenteista.

```
:Taru rdf:type :Henkilö .
:Nainen rdfs:subClassOf :Henkilö .
:Timi :vaimo :Taru .
:Timi owl:sameAs toinenOntologia:Timi .
```

Esimerkki 11: Esimerkki Turtle-formaatin käytöstä OWL:n kanssa.

4.5 Alikielet

OWL koostuu kolmesta lajiksi kutsutusta alikielestä [DeS04]. Ilmaisuvoiman mukaan kasvavasti järjestettynä lajit ovat OWL Lite, OWL DL ja OWL Full. Jokainen pätevä OWL Lite -ontologia on myös pätevä OWL DL -ontologia ja jokainen pätevä OWL DL -ontologia on pätevä OWL Full -ontologia. OWL 2 sisältää myös vastaavat DL- ja Full-alikielet, mutta määrittelee lisäksi kolme profiileiksi kutsuttua uutta alikieltä: OWL 2 EL, OWL 2 QL ja OWL 2 RL [HKP12]. Alikielten suurimmat erot liittyvät niiden käytössä olevien määritysten rajoitteisiin ja sitä kautta päättelyiden tekemisen laskennalliseen vaikeuteen. Esimerkiksi Full-kielillä on mahdollista toteuttaa rakenteita, jotka aiheuttavat päättelyketjun ajautumisen ikuiseen silmukkaan.

4.5.1 OWL-lajit

OWL Lite on OWL:n alikielistä yksinkertaisin ja on tarkoitettu pääasiassa luokitteluun ja yksinkertaisten rajoitteiden määrittelyyn [DeS04]. Se muun muassa sallii

ominaisuuksien määrää säätävien kardinaalisuusrajoitteiden arvoiksi ainoastaan 0 tai 1. Liten näennäisestä rajoittuneisuudesta huolimatta on käynyt ilmi, että suurin osa monimutkaisemman OWL DL -alikielen ominaisuuksista on mahdollista toteuttaa yhdistelemällä Liten toimintoja [GHM08]. Näin Liten perusteella tehtyjen päättelyjen tekeminen on vain hieman vähemmän laskennallisesti kallista kuin OWL DL -alikielen, eikä Lite olekaan noussut merkittäväksi alikieleksi.

OWL DL -alikieli on tarkoitettu tarjoamaan suurimman mahdollisen ilmaisuvoiman säilyttäen samalla laskennallisen täydellisyyden ja ratkeavuuden [DeS04]. Toisin sanoen kaikki sen mahdollistamat päättelyt ovat koneellisesti laskettavissa äärellisessä ajassa. OWL DL:ssä on käytettävissä kaikki OWL-kielen ominaisuudet, mutta tiettyjen toimintojen käyttöä on rajoitettu laskennallisuuden säilyttämiseksi. Siinä luokka voi esimerkiksi olla monen muun luokan aliluokka, mutta ei toisen luokan ilmentymä.

OWL Full sisältää kaikki OWL:n ominaisuudet ilman rajoitteita ja on tarkoitettu mahdollistamaan maksimaalinen ilmaisuvoima takaamatta ratkeavuutta [DeS04]. Siinä voi esimerkiksi määritellä luokan olevan kokoelma yksilöitä ja samalla itsessään yksilö. Ratkeamattomuutensa takia OWL Full ei ole kovin hyödyllinen käytännön sovelluksissa [GHM08].

4.5.2 OWL 2 -profiilit

OWL 2 sisältää aikaisemman version DL- ja Full-alikieliä melko tarkasti vastaavat omat versionsa, mutta määrittelee niiden lisäksi kolme uutta alikieltä [HKP12]. Niiden ilmaisuvoima ja päättelyn tehokkuus vaihtelevat, mutta alkuperäisistä OWL-alikielistä poiketen niiden suunnittelussa on erityisesti keskitytty tiettyjen käytännön sovellusten tarpeiden täyttämiseen.

OWL 2 EL muistuttaa DL-alikieltä, mutta se on suunniteltu erityisesti monimutkaisia luokituksia ja valtavia määriä luokkia sisältäviä ontologioita varten [HKP12]. EL-alikielen päättelyt voidaan toteuttaa polynomisessa ajassa suhteessa ontologian kokoon. OWL 2 EL on käytännöllinen varsinkin suurten ontologioiden toteuttamiseen, esimerkiksi lääketieteen tietopankki SNOMED-CT on tunnettu ontologia, joka on otettu huomioon EL-alikieltä suunniteltaessa [GHM08].

OWL 2 QL on suunniteltu käytettäväksi helposti yhdessä perinteisten relaatiotietokantojen kanssa [HKP12]. Ontologian sisältämä tieto voidaan tallentaa

relaatiotietokantaan, mutta rajapintoja käyttämällä se näyttää ulospäin tavalliselta OWL-ontologialta. Helppo integroituminen olemassa olevien tietokantajärjestelmien kanssa mahdollistaa niiden tarjoamien toimintojen hyödyntämisen.

OWL 2 RL on tarkoitettu sovelluksille, jotka haluavat käyttää ontologioita sääntöjen kuvailemiseen [HKP12]. Se sopii käytettäväksi erityisesti jos valmista RDF-dataa halutaan manipuloida ontologian määrittelemien sääntöjen avulla.

5 Linkitetty data ja avoimet tietolähteet

Klassinen ihmisten luettavaksi tarkoitettu web rakentuu toisiinsa linkitetyistä dokumenteista, joita ihmiset selaavat verkkoselaimilla ja hakurobotit indeksoivat ja analysoivat. Linkit dokumenttien välillä luovat pohjan webin selaamiselle ja tekevät siitä yhtenäisen globaalin tietoverkon. Linkitetty data (*linked data*), tai linkitetyn datan verkko (*web of linked data, web of data*), toimii samalla periaatteella, mutta keskenään linkitettyjen dokumenttien sijaan se koostuu linkitetyistä RDF-pohjaisista semanttisista tietolähteistä [Biz09]. Lisäksi varsinaiset kuvailtavat käsitteet ovat linkittyneet toisten lähteiden käsitteisiin sen sijaan, että tietolähteestä olisi vain linkki toiseen tietolähteeseen.

Linkitetty data itsessään perustuu joukkoon käytäntöjä ja standardeja rakenteellisen datan julkaisusta ja yhteenlinkittämisestä verkossa [BBH09]. Linkitetystä datasta muodostuva verkko mahdollistaa uudenlaisten älykkäiden sovellusten rakentamisen ja semanttisen tiedonhaun jatkuvasti kasvavasta määrästä tietoa.

Linkitetyn datan julkaisua koordinoi löyhästi W3C:n Linked Open Data Project, jonka tarkoituksena on muuntaa ja julkaista olemassa olevia avoimia tietokokoelmia RDF-muodossa [Biz09]. Linkitetty data on kuitenkin luonteeltaan täysin avointa, joten kuka vain voi osallistua julkaisemalla verkossa dataa RDF-muodossa ja linkittämällä sen muiden tietolähteiden dataan.

5.1 Linkitetyn datan periaatteet

Avoimesta luonteestaan huolimatta linkitetyn datan julkaisua verkossa ohjaa Tim Berners-Leen vuonna 2006 asettamat neljä periaatetta [Ber06]. Ensimmäinen periaate suosittelee käyttämään URI-tunnisteita asioiden nimeämiseen ja toinen periaate täsmentää käyttämään HTTP-URI-tunnisteita. Näin jokainen tiettyä käsitettä kuvaava tunniste on uniikki ja sen voi etsiä verkosta. Kolmas periaate kehottaa tarjoamaan hyödyllistä tietoa kyseisestä käsitteestä sen URI-tunnisteen osoittamaa sivua tarkasteltaessa [Ber06]. Tässä tulisi käyttää vakiintuneita standardeja, kuten RDF:ää. Käytännössä siis käsitteestä pitäisi löytyä lisätietoa RDF-muodossa, esimerkiksi tietoa käsitteen ominaisuuksista tai linkkejä muiden tietolähteiden samaa tarkoittaviin käsitteisiin. Neljännen periaatteen mukaan tiedon tulisi sisältää linkkejä muihin

tietolähteisiin URI-tunnisteita käyttäen, jotta käsitteestä on helppo löytää lisätietoa linkkejä seuraten [Ber06].

Tiivistettynä periaatteet ohjaavat käyttämään standardoituja web-teknologioita tiedon linkittämiseksi datan tasolla muihin tietolähteisiin. Periaatteet eivät ole ehdottomia käskyjä, eikä niiden noudattamatta jättäminen hajota mitään. Enemmänkin se vain vaikeuttaa kyseisen tiedon linkittymistä ja uudelleenkäyttämistä, jotka ovat linkitetyn datan tuottamaa suurinta arvoa [Ber06].

5.2 Avoimet tietolähteet

Kaikille avointa dataa julkaisevat avoimet tietolähteet luovat pohjan linkitetyn datan verkolle [BBH09]. Avointa dataa julkaistaan monissa muodoissa, mutta linkitetyn datan kannalta mielenkiintoisinta on semanttisessa ja linkitetyssä muodossa julkaistu data, eli käytännössä RDF-muotoinen tieto. Avoimen datan lähtökohtana on se, että data on jokaisen vapaasti käytettävissä ilman tekijänoikeussuojauksia tai muita rajoitteita. Tämän takia tietojoukon julkaisijan olisi hyvä selvästi ilmaista lisenssi tiedon käytölle [HeB11]. Valitettavasti näin ei käytännössä aina tapahdu, jolloin tiedon uudelleenkäytöstä voi olla epäselvyyksiä.

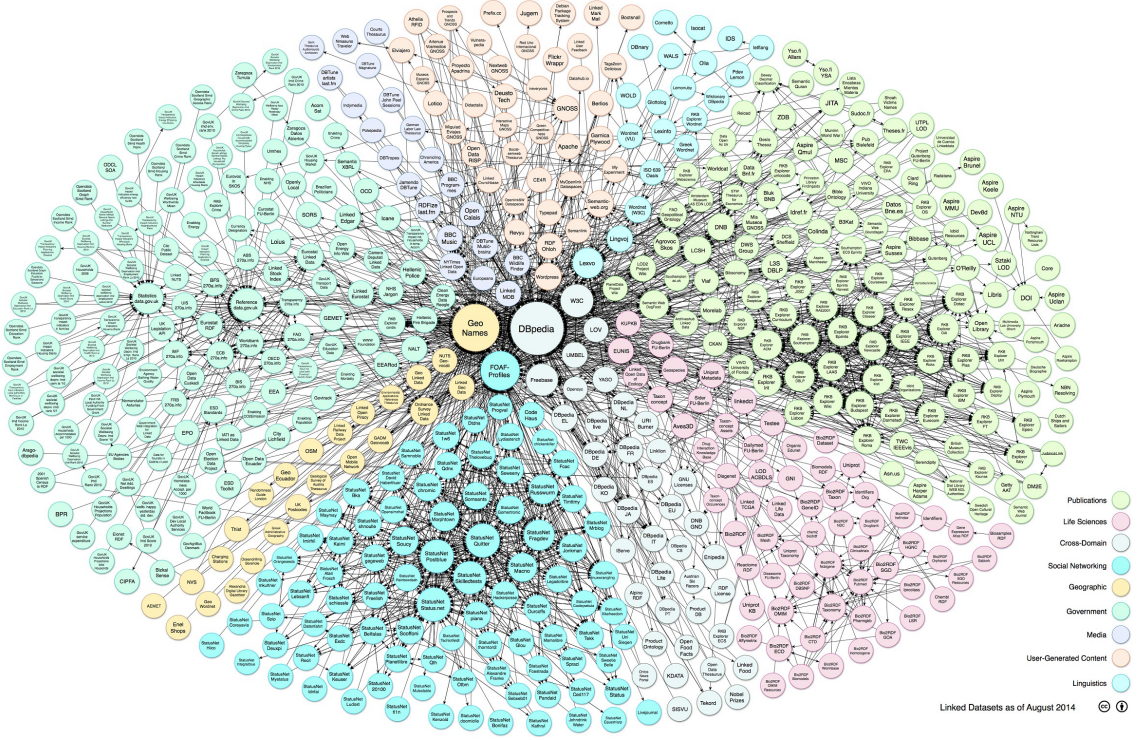
Kuka tahansa voi julkaista verkossa avointa dataa. Yleensä julkaistut tietojoukot ovat joko jotain tiettyä alaa, kuten biotieteitä, tai useita sovellusalueita yhdistelevää yleistä tietoa [HeB11]. Muun muassa juurikin biotieteiden ala on merkittävä linkitetyn avoimen datan julkaisija. Esimerkiksi DrugBank on kattava lääkkeiden ja niihin liittyvän tiedon tietopankki, joka linkittyy useisiin muihin tietolähteisiin [WKG06]. Myös viranomaiset ja muut julkiset tahot julkaisevat nykyään paljon avointa dataa [HHM12, SOB12], jota käsitellään erikseen kappaleessa 5.3.

Yksi tunnetuimmista usean sovellusalueen semanttisen tiedon lähteistä verkossa on DBpedia [BLK09]. Sen tarkoituksena on kerätä Wikipediasta tietoa ja julkaista se RDF-muodossa, jotta tietoon voi kohdistaa semanttisia kyselyitä. DBpedia myös linkittää sisältämiään käsitteitä muihin tietojoukkoihin. Muita suosittuja tietolähteitä ovat muun muassa tekoälyä varten arkista yleistietoa keräävä OpenCyc [OPCYC] ja eri lähteistä tietoa poimiva Yago [SKW07].

Usealta sovellusalueelta tietoa yhdistelevät tietolähteet, kuten DBpedia, ovat keskeisessä asemassa datan linkittämiseksi yhdeksi globaaliksi tietoavaruudeksi

eristyneiden saarekkeiden sijaan [Biz09]. Näiden tietolähteiden laajat aihealueet mahdollistavat tiedon tarjoajien linkittävän oman datansa esimerkiksi DBpediassa oleviin käsitteisiin. Näin usean sovellusalueen tietolähteistä muodostuu linkitetyn datan solmukohtia, jotka sitovat eri alueisiin keskittyneitä tietojoukkoja yhteen.

Vuonna 2014 suoritetussa linkitetyn datan verkon indeksoinnissa löytyi yhteensä 1091 tietojoukkoa [SBP14]. Verkon kasvuvauhti on nopea, sillä tietojoukkojen määrä on noin kaksinkertaistunut vuoteen 2011 verrattuna. Lisäksi noin 56 % tietojoukoista sisälsi vähintään yhden linkin muihin tietojoukkoihin. Kuva 2 havainnollistaa yli tuhat RDF-kolmikkoa sisältävien tietojoukkojen määrää ja yhteyksiä sovellusaluein jaoteltuna [SBJ14].



Kuva 2: Merkittävät verkon linkitetyt tietojoukot ja niiden yhteydet elokuussa 2014 [SBJ14].

5.3 Julkisen sektorin avoin data

Valtiot ja muu julkinen sektori tuottavat nykyään ympäri maailmaa merkittävästi avointa dataa [FIDAT, UKDAT, USDAT, PKDAT]. Datan avaaminen edistää hallinnon läpinäkyvyyttä ja edesauttaa tietoon perustuvien innovaatioiden syntyä [ShO13]. Usein data julkaistaan linkitetyn datan kannalta vähemmän hyödyllisissä formaateissa, kuten

PDF-dokumentteina tai Excel-taulukkoina, vaikka suurimman mahdollisen hyödyn tiedosta saisi koneellisesti ymmärrettävässä RDF-muodossa [ShO13]. Julkinen sektori on merkittävä linkitetyn datan tuottaja, sillä vuonna 2014 sen tuottama data vastasi noin 18 %:n osuutta kaikesta verkosta löydetystä linkitetystä datasta [SBP14].

Yhdysvaltain hallinnon avoimen datan palvelu [USDAT] on suurin avointa dataa julkaiseva julkisen sektorin sivusto [HHM12]. Sivusto sisältää laidasta laitaan erilaista kansalaista kiinnostavaa tietoa, kuten tilastoja paikallisen hallinnon menoista ja myrkyllisten jätteiden kaatopaikkojen sijainteja. Vaikka suuri osa tiedosta ei ole konellisesti ymmärrettävässä muodossa, Yhdysvaltain hallinto kuitenkin pyrkii edistämään avoimen datan julkaisua RDF-muodossa [HHM12]. Vuonna 2012 sivuston data sisälsi noin 6 miljardia RDF-kolmikkoa. Lisäksi sivustolla on saatavissa useita kansalaisten ja yritysten kehittämiä dataa hyödyntäviä sovelluksia, joista suurin osa on ilmaisia.

Myös Iso-Britannian hallinto on sitoutunut julkaisemaan dataansa koneellisesti ymmärrettävässä muodossa [SOB12]. Sen avoimen datan sivusto sisältää esimerkiksi tietoa tieturvallisuudesta ja kotitalouksien energiankäytöstä [UKDAT]. Sivuston tietojoukoista noin kymmenesosa oli vuonna 2016 saatavilla RDF-muodossa. Kuten Yhdysvaltain avoimen datan sivustolla, myös Iso-Britannian sivustolla on saatavilla useita dataa hyödyntäviä sovelluksia.

Myös Suomessa julkishallinto julkaisee verkossa avointa dataa [FIDAT]. Palvelun tavoitteena on edistää hallinnon läpinäkyvyyttä, tukea palveluiden kehittämistä, edistää julkisen hallinnon toimintojen yhteentoimivuutta ja tukea liiketoimintaa. Lisäksi esimerkiksi pääkaupunkiseudun kunnilla on oma palvelunsa avoimen datan julkaisuun verkossa [PKDAT].

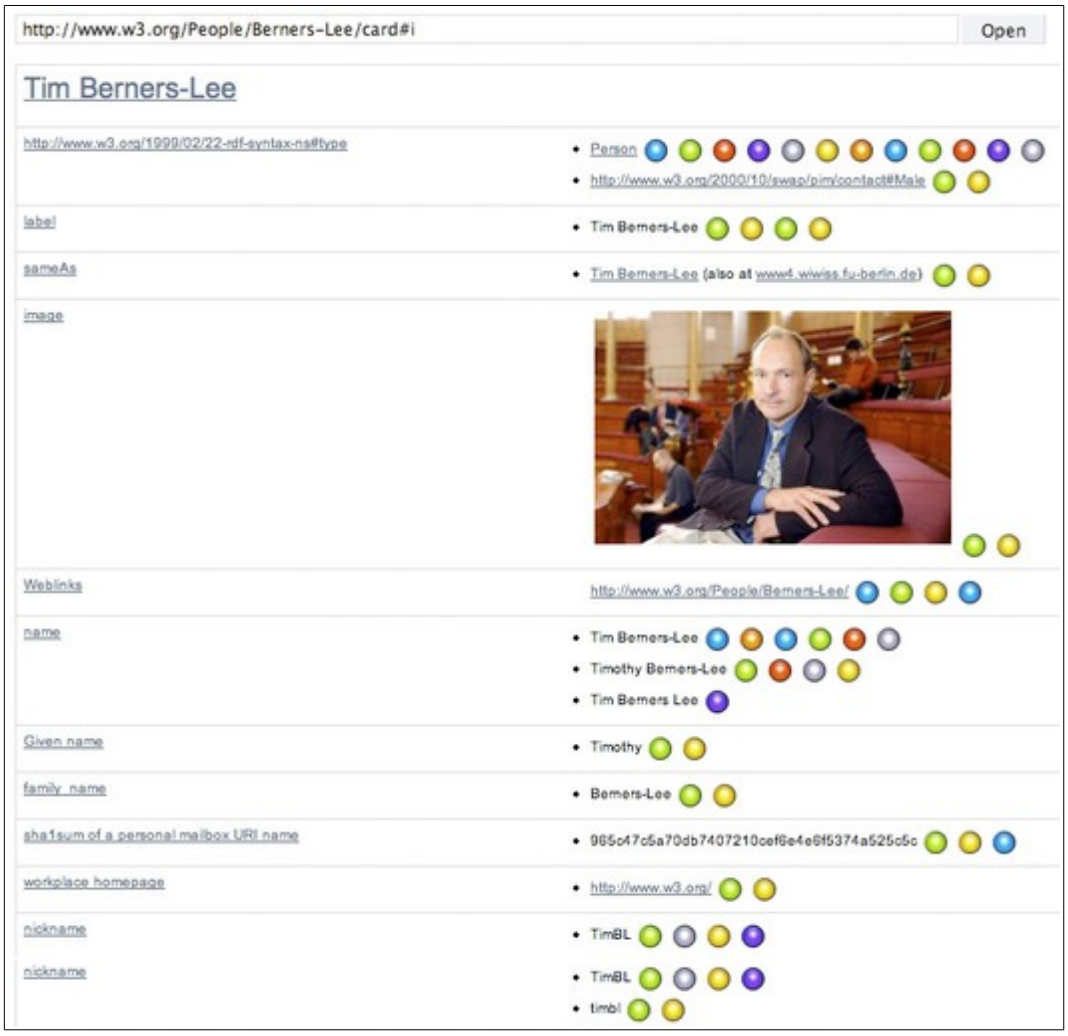
5.4 Linkitetyn datan hyödyntäminen

Globaalin dataverkon hyödyntäminen perustuu linkitetyn datan periaatteiden noudattamiseen [HeB11]. Verkossa esiintyvät käsitteet voidaan tunnistaa globaalisti URI-tunnisteiden ansiosta ja niistä saadaan lisätietoa tunnisteiden ollessa HTTP-tyyppisiä. Lisäksi tieto on linkittynyt muihin tietojoukkoihin, jolloin linkkejä seuraamalla löydetään uutta tietoa.

Linkitetty data on luonteeltaan avointa ja mahdollistaa uusien tietolähteiden löytämisen

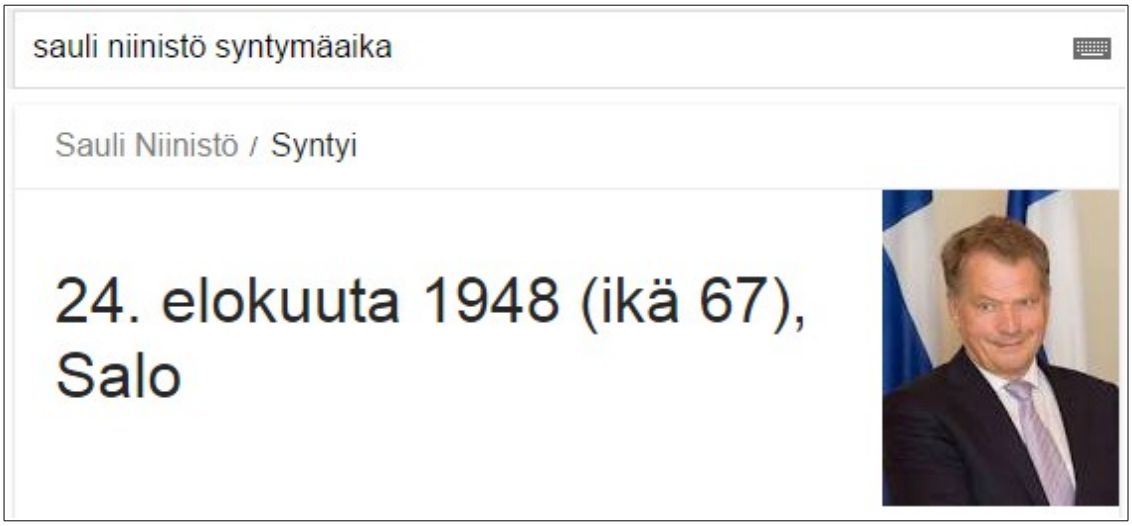
ajonaikaisesti. Sovellukset voivat siis automaattisesti käyttää hyväkseen linkitetyn datan verkossa julkaistua uutta dataa ilman erillisiä konfiguraatioita tai koodimuutoksia seuraamalla datassa olevia linkkejä muihin tietolähteisiin ja liittämällä niistä löydetty tieto yhteen [HeB11].

Perinteisen dokumenteista koostuvan verkon selaamiseen käytetään web-selaimia. Myös linkitetyn datan verkkoa voidaan yksinkertaisimmillaan käyttää hyväksi selaamalla sitä dataselaimella, joka dokumenttien sijasta näyttää URI-tunnisteella yksilöidyn käsitteen tietoa ja linkkejä muihin tietojoukkoihin [BCC06]. Dataselain voi myös hakea linkkien perusteella käsitteestä lisätietoa muista lähteistä. Esimerkiksi Tabulator [BCC06] ja Marbles [MARBL] ovat tunnettuja linkitetyn datan selaimia. Kuvassa 3 on Marbles-selaimen eri lähteistä linkkejä seuraamalla löytämä tieto Tim Berners-Leestä [BBH09]. Eri väriset täplät kertovat, mistä lähteistä tieto löydettiin.



Kuva 3: Marbles-selaimen löytämä tieto Tim Berners-Leestä [BBH09].

Linkitetyn datan verkkoa voi selainten lisäksi tutkia myös hakukoneilla [Biz09]. Linkitetyn datan hakukoneet vaihtelevat yksinkertaisista dataa indeksoivista roboteista [OCD08] luonnollista kieltä muistuttaviin kyselyihin vastaaviin hakukoneisiin [Har10]. Myös tunnetut hakujätit, kuten Google ja Yahoo, ovat alkaneet lisäämään hakukoneisiinsa semanttista tietoa hyödyntäviä ominaisuuksia pelkkien relevanttien web-sivujen listaamisen sijaan [Biz09]. Kuva 4 havainnolistaa Googlen hakuominaisuutta, jolla se osaa kertoa esimerkiksi henkilöistä yksinkertaisia faktoja. Haussa etsittiin Sauli Niinistön syntymäaikaa, jonka Google osasikin ilmoittaa suoraan. Googlen tekniikka perustuu Knowledge Vault -järjestelmään, joka kerää tietoa niin semanttisista lähteistä kuin myös muusta verkon sisällöstä koneoppimista hyödyntäen [DGH14].



Kuva 4: Google-kysely Sauli Niinistön syntymäajasta.

Linkitettyä dataa voi myös hyödyntää osana erilaisia sovelluksia [Hau09]. Esimerkiksi DBpedia Mobile [BeB09] on mobiililaitteille ja selaimille saatavissa oleva sovellus, joka käyttäjän sijainnin perusteella etsii semanttisesta webistä tietoa ympäröivästä alueesta ja näyttää löydetyn tiedon kartalla. Löydetyistä resursseista saa myös lisätietoa valitsemalla ne. Tiedon näyttämisen lisäksi sovelluksella voi lähettää verkkoon esimerkiksi kuvia tai ravintola-arvosteluita. Pelkän koordinaateilla merkitsemisen sijaan ladattuihin resursseihin lisätään automaattisesti tieto lähellä olevista DBpedia-resursseista.

Julkisen sektorin julkaisemaa avointa dataa hyödyntää esimerkiksi Yhdysvaltain ja Iso-

Britannian ulkomaanapua vertaileva sovellus [FRAID]. Se kerää yhteen molempien maiden julkaiseman RDF-muotoisen datan ulkomaanavusta ja mahdollistaa tiedon vertailun kartan ja graafien avulla. Sovellus myös hakee maihin liittyviä uutisartikkeleita ja faktatietoa.

6 Semanttisen webin sanastot

Semanttisen webin sisältämän tiedon tehokas hyödyntäminen vaatii sanastoja ja ontologioita [MaS01]. Sanastot ja ontologiat mallintavat tietoa ja siihen liittyvien käsitteiden ominaisuuksia ja suhteita formaalilla semantiikalla, minkä ansiosta älykkäät sovellukset voivat tehdä johtopäätöksiä webissä esiintyvästä tiedosta. Yhteistä ontologiaa tai sanastoa käytettäessä sovellukset myös ymmärtävät toisiaan ja eri lähteissä olevaa tietoa on helppo yhdistää.

Termien ”sanasto” ja ”ontologia” välillä ei ole selvää rajausta [CJB99]. Tässä tutkielmassa ontologialla tarkoitetaan formaalimpaa käsitteiden välisiä suhteita sisältävää rakennetta, kun taas sanasto on yleisempi termi kuvaamaan joukkoa sovellusaluetta mallintavia käsitteitä.

Semanttisen webin ontologiat ovat yleensä joko tietyn sovellusalueen ontologioita tai ylätason ontologioita [NiP01]. Ylätason ontologiat kuvaavat yleisiä usealla aihealueella käytössä olevia termejä. Sovellusalueen ontologiat taas keskittyvät mallintamaan tietyllä sovellusalueella esiintyviä käsitteitä ja niiden suhteita [MaS01].

Tässä kappaleessa tarkastelemme sanastojen ja ontologioiden merkitystä semanttisessa webissä. Lisäksi esittelemme muutaman suositun semanttisessa webissä käytetän sanaston tarkemmin ja tutkimme lääketietopankki DrugBankin [WKG06] käyttämiä sanastoja.

6.1 Sanastot semanttisessa webissä

Ontologiat ja sanastot ovat keskeisessä asemassa semanttisen webin hajallaan olevien ja itsenäisten tietojoukkojen sisältämän tiedon yhteentoimivuudessa [MaS01]. Ontologiat määrittelevät käsitteitä formaalisti ja kuvailevat käsitteiden välisiä suhteita. Tämä tekee webissä olevasta tiedosta helpommin koneellisesti ymmärrettävää. Usein semanttisen webin tietojoukot käyttävät tietonsa kuvailuun useita sanastoja kerralla.

Kun useat tietojoukot käyttävät samaa sanastoa käsitteiden kuvailuun, niin niiden sisältämää tietoa voidaan helposti käyttää yhdessä [HeB11]. Webissä esiintyvän tiedon tulkitsemisen kannalta onkin tärkeää, että tietojoukot käyttävät sanastoinaan jo valmiiksi olemassa olevia ja yleisesti tunnettuja sanastoja suurimman mahdollisen

yhteentoimivuuden takaamiseksi. Tunnettuja sanastoja käyttämällä on suurempi todennäköisyys, että tietoa tulkitseva sovellus ymmärtää tiedon merkityksen. Vuonna 2014 linkitetyn datan tietojoukoista lähes kaikki (99,87 %) käyttivät yleisiä sanastoja, kun taas omia sanastoja sisälsi noin 23 % tietojoukoista [SBP14]. Omalla sanastolla tarkoitetaan sanastoa, jota käyttää vain yksi tietojoukko.

Esimerkiksi jos kaksi tietojoukkoa sisältää tietoa ihmisistä, niin voivat ne molemmat käyttää tiedon kuvailuun myöhemmin tarkemmin esiteltävää FOAF-sanastoa [BrM14]. Yhteistä sanastoa käyttämällä niihin voidaan kohdistaa molempien sisältämää tietoa käyttäviä kyselyjä, koska sanasto antaa tiedon osasille, kuten etunimelle tai sähköpostiosoitteelle, koneellisesti ymmärrettävän merkityksen. Lisäksi samaa sanastoa ymmärtävät sovellukset voivat yhdistää niiden tietoa saumattomasti sanaston määritellessä yhteiset termit.

Käytetyimpiä semanttisen webin sanastoja ovat muun muassa FOAF ja Dublin Core [SBP14]. Vuonna 2014 FOAF-sanastoa käytti noin 69 % tietojoukoista ja Dublin Core -sanastoa noin 56 %. Taulukossa 4 on listattu suosituimmat sanastot ja kuinka monta prosenttia verkon linkitetyistä tietojoukoista käytti niitä.

Lyhenne	Osoite	Käyttöaste
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	98,22 %
rdfs	http://www.w3.org/2000/01/rdf-schema#	72,58 %
foaf	http://xmlns.com/foaf/0.1/	69,13 %
dcterm	http://purl.org/dc/terms/	56,02 %
owl	http://www.w3.org/2002/07/owl#	36,49 %

Taulukko 4: Semanttisen webin käytetyimmät sanastot vuonna 2014.

6.2 Sanastojen säilytys

Koska semanttisen webin sanastot julkaistaan verkossa, täytyy sanastoilla myös olla niihin viittaava URL-osoite. Sanastoa julkaistaessa on tärkeää valita pysyvä ja looginen osoite. Jos sanaston osoite muuttuu julkaisun jälkeen, niin mikään vanhaan osoitteeseen viittaava tietojoukko ei enää löydä sanaston määritelmiä. Tästä syystä sanastoa julkaistaessa olisi hyvä myös välttää metatiedon, kuten versionumeron, sisällyttämistä URL-osoitteeseen. Esimerkiksi FOAF-sanastoon viitataan osoitteella

http://xmlns.com/foaf/0.1/, vaikka sanasto ei enää ole versiossa 0.1. Osoitteen muuttaminen ei kuitenkaan ole enää mahdollista, sillä se aiheuttaisi häiriöitä sanaston lukuisille käyttäjille. FOAF-spesifikaatio toteaakin tämän olevan varoittava esimerkki kaikille, jotka harkitsevat ylimääräisen tiedon sisällyttämistä sanastonsa osoitteeseen [BrM14].

Mahdollisia vaihtoehtoja sanastojen säilytykseen ovat oman verkkotunnuksen lisäksi muun muassa W3C:n w3.org, xmlns.com (*XML Namespace*) ja purl.org (*Persistent Uniform Resource Locator*). Xmlns.com on FOAF-projektin yhteydessä perustettu palvelu muuttumattomien osoitteiden tarjoamiseksi sanastoille [XMLNS]. Palvelua ylläpitää yksityishenkilö. Purl.org on uudelleenohjauspalvelu muuttuville web-resursseille, jota käyttämällä sanaston osoite voi muuttua, kunhan pysyvä uudelleenohjausosoite (esimerkiksi *http://purl.org/dc/terms/*) päivitetään osoittamaan uuten osoitteeseen [PURL]. Palvelua ylläpitää voittoa tavoittelematon Online Computer Library Center -järjestö.

Selvittääksemme semanttisen webin sanastojen säilytyskäytäntöjä tutkimme vuonna 2014 tehdyn verkon linkitettyjen tietojoukkojen lapikäynnin tuloksia [SBP14]. Yhteensä käytettyjä sanastoja löytyi 135 kappaletta. Niihin viittaavista URL-osoitteista noin 47 % käytti purl.org-palvelua, noin 33 % W3C:n omia w3.org-sivuja ja noin 2 % xmlns.com-palvelua. Loput, eli noin 17 %, käyttivät sekalaisia omia verkkotunnuksiaan. Purl.org ja w3.org ovat siis selvästi suosituimmat osoitemuodot sanastojen säilytykseen. Yhteensä sanastoja säilytettiin kuudentoista eri verkkotunnuksen alla.

Sanastojen säilyttämiselle ei tällä hetkellä ole selkeää vallitsevaa standardia ja käytännöt vaihtelevat sanastojen välillä. Esimerkiksi Purl.org-palvelun toiminnan loppuminen aiheuttaisi suurta vahinkoa semanttiselle webille, ellei sitä siirrettäisi hallitusti toiselle ylläpitäjälle. Nykyisten suosittujen sanastojen osoitteita ei voi enää vaihtaa aiheuttamatta vahinkoa niiden käyttäjille, mutta uusien sanastojen säilyvyyden varmistamiseksi olisi hyvä luoda standardi sanastojen säilyttämiselle. Luonteva taho sen toteuttamiseen olisi W3C, sillä semanttista webiä toteutetaan järjestön standardien perusteella.

6.3 RDFS

RDFS (*RDF Schema*) mahdollistaa RDF-muotoiselle tiedolle yksinkertaisen kuvailun ja

luokittelun [BrG14]. RDFS laajentaa RDF:ää pienellä joukolla luokkia ja ominaisuuksia tiedon mallintamista varten. RDFS on W3C:n suosittelema teknologia, ja sen ensimmäinen versio julkaistiin jo vuonna 1998.

RDFS ei ole itsessään kovin ilmaisuvoimainen sanasto, mutta sitä käytetäänkin erityisesti muiden sanastojen määrittelemiseen ja pohjana voimakkaammille sanastoille [BrG14]. RDFS luo perusteet sanastojen luomiselle luokka- ja ominaisuusmäärityksillä, jotka muistuttavat olio-ohjelmoinnista tuttua tyyppijärjestelmää. Siinä missä olio-ohjelmoinnissa määritellään luokka ja sen sisältämät ominaisuudet, RDFS:ssä sen sijaan määritellään ominaisuudet ja mihin luokkiin ne voivat kuulua. Tämän ansiosta RDFS-sanastoja on helppo laajentaa muokkaamatta alkuperäisiä määrityksiä.

Luokkia määritellään RDFS:ssä *rdfs:Class*-määrityksellä [BrG14]. Luokista luodaan instansseja RDF:stä perityn *rdf:type*-määrityksen avulla. Sillä kerrotaan, että joku on jonkin luokan instanssi, esimerkiksi *timi rdf:type ihminen*. Luokkia ja niistä luotavia instansseja kuvaillaan ominaisuuksien avulla. Ominaisuuksia ovat muun muassa *rdfs:subClassOf* ja *rdfs:subPropertyOf*, jotka määrittelevät aliluokkia ja aliominaisuuksia. Ominaisuudella *rdfs:seeAlso* voi kertoa jonkin resurssin tarjoavan mahdollisesti hyödyllistä lisätietoa kuvailtavasta resurssista. Kyseisellä ominaisuudella on merkittävä rooli semanttisen webin tietojoukkojen yhteenlinkittäjänä [SBP14].

Ominaisuudella *rdfs:domain* voidaan määritellä jonkin ominaisuuden omaavan instanssin kuuluvan tiettyyn luokkaan [BrG14]. Sen avulla voisi esimerkiksi määritellä, että kaikki ominaisuuden sosiaaliturvatunnus omaavat kuuluvat luokkaan ihminen. Ominaisuudella *rdfs:range* voi taas määritellä, että jonkin kuvailtavan asian arvo kuuluu tiettyyn luokkaan. Sillä voi esimerkiksi määritellä, että jos jotain instanssia kuvaillaan jonkun työnantajaksi, niin se kuuluu luokkaan yhtiö.

Vuonna 2014 RDFS oli semanttisen webin toiseksi käytetyin sanasto heti RDF:n itsensä jälkeen [SBP14]. Sitä käytti noin 73 % linkitetyn datan tietojoukoista.

6.4 FOAF

FOAF (*Friend of a Friend*) on suosittu webissä käytetty yksinkertainen sanasto [BrM14]. Se kuvaa ihmisiä, heidän ominaisuuksiaan, kiinnostuksiaan ja suhteitaan RDF:n avulla. Sanaston tarkoituksena on mahdollistaa sosiaalisten verkostojen kuvailu koneellisesti ymmärrettävästi. Sitä voi käyttää sanastona sosiaalisten verkostojen

applikaatioissa tai yleisesti ihmisten suhteiden kuvaamiseen tietojoukoissa.

FOAF on yksi semanttisen webin käytetyimmistä sanastoista [SBP14]. Vuonna 2014 sitä käytti noin 69 % linkitetyn datan tietojoukoista. Käytetympiä sanastoja olivat vain RDFS ja RDF itse.

FOAF-sanastoa on kehitetty 2000-luvun puolivälistä lähtien [BrM14]. Sille on vakiintunut pysyvien luokkien ja ominaisuuksien ydin, mutta siihen voidaan lisätä tarpeen vaatiessa uusia termejä. FOAF voi siis kehittyä hiljalleen luonnollisen kielen sanakirjojen tapaan. FOAF on myös suunniteltu käytettäväksi yhdessä muiden sanastojen kanssa.

FOAF-sanaston määrittelemiä luokkia ovat muun muassa agentti, henkilö ja organisaatio [BrM14]. Henkilö ja organisaatio ovat agentin aliluokkia. Sanaston määrittelemien ominaisuuksien avulla voidaan kertoa luokkien instansseista lisätietoa. Ominaisuuksia ovat esimerkiksi sähköpostiosoite, nimi ja kotisivu. FOAF sisältää myös *knows*-ominaisuuden henkilön tuttavuuksien määrittelemiseen.

FOAF:n käyttö yhtenä monesta sanastosta suuressa tietojoukossa on yleistä. Sen lisäksi sillä voi luoda itseään kuvailevan FOAF-tiedoston [BrM14]. Tiedostossa voi kertoa itsestään tietoa harkintansa mukaan sanaston avulla. Tiedoston linkittyessä muihin vastaaviin FOAF-tiedostoihin syntyy hajautettu semanttinen sosiaalinen verkosto. Esimerkissä 12 on yksinkertainen näyte tiedon kuvailemisesta FOAF-sanaston avulla Turtle-muodossa. Siinä määritellään resurssin *http://esim.fi/timi* olevan henkilö ja kerrotaan nimi, kotisivu sekä kuvan osoite.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<http://esim.fi/timi>
  a foaf:Person ;
  foaf:name "Timi" ;
  foaf:homepage <http://esim.fi/timi> ;
  foaf:img <http://esim.fi/timi/kuva.jpg> .
```

Esimerkki 12: Esimerkki tiedon kuvailusta FOAF-sanaston avulla Turtle-muodossa.

6.5 Dublin Core

Dublin Core on Dublin Core Metadata Initiativen (DCMI) ylläpitämä suosittu sanasto [DBLCR]. Dublin Corea käytetään kuvailemaan monenlaisia resursseja, kuten web-

sivuja tai fyysisiä teoksia. Dublin Core on yksi semanttisen webin suosituimmista sanastoista [SBP14].

Dublin Core -sanasto sisältää 15 alkuperäistä termiä (*Dublin Core Metadata Element Set*) ja joukon myöhemmin lisättyjä määrityksiä [DBLCR]. Alkuperäiset 15 termiä ovat *title* (otsikko), *creator* (tekijä), *subject* (aihe), *description* (kuvaus), *publisher* (julkaisija), *contributor* (osallistuja), *date* (päivämäärä), *type* (tietotyyppi), *format* (formaatti), *identifier* (tunniste), *source* (lähde), *language* (kieli), *relation* (yhteys), *coverage* (kattavuus) ja *rights* (oikeudet). Myöhemmin on lisätty muun muassa tiivistelmää tarkoittava *abstract.* Sanasto määrittelee myös *type*-ominaisuudelle mahdolliset tietotyypit, joita ovat esimerkiksi *software* (ohjelmisto), *image* (kuva) ja *physicalObject* (fyysinen esine). Esimerkissä 13 on näyte tiedon kuvailusta Dublin Core -sanastolla Turtle-muodossa.

```
@prefix dc: <http://purl.org/dc/terms/> .
<http://esim.fi/kirjat/esimerkkikirja>
  dc:title "Esimerkkikirja" ;
  dc:description "Kuvaus Esimerkkikirjasta." ;
  dc:creator <http://esim.fi/timi> .
```

Esimerkki 13: Esimerkki tiedon kuvailusta Dublin Core -sanastolla Turtle-muodossa.

Vuonna 2014 Dublin Core -sanastoa käytti noin 56 % semanttisen webin tietojoukoista [SBP14]. Yhteensä se oli neljänneksi suosituin heti FOAF-sanaston jälkeen. Tunnettuja käyttäjiä ovat muun muassa DBpedia ja DrugBank.

6.6 DrugBank

DrugBank on erittäin laaja lääkkeiden ja niihin liittyvän tiedon tietokanta, johon sisältyy yli 8000 lääkettä [WKG06]. Se sisältää yksityiskohtaista tietoa lääkkeiden kemiallisista ja farmaseuttisista ominaisuuksista. DrugBankia käytetään laajasti apuvälineenä terveydenhuoltoalalla.

DrugBank on vapaasti kenen vain käytettävissä ja kaikki sen sisältämä data on avointa [WKG06]. Se tarjoaa tietopankin selaamiseen käyttöliittymän ja joukon hakutoimintoja eri tarkoituksia varten. DrugBank on myös saatavilla semanttisena RDF-versiona osoitteessa *http://wifo5-03.informatik.uni-mannheim.de/drugbank/*.

Tässä kappaleessa tutkimme, mitä sanastoja DrugBank-tietopankin semanttinen versio käyttää resurssiensa kuvailemiseen. Käytetyt sanastot selvitämme selaamalla tietopankin RDF-version resursseja ja tutkimalla niihin linkitettyjä sanastoja.

DrugBankin laajimmin käyttämä sanasto on sen oma *drugbank*-sanasto. Se sisältää laajasti lääkkeisiin ja niiden vaikutuksiin liittyviä termejä. Sanastossa määritellään esimerkiksi lääkkeiden tuotemerkki (*drugbank:brandName*), tietoa myrkyllisyydestä (*drugbank:toxicity*) ja tietoa vuorovaikutuksista ruoka-aineiden kanssa (*drugbank:foodInteraction*). Lisäksi määritellään kemiallisia termejä, kuten kemiallinen kaava (*drugbank:chemicalFormula*). Suurin osa lääkkeiden ominaisuuksista on ilmaistu oman *drugbank*-sanaston avulla.

DrugBank käyttää myös FOAF-sanastosta *foaf:page*-määritystä. Sillä viitataan sivuun tai dokumenttiin, joka liittyy kuvailtavaan resurssiin [BrM14]. DrugBankissa määritystä käytetään linkittämään lääkkeet niitä vastaaviin sivuihin Wikipediassa, DBpediassa ja DrugBankin ihmisille tarkoitetussa käyttöliittymässä.

OWL-sanastosta DrugBank käyttää *owl:sameAs*-määritystä, jolla kuvaillaan samaa tarkoittavaa resurssia [HKP12]. Sen avulla DrugBankin termit liitetään muun muassa DBpedian vastaaviin termeihin. Ero *foaf:page*-määrityksen käyttämiseen on se, että *owl:sameAs* viittaa semanttiseen resurssiin, eikä siis esimerkiksi Wikipedia-sivuun. OWL-sanaston *owl:sameAs*-termi on tärkeä semanttisen webin tietojoukkojen yhteenlinkittymisen kannalta [SBP14]. Linkkiä seuraamalla kysely tai sovellus voi saada helposti lisää mahdollisesti relevanttia tietoa tarkasteltavasta resurssista.

RDFS-sanastosta käytetään ominaisuutta *rdfs:label.* Sillä tarkoitetaan ihmisen ymmärrettävää nimeä kuvailtavalle asialle [BrG14]. DrugBankissa sillä kerrotaan lääkkeistä niiden yleinen nimi.

Päivämääriä ilmaisemaan DrugBank käyttää Dublin Core -sanastosta päivämäärää tarkoittavaa *dc:date*-termiä ja oman *drgubank*-sanastonsa termejä. Dublin Coren *dc:date*-ominaisuutta käytetään ilmaisemaan näytettävän tiedon hakuaikaa. Tiedon varsinaisen lisäys- ja muokkauspäivän ilmaisemiseen käytetään kuitenkin DrugBankin oman sanaston *drugbank:creationDate* ja *drugbank:updateDate* -termejä.

DrugBank käyttää myös VoID-sanastoa (*Vocabulary of Interlinked Datasets*). VoID on tarkoitettu metatiedon välittämiseen linkitetyistä tietojoukoista [ACH11]. Sitä voidaan

käyttää esimerkiksi tietojoukkoon liittyvän SPARQL-päätepisteen tai tietojoukon kolmikkojen määrän ilmoittamiseen. DrugBank käyttää sanastosta *void:inDataset*-termiä ilmoittamaan kaikkien sen sisältämien resurssien kuuluvan DrugBank-tietojoukkoon.

Prv-sanastoa (*Provenance Vocabulary)* käytetään DurgBankissa ilmaisemaan tiedon alkuperää. Prv-sanaston käyttö auttaa linkitetyn datan julkaisijaa ilmoittamaan julkaistun datan alkuperän ja linkitetyn datan kuluttajan verkosta hakeman tiedon alkuperän ymmärtämistä [HaZ12]. DrugBank käyttää sanaston *prv:containedBy*-termiä ilmaisemaan sen resurssien sisältyvän DrugBank-tietojoukkoon.

Yhteensä DrugBank käyttää yleisistä sanastoista noin kuutta termiä, kun taas omaa sanastoa käyttäviä termejä on useita kymmeniä. On valitettavaa, että DrugBank käyttää useimmiten termiensä määrittelemiseen omaa sanastoaan, joka sisältää sellaisiakin yleisiä termejä, kuten kemiallinen kaava, lisäyspäivämäärä ja päivityspäivämäärä. Yhteentoimivuuden näkökulmasta olisi toivottavaa, että tietojoukot käyttäisivät omia sanastojaan vain sellaisten termien määrittelemiseen, joita ei löydy yleisesti käytetyistä sanastoista. Omien sanastojen käyttö yleisten sanastojen sijaan vaikeuttaa tiedon uudelleenkäyttöä ja sen merkityksen koneellista ymmärtämistä. Ongelmana saattaa olla se, että vielä ei ole kehittynyt tarpeeksi tarkoituksenmukaisia sanastoja semanttisen webin tarpeisiin, joten tietojoukot turvautuvat käyttämään omia sanastojaan.

7 Hajallaan olevan RDF-muotoisen tiedon hallinta

Lukuisista itsenäisistä tietolähteistä muodostuvan linkitetyn datan hajanainen luonne ja tiedon suuri määrä tuottavat haasteita RDF-muotoisen tiedon hallintaan [FCO12]. Verkossa oleva tieto ei noudata yhtä globaalia mallia, joten tiedon yhteensovittamiseen tarvitaan joko yhteisiä ontologioita, linkkejä tietojoukkojen käsitteiden välille tai eri ontologioiden yhteensovittamista [LaS12].

Suora tiedonhaku useista webissä olevista RDF-tietolähteistä tuli osaksi SPARQL-standardia vasta version 1.1 myötä vuonna 2013 [PrB13], vaikka sovelluksia sille oli olemassa jo aiemminkin [QuL08]. Käytännössä usean lähteen kyselyt tapahtuvat usein alikyselyinä tietolähteiden julkaisemiin SPARQL-päätepisteisiin, minkä jälkeen saadut tulokset kootaan yhteen [PrB13]. On myös mahdollista yhdistää tietokokoelmia ennalta yhdeksi ja tehdä siihen normaali SPARQL-kysely [Ala11].

Hajallaan olevaan RDF-muotoiseen tietoon on myös mahdollista tehdä kyselyitä määrittelemättä tai tietämättä kaikkia tietolähteitä etukäteen, sillä tiedon kytkökset mahdollistavat ajonaikaisen lisätiedon etsimisen linkkejä seuraten [HBF09]. Tämä toiminto on yksi linkitetyn datan kiinnostavimmista käyttötarkoituksista, mutta siihen liittyy vielä useita haasteita [LaT10].

7.1 Ontologioiden yhteensovittaminen

Linkitetty data muodostuu erilaisia sanastoja käyttävistä itsenäisistä tietokokoelmista. Jotta kokoelmia voitaisiin käyttää tehokkaasti hyödyksi, tarvitaan tiedon integroimiseksi niiden välillä heterogeenisten ontologioiden yhteensovittamista (*ontology mapping, ontology alignment*) [MaS01]. Ontologioiden yhteensovittamista voidaan yleisen yhteentoimivuuden mahdollistamisen lisäksi käyttää hyödyksi esimerkiksi semanttisiin kyselyihin vastaamisessa [HeC06], ontologioiden evoluutiossa [NCL06] tai älykkäiden agenttien apuna [WRP05].

Yksinkertaisimmillaan ontologioita voidaan sovittaa yhteen määrittelemällä manuaalisesti suhteita kahden eri ontologian käsitteiden välillä. OWL-kielellä voi määritellä esimerkiksi kahden käsitteen tarkoittavan samaa tai jonkin käsitteen kuuluvan tiettyyn luokkaan toisessa ontologiassa [DeS04]. Yhteensovittamiseen on kuitenkin

myös kehitetty erilaisiin strategioihin perustuvia automaattisia apuvälineitä [LaT06, LTL09, HuQ08]. Tyypillisesti mikään strategia ei ole yksinään riittävän hyvä yhteensovittamisen toteuttamiseksi, vaan niitä käytetään useita yhdessä [ShE13].

7.1.1 Kieleen ja merkkijonoihin perustuvat strategiat

Yleisimpiä strategioita ontologioiden yhteensovittamiseen ovat kielen tai merkkijonojen vastaavuuteen perustuvat strategiat [LaT06]. Ne tutkivat ontologioista löytyvien käsitteiden ominaisuuksien tekstuaalisia kuvauksia ja laskevat niiden perusteella arvon käsitteiden vastaavuudelle. Yksinkertaisimmillaan kahdella käsitteellä voidaan todeta olevan sama nimi ja yhdistää ne. Käsitteiden nimiä tai kuvauksia voidaan tutkia myös muita merkkijonoalgoritmeja käyttäen, esimerkiksi vertaamalla niiden muokkausvälimatkaa (*edit distance*) [LTL09]. Termien vertailuun voidaan myös käyttää ulkopuolisia sanastoja niiden kielellisten yhteyksien, kuten synonyymien, löytämiseen [MLR10].

Strategian ongelmana ovat käsitteet, jotka voivat kontekstista riippuen tarkoittaa eri asioita. Esimerkiksi käsite ”kolmio” voi tarkoittaa joko geometrista kuviota tai kolmen huoneen asuntoa. Myös eri kielisten ontologioiden yhteensovittaminen on vaikeaa, koska kielellinen vertaileminen on hyödytöntä kääntämättä niitä ensin samalle kielelle.

7.1.2 Rakenteeseen perustuvat strategiat

Rakenteeseen perustuvat strategiat käyttävät hyväkseen vertailtavien ontologioiden käsitteiden rakenteita tutkiakseen samankaltaisuutta [LTL09]. Periaatteena on, että jos käsitteisiin liittyvät muut käsitteet ovat samankaltaisia, niin itse käsitteetkin ovat suuremmalla todennäköisyydellä vastaavia. Strategiassa voidaan esimerkiksi verrata käsitteiden ylä- ja aliluokkia ja laskea vastaavuus niiden perusteella.

7.1.3 Rajoitteisiin perustuvat strategiat

Rajoitteisiin perustuvat strategiat tarkastelevat käsitteille asetettuja rajoitteita ja laskevat yhtäläisyyden niiden vastaavuuden mukaan [LaT06]. Jos esimerkiksi vertailtavat käsitteet ovat molemmat todettu olevan saman käsitteen vastakohtia, niin kasvaa niiden vastaavuuden todennäköisyys.

7.1.4 Sovellukset

SAMBO (*System for Aligning and Merging Biomedical Ontologies*) on erityisesti lääketieteen alan ontologioiden yhteensovittamiseen kehitetty sovellus [LaT06]. Se tukee ainoastaan OWL-ontologiakieltä. Järjestelmä tutkii kahden ontologian käsitteiden samankaltaisuutta joukolla eri strategioita käyttäviä täsmääjiä (*matcher*). Täsmääjien tulokset yhdistetään ja niiden perusteella tehdään ehdotukset yhteensovittamisesta. Käyttäjä voi joko hyväksyä tai hylätä ehdotukset.

Falcon on OWL-kieltä tukeva graafisen käyttöliittymän sisältävä järjestelmä ontologioiden yhteensovittamiseen [HuQ08]. Käyttöiittymän lisäksi siihen kuuluu Falcon-AO-sovellus, joka hoitaa ontologioiden käsitteiden vastaavuuksien määrittelyn. Myös se hyödyntää yhteensovittamisessa useita eri strategioita, muun muassa kieleen- ja rakenteeseen perustuvia tekniikoita.

RiMOM on yksi parhaiten ontologioiden yhteensovittamista vertailevissa tutkimuksissa menestynyt järjestelmä [LTL09]. Myös se perustuu useiden eri strategioiden käyttämiseen käsitteitä sovitettaessa, mutta sen vahvuutena on juuri kyseessä olevaan tilanteeseen sopivimpien strategioiden valinta dynaamisesti kaikkien käyttämisen sijaan. Järjestelmä tekee aluksi aineistolle esikäsittelyn, jonka perusteella se valitsee käytettävät strategiat.

7.1.5 Haasteet

Ontologioiden yhteensovittamista on tutkittu jo vuosia, mutta aiheeseen liittyy vielä useita haasteita [ShE13]. Yksi suurimmista haasteista on ihmisten osallisuus prosessissa. Yleensä yhteensovittamisen tulokset täytyy hyväksyä manuaalisesti ihmisen toimesta, sillä teknologiat eivät ole niin kehittyneitä, että prosessi voisi olla täysin automaattinen. Tämä on kuitenkin ongelmallista, koska käsiteltävät tietojoukot voivat olla valtavia. Yleisesti ongelmaa lievittäisi tarkemmat ja tehokkaammat yhteensovittajat, jotta ihmisiä tarvittaisiin mahdollisimman vähän. Tulisi myös kehittää tapoja tehdä ihmisen osallitumisesta prosessiin mahdollisimman vaivatonta ja kätevää.

Yhteensovittamisen automatisoimiseksi on kehitetty useita strategioita ja yleensä sovittamisessa käytetäänkin useiden strategioiden yhdistelmiä. Jokainen strategia ei kuitenkaan sovi jokaiseen tilanteeseen [ShE13]. Oikeiden strategioiden valinta tiettyyn tilanteeseen on prosessin yksi haasteista.

Verkon avoin ja sosiaalinen luonne mahdollistavat yhteensovittamisen tekemisen jonkin yhteisön yhteistoimintana [ShE13]. Yhdelle henkilölle liian suuren työn voisi siis verkkoa käyttäen jakaa usealle ihmiselle. Tämän mahdollistamiseksi on kehitetty apuvälineitä, mutta tekniikka tarvitsee vielä lisätutkimusta [NCL06].

Infrastruktuuri yhteensovitusten tuloksien jakamiseen ja uudelleenkäyttöön tarvitsee myös vielä kehitystä [ShE13]. Tehokkaampi tulosten jakaminen mahdollistaisi helpomman uusiokäytön, eikä jokaisen tarvitsisi käyttää resursseja saman yhteensovittamisen tekemiseksi erikseen.

7.2 Useisiin lähteisiin kohdistetut kyselyt

Saatavilla olevien RDF-muotoisten tietokokoelmien suuri määrä ja keskinäinen linkitys mahdollistavat tiedon yhdistelemisen eri lähteistä. Hakemalla tietoa esimerkiksi lääkkeistä voi dataa yhdistellä DBpediasta [BLK09], lääketietokanta DrugBankista [WKG06] ja bioinformatiikan tietopankista KEGGistä [KEGG]. Näin ei tarvitse tyytyä vain yhden lähteen tuottamaan tietoon ja on mahdollista vastata monimutkaisiin kyselyihin, joihin vaaditaan monen tietolähteen yhdistämistä.

Tietokokoelmien kasvava määrä lisää tarvetta tehokkaille tavoille tehdä kyselyjä useisiin lähteisiin. Kyselyiden tekemiseen käytetyt tekniikat voidaan jakaa kolmeen luokkaan: keskitetty kysely [OCD08], federoitu kysely [BAC13] ja linkkien seuraaminen [HBF09]. Suurimmat erot tekniikoissa liittyvät kyselyiden tehokkuuteen ja tiedon tuoreuteen. Tehokkuuden ja tuoreuden varmistaminen on myös yleisesti hajallaan olevien kyselyiden tekemisen olennainen haaste [Ala11]. Myös eri tekniikoiden vahvuuksia yhdistelevää hybridiratkaisua on ehdotettu [UKH12].

7.2.1 Keskitetty kysely

Hajallaan olevan RDF-muotoisen tiedon keskitetty kysely (*centralised query*) tarkoittaa RDF-tietojoukkojen lataamista ennalta useista lähteistä ja yhdistämistä yhdeksi suureksi tietojoukoksi [Ala11]. Yhdistämisen jälkeen tietojoukkoon voidaan tehdä kyselyitä kuten mihin tahansa normaaliin RDF-joukkoon. Keskitetty tietojoukko voi olla paikallinen RDF-tietojoukko tai SPARQL-päätepisteen kautta käytettävä etätietovarasto. Menetelmä perustuu perinteisen tietovaraston konseptiin, jossa usean tietokannan tietoa kerätään yhteen keskitettyyn varastoon [ChD97].

Keskitetty kysely on yksinkertaisin tapa hakea tietoa useasta RDF-tietojoukosta ja sen mahdollistavia toteutuksia on olemassa useita, kuten Sindice [OCD08], Virtuoso [ErM09] ja FactForge [BKO11]. Esimerkiksi Sindice käy läpi tietojoukkoja, indeksoi niiden datan ja tarjoaa SPARQL-päätepisteen tiedonhakuun.

Keskitetyn kyselyn etuna on sen kyselyjen suorituskyky, sillä käytettävä data on valmiiksi ladattua, eikä sitä tarvitse hakea enää erikseen verkosta [Ala11]. Valmiiksi ladatun datan kääntöpuolena kuitenkin on datan tuoreus. Jos tietoa ei ole päivitetty tarpeeksi usein, voi seurauksena olla vanhentuneita vastauksia kyselyihin [UKH12]. Datan säilöminen paikallisesti yhdessä paikassa vaatii myös paljon tilaa. Lisäksi linkitetyn datan verkko on niin laaja, ettei kaikkea sen sisältämää tietoa ole realistista säilyttää yhdessä paikassa.

7.2.2 Federoitu kysely

Hajallaan olevan RDF-muotoisen tiedon federoidussa kyselyssä (*federated query*) kyselyn osien prosessoiminen ohjataan välittäjän (*query mediator, query federator*) kautta alikyselyinä itsenäisille tietolähteille yhden keskitetyn tietovaraston käyttämisen sijaan [BAC13]. Tietolähteet prosessoivat omat kyselyn osansa ja palauttavat vastaukset välittäjälle, joka yhdistää ne. Keskitetyn kyselyn tavoin myös federoitu kysely perustuu perinteisiin tietokantatekniikoihin [ShL90].

Useisiin lähteisiin kohdistetut kyselyt tulivat virallisesti osaksi SPARQL-standardia version 1.1 myötä [PrB13]. Ennen tätä se oli mahdollista käyttämällä epävirallisia laajennoksia SPARQL-standardiin [QuL08]. Nykyään useat linkitetyn datan tietokokoelmat tarjoavat julkisen SPARQL-päätepisteen, johon voidaan ohjata federoidun kyselyn alikysely.

SPARQL-kielessä federoituja kyselyitä tehdään *SERVICE*-komennon avulla [PrB13]. *SERVICE*-komento saa parametrina tiedon kyseltävästä tietojoukosta SPARQL-päätepisteen osoitteen muodossa. Se myös aloittaa uuden lohkon, jonka sisään tehdään kyseiseen päätepisteeseen kohdistuva alikysely. Tulokset yhdistetään, kun päätepisteet ovat palauttaneet vastaukset omiin alikyselyihinsä. Esimerkissä 14 on abstrakti esitys kahteen päätepisteeseen kohdistuvasta federoidusta kyselystä *SERVICE*-komentoa käyttämällä.

```
SELECT ?muuttujat WHERE {
   SERVICE <http://päätepiste1.esim/sparql> {
      ?määritykset
   }
   SERVICE <http://päätepiste2.esim/sparql> {
      ?määritykset
   }
}
```

Esimerkki 14: Abstrakti esimerkki SERVICE-komennon käytöstä.

Useisiin lähteisiin kohdistetun kyselyn toteuttamiseksi tarvitaan SPARQL-kyselymoottori, joka tukee federoitua kyselyä. Esimerkiksi Apachen Jena-ohjelmistokehys sisältää ARQ-kyselytyökalun, jolla federoituja kyselyitä voi tehdä komentorivikäyttöliittymän avulla [JENA].

Usean lähteen tieto on federoitua kyselyä käyttäessä yksinkertaisinta yhdistää, jos tietolähteet käyttävät käsitteen tunnistamiseen samaa URI-tunnistetta. Tietolähteet ovat myös saattaneet yhdistää käsitteet esimerkiksi OWL:n *sameAs*-määrityksellä. Lisäksi ulkopuolisen ontologian käyttäminen yhdistämiseen on mahdollista.

Federoidun kyselyn etuna on se, että tietoa ei tarvitse ladata ennalta, eikä se siis myöskään vie turhaa levytilaa [HaL10]. Lisäksi löydetty tieto on aina tuoretta, sillä se haetaan kyselyä suorittaessa ajonaikaisesti. Toisaalta tiedon haku verkosta voi olla hidasta, koska dataa täytyy siirtää mahdollisesti suuriakin määriä. Kyselyn suorituskyky riippuu myös aina käytettyjen päätepisteiden nopeudesta. Verrattuna linkkien seuraamiseen perustuvaan tekniikkaan federoidussa kyselyssä täytyy päätepisteet lisäksi tietää ja määritellä ennalta.

7.2.3 Linkkien seuraaminen

Keskitetty ja federoitu kysely pohjautuvat vanhoihin tietokantojen tiedonhakutekniikoihin, mutta linkitetyn datan verkko eroaa merkittävästi perinteisistä tietokannoista. Niiden kyselyt perustuvat ennalta määriteltyihin tietolähteisiin, vaikka verkon suuri koko ja avoimuus tarkoittavat, että on mahdotonta tietää ennalta kaikki lähteet, joista voi olla hyötyä kyselyä tehdessä [HBF09]. Linkitetyn datan tehokkaaseen hyödyntämiseen tarvitaan siis kyselytekniikoita, jotka ottavat huomioon sen erityisen luonteen.

Linkkien seuraamiseen (*link traversal*) perustuva kyselytekniikka on alun perin Berners-Leen ja kumppaneiden ehdottama tekniikka [BCC06], joka hyödyntää tietojoukkojen välisiä RDF-linkkejä etsiäkseen lisää mahdollisesti relevanttia tietoa kyselyyn vastatakseen. Näin kaikkia käytettäviä tietolähteitä ei tarvitse määritellä ennalta, vaan linkitetyn datan koko potentiaali saadaan valjastettua löytämällä uutta dataa ajonaikaisesti [HaF12].

Linkkien seuraamisen toteuttamiseksi käytännössä on erilaisia tyylejä [Har11, Sch11, LaT10], mutta perusperiaatteena on, että kyselyä tehdessä haetaan alkutiedot kyselyssä mainituista URI-osoitteista [HaF12]. Tämän jälkeen rakennetaan kyselyn vastausta sovittamalla yhteen kyselyssä määriteltyjä RDF-kolmikoita ja haetussa datassa esiintyviä kolmikoita. Jos kyselyyn ei löydy vastausta aluksi ladatuista tiedoista, niin seurataan niissä esiintyviä URI-tunnisteita ja haetaan taas lisätietoa. Näitä askeleita toistetaan, kunnes kyselyyn on löytynyt vastaus.

Linkkien seuraamisen tekniikan vahvuutena on sen potentiaali ennalta tuntemattoman tiedon löytämisessä [HaF12]. Se ei vaadi määriteltyjä tietokokoelmia eikä edes SPARQL-päätepistettä, sillä tiedonhaku tapahtuu täysin URI-tunnisteita seuraamalla. Sen toimintaan vaaditaan vain, että tieto noudattaa linkitetyn datan periaatteita.

Tekniikan suurin ongelma on suorituskyky. Mahdollisia tutkittavia tietokokoelmia voi olla valtava määrä, joten on olennaista pyrkiä valitsemaan mahdollisimman relevantit kokoelmat tutkittavaksi [Har11]. Kuten federoidussa kyselyssä, myös linkkejä seuraamalla ongelmaksi voi muodostua kyselyssä mukana olevien tietolähteiden suorituskyky ja tavoitettavuus.

7.2.4 Hybridiratkaisut

Useisiin lähteisiin kohdistettujen kyselyiden hybridiratkaisut pyrkivät yhdistämään eri kyselytapojen parhaita puolia. Umbrich ja kumppanit esittelevät vuonna 2012 artikkelissaan [UKH12] lähestymistavan, joka perustuu osittain keskitettyyn kyselymalliin ja osittain federoituun kyselymalliin. Ratkaisu pyrkii palauttamaan tuoreempia tuloksia kuin pelkkä keskitetty malli, mutta nopeammin kuin pelkkä federoitu malli.

Umbrichin ja kumppanien malli sisältää kyselysuunnittelijan, joka jakaa kyselyn tarvittaessa kahteen osaan [UKH12]. Kyselyn toinen osa suunnataan valmiina olevaan

keskitettyyn tietojoukkoon ja toinen verkossa sijaitsevaan päätepisteeseen. Mallissa ehdotetaan useita algoritmeja kyselyn jakokohdan määrittelyyn. Yksi mahdollisuus on käyttää aina määrättyä pistettä, kuten kyselyn puoliväliä. Toinen mahdollisuus on suorittaa kyselyn rajoittavimmat kolmikkoehdot verkossa sijaitsevaa päätepistettä vasten.

7.3 Koekyselyt

Tässä kappaleessa teemme kahteen hajallaan olevaan RDF-tietolähteeseen kohdistuvan federoidun koekyselyn ja linkkejä seuraamalla tapahtuvaa tiedonhakua havainnollistavan esimerkkikyselyn.

7.3.1 Federoitu kysely

Tässä esimerkissä haluamme kysellä hajallaan olevia RDF-tietolähteitä federoidulla SPARQL-kyselyllä ja saada tietoa tietystä lääkkeestä. Lääke on tässä kokeessa aspiriini. Lääkkeestä halutaan ymmärrettävä selväkielinen kuvaus ja tietoa sen myrkyllisyydestä. Tieto myrkyllisyydestä löytyy lääketietokanta DrugBankista [WKG06], mutta sen kuvaukset ovat lyhyitä ja tieteellisiä. Kattavan kuvauksen saa kuitenkin DBpediasta [BLK09], joten voimme tehdä näihin kahteen lähteeseen kohdistuvan federoidun kyselyn ja yhdistää niistä löydetyn tiedon.

Kyselyn suorittamiseen tarvitaan federoitua kyselyä tukeva SPARQL-sovellus. Tässä kokeessa käytimme Apachen Jena-järjestelmää [JENA]. Kyselyn tekemiseksi Jenalla tarvitaan itse kysely tallennettuna tiedostoon SPARQL-muodossa ja tyhjä datatiedosto esimerkiksi Turtle-muodossa. Tyhjä datatiedosto tarvitaan, koska Jena vaatii kyselyä suorittaessa parametrina paikallisen RDF-tietojoukon, johon kysely suoritetaan. Koska kyselymme kohdistuu pelkästään etänä sijaitseviin SPARQL-päätepisteisiin, voi kyseinen datatiedosto olla tyhjä.

Kysely suoritetaan Jenassa komennolla *arq --query kysely.sparql --data data.ttl*. Kyselyn alussa oleva *arq* viittaa Jenan SPARQL-prosessoriin nimeltä ARQ, joka suorittaa kyselyn [JENA]. Komento *--query kysely.sparql* tarkoittaa, että SPARQL-prosessorille annetaan parametrina kysely, joka löytyy tiedostosta kysely.sparql. Komento *--data data.ttl* tarkoittaa parametrina annettavaa tietojoukkoa, johon kysely kohdistetaan. Tässä tapauksessa data.ttl on vain Turtle-muodossa oleva tyhjä tiedosto,

koska teemme kyselyn etänä sijaitseviin päätepisteisiin.

Varsinainen kysely on esitetty esimerkissä 15. Sen alussa määritellään käytössä olevat nimiavaruuslyhenteet: *owl, rdfs, dbo* ja *drugbank*. Lyhenne *dbo* viittaa DBpedian ontologiaan ja *drugbank* taas DrugBankin ontologiaan. Käytämme myös OWL:n ja RDFS:n perussanastoja. OWL-sanastoa käytämme tietolähteiden sitomiseksi yhteen *owl:sameAs*-määrityksen avulla.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX drugbank:
<http://wifo5-04.informatik.uni-mannheim.de/drugbank/resource/drugbank/>

SELECT ?name ?description ?toxicity
WHERE  {
   SERVICE <http://wifo5-04.informatik.uni-mannheim.de/drugbank/sparql> {
      ?drug rdfs:label "Aspirin" .
      ?drug rdfs:label ?name .
      ?drug drugbank:toxicity ?toxicity .
      ?drug owl:sameAs ?dbpediaLink .
   }
   SERVICE <http://dbpedia.org/sparql> {
      ?dbpediaLink dbo:abstract ?description .
      FILTER (lang(?description) = 'en')
   }
}
```

Esimerkki 15: Kahteen SPARQL-päätepisteeseen kohdistuva federoitu kysely.

Koska haluamme kyselyyn vastauksena taulukoitua dataa, käytämme *SELECT*-kyselyä. Kyselyyn määritellään tulosmuuttujilla haluavamme tietää lääkkeestä nimen (*?name*), kuvauksen (*?description*) ja tietoa myrkyllisyydestä (*?toxicity*). Varsinainen kysely koostuu kahdesta *SERVICE*-komennolla määritellystä federoidusta alikyselystä. Toinen suunnataan DrugBankin tarjoamaan SPARQL-päätepisteeseen ja toinen DBpedian vastaavaan päätepisteeseen.

DrugBankiin kohdistuvassa alikyselyssä määritellään haettavan lääkkeen nimike RDFS-sanastoa käyttäen (*rdfs:label*) ja asetetaan kyseinen nimike myös tulosmuuttujaan *?name*. Kyselyssämme haettavaksi arvoksi on määritelty "Aspirin". Määritystä muuttamalla on mahdollista hakea tietoa mistä vain DrugBankin sisältämästä lääkkeestä. Haettavan nimen täytyy kuitenkin olla englanniksi. DrugBankin ontologiaa

hyväksikäyttäen (*drugbank:toxicity*) asetamme myös tulosmuuttujaan *?toxicity* tiedon lääkkeen myrkyllisyydestä.

Lopulta määrittelemme, että löydetyn käsitteen täytyy tarkoittaa samaa asiaa kuin toisesta päätepisteestä haettu tieto. Koska DrugBankin resurssit on liitetty DBpedian vastaaviin resursseihin *owl:sameAs*-määrityksellä, voimme yksinkertaisesti asettaa kyselyyn ehdoksi, että molemmista alikyselyistä etsitään samaa tarkoittavaa asiaa: *?drug owl:sameAs ?dbpediaLink*. Jos tietojoukot eivät olisi linkittäneet resurssejaan, ei vastaavanlaisen kyselyn tekeminen olisi yhtä helppoa.

DBpedia-alikyselyssä asetamme kuvauksen lääkkeestä tulosmuuttujaan *?description* käyttämällä DBpedian sanaston tiivistelmää tarkoittavaa määritystä *dbo:abstract*. Lisäksi suodatamme pois muut kuin englanninkieliset tulokset, sillä emme halua kyselyn vastaukseen mukaan useita erikielisiä kuvauksia. Kyselyn tekemisen jälkeen molemmat päätepisteet prosessoivat alikyselynsä ja palauttavat vastauksensa verkon yli Jenalle, joka yhdistää niistä saadun tiedon. Kyselyn tulokset ovat taulukossa 5.

name	description	toxicity
”Aspirin”	”Aspirin, also known as acetylsalicylic acid, is a salicylate drug, often used as an analgesic to relieve minor aches and pains, as an antipyretic to reduce fever, and as an anti-inflammatory medication.”	”Effects of overdose include: abdominal pain, hypotension, pyrexia, renal failure, confusion, seizure, coma and death.”

Taulukko 5: Kyselyn palauttamat tulokset. Tekstejä lyhennetty selkeyden vuoksi.

Hajallaan olevien tietojoukkojen tehokkaaseen hyödyntämiseen tarvitaan tarkoituksenmukaisien sanastojen käyttöä. Esimerkki havainnollistaa, kuinka tietojoukkojen yhteenlinkittyminen sanaston avulla yksinkertaistaa niistä haetun tiedon yhdistämistä.

7.3.2 Linkkien seuraaminen

Tässä esimerkissä havainnollistamme hajallaan olevan RDF-muotoisen tiedon kyselyä seuraamalla tiedossa esiintyviä linkkejä. Esimerkissä 16 on SPARQL-kysely, jolla haluamme selvittää millä alalla Topin tutut työskentelevät. Koska kyselyssä ei määritellä erikseen tietolähteitä federoidun kyselyn tapaan, niin tieto täytyy löytää

linkkejä seuraamalla. Taulukossa 6 on kyselyn aikana linkkejä seuraamalla löytämämme tietokokoelmat ja niihin kuuluvat RDF-kolmikot.

```
SELECT ?henkilo ?ala WHERE {
  <esim.fi/topi> <sanasto.fi/tuntee> ?henkilo .
  ?henkilo <sanasto.fi/tyopaikka> ?tyo .
  ?tyo <sanasto.fi/ala> ?ala .
}
```

Esimerkki 16: Esimerkkikysely SPARQL-kielellä.

Kyselyn aluksi tutkitaan kyselyssä määriteltyjä URI-tunnisteita. Tunnisteen *esim.fi/topi* kautta löydetään tietokokoelma 1, jossa on kolme RDF-kolmikoilla ilmaistua toteamusta: Topi tuntee Tarun (*esim.fi/taru*), Topi tuntee Timin (*esim.fi/timi*) ja Topin nimi on ”Topi”. Kaksi ensimmäistä kolmikkoa sopivat kyselyn ensimmäiseen määrittelyyn, mutta koko kyselyyn ei ole vielä vastattu, joten tutkitaan myös Tarun ja Timin URI-tunnisteiden kautta löytyvä tieto. Niistä löytyvät tietokokoelmat 2 ja 3, joissa muun muassa kerrotaan Tarun työskentelevän Koodipajassa (*firmat.fi/koodipaja*) ja Timin työskentelevän Lehtimestassa (*firmat.fi/lehtimesta*). Työpaikan kertovat kolmikot sopivat kyselyn toiseen määrittelyyn. Tutkimalla URI-tunnisteet *firmat.fi/koodipaja* ja *firmat.fi/lehtimesta* löydetään tietokokoelmat 3 ja 4, joissa kerrotaan Koodipajan alan olevan ”IT” ja Lehtimestan alan olevan ”Viestintä”. Nämä kolmikot sopivat kyselyn kolmanteen määrittelyyn.

Tietokokoelma 1 (esim.fi/topi)
esim.fi/topi, sanasto.fi/tuntee, esim.fi/taru
esim.fi/topi, sanasto.fi/tuntee, esim.fi/timi
esim.fi/topi, sanasto.fi/nimi, ”Topi”
Tietokokoelma 2 (esim.fi/taru)
esim.fi/taru, sanasto.fi/nimi, ”Taru”
esim.fi/taru, sanasto.fi/tyopaikka, firmat.fi/koodipaja
Tietokokoelma 3 (esim.fi/timi)
esim.fi/timi, sanasto.fi/tyopaikka, firmat.fi/lehtimesta
Tietokokoelma 4 (firmat.fi/koodipaja)
firmat.fi/koodipaja, sanasto.fi/ala, ”IT”
Tietokokoelma 5 (firmat.fi/lehtimesta)
firmat.fi/lehtimesta, sanasto.fi/ala, ”Viestintä”

Taulukko 6: Esimerkkikyselyn aikana löydetyt tietokokoelmat ja niiden sisältämät RDF-kolmikot.

Näin kyselyn vastaukseksi saatiin tauulukossa 7 kaksi tulosta: Taru on IT-alalla ja Timi on viestintäalalla. Huomattavaa on, että ennen kyselyn suoritusta emme tienneet mitään tietokokoelmista 2 *(esim.fi/taru)*, 3 (*esim.fi/timi*), 4 (*firmat.fi/koodipaja*) tai 5 (*firmat.fi/lehtimesta*), vaan löysimme ne ajonaikaisesti linkkejä seuraamalla.

henkilo	**ala**
esim.fi/taru	”IT”
esim.fi/timi	”Viestintä”

Taulukko 7: Linkkejä seuraamalla löydetyt tulokset.

8 Yhteenveto

Linkitetyistä dokumenteista muodostuvan perinteisen verkon rinnalla toimii myös linkitetystä datasta koostuva verkko. Linkitetty data on osa laajempaa visiota tietokoneiden ymmärtämästä semanttisesta webistä, jossa tiedolla on eksakti merkitys. Käytännön tasolla visiota toteutetaan W3C:n määrittelemillä standardeilla.

RDF muodostaa pohjan semanttiselle webille. Se mahdollistaa webin resurssien ominaisuuksien ja yhteyksien joustavan kuvailun. RDF:n yksinkertainen idea ja helppo käyttö edesauttavat tietomallin omaksumista. Siinä resursseja kuvaillaan subjekti-predikaatti-objekti -kolmikoilla, jotka kertovat kuvailtavan kohteen, ominaisuuden ja arvon. Jokainen resurssi tunnistetaan URI-tunnisteella. RDF ei ole sidottu yhteen tiettyyn esitysformaattiin. Alun perin yhdessä RDF-standardin kanssa julkaistiin XML-pohjainen RDF/XML-syntaksi, mutta nykyään muun muassa yksinkertaistettu Turtle on suosittu.

RDF-muotoisen tiedon kyselyyn käytetään SPARQL-kyselykieltä. SPARQL on W3C:n standardoima SQL-tyylinen semanttinen kyselykieli, jonka eri kyselymuodot mahdollistavat RDF-datan monipuolisen kyselyn. Kyselyn lisäksi sillä voi myös muokata tietoa. Version 1.1 myötä kieleen lisättiin useiden hajallaan olevien tietolähteiden federoitu kysely.

RDF-muotoisen tiedon semanttisesti voimakkaampaan kuvailuun ja luokitteluun käytetään ontologioita. W3C:n OWL-kieliperhe on hallitseva web-ontologiastandardi. OWL pohjautuu aikaisempiin teknologioihin, kuten DAML+OIL-ontologiakieleen. Ontologiat ovat tärkeitä tehokkaan päättelyn ja sovellusten välisen yhteentoimivuuden mahdollistamiseksi.

Verkossa avoinna olevan semanttisen linkitetyn datan määrä kasvaa jatkuvasti. Datan julkaisua ohjaavat neljä Tim Berners-Leen laatimaa periaatetta, jotka suosittelevat käyttämään käsitteiden nimeämiseen HTTP-URI-tunnisteita, kuvailemaan tietoa vallitsevia standardeja käyttäen ja linkittämään datan muihin tietojoukkoihin. Mikään ei pakota noudattamaan periaatteita, mutta julkaistun datan hyödyllisyys kasvaa niitä noudatettaessa. Linkitetty data mahdollistaa uuden tiedon löytämisen ajonaikaisesti linkkejä seuraamalla, mikä taas luo pohjaa uudenlaisille älykkäille verkkoa

hyödyntäville sovelluksille.

Saatavilla olevan tiedon suuri määrä ja hajanainen luonne luovat haasteita RDF-muotoisen tiedon hallintaan. Koska verkon sisältämää tietoa ei hallitse yksi globaali ontologia, tarvitaan tiedon tehokkaaseen hyödyntämiseen yhteisiä ontologioita tai ontologioiden yhteensovittamista. Yhteensovittamiseen on useita strategioita ja automaattisia apuvälineitä, mutta niihin liittyy vielä haasteita. Yksi suurimmista on ihmisten rooli yhteensovittamisprosessissa. Prosessin tulokset eivät ole täydellisiä, joten ne tulisi tarkastaa ja hyväksyä ihmisen toimesta, mutta käsiteltävän tiedon määrä voi olla niin suuri, että se on epäkäytännöllistä.

Semanttinen web sisältää runsaasti sanastoja ja ontologioita eri tarkoituksia varten. Nostimme esille ongelmia sanastojen säilytyksessä, sillä tällä hetkellä sanastojen säilytykselle ei ole vallitsevaa standardia ja suosittujen säilytyspalveluiden sulkeutuminen voisi aiheuttaa haittaa koko semanttiselle webille. Lisäksi kritisoimme omien sanastojen käyttämistä yleisluontoisienkin termien kanssa yleisien sanastojen sijaan, mikä vaikeuttaa tiedon uudelleenkäyttöä ja ymmärtämistä. Ongelmana voi olla, että nykyiset sanastot eivät ole tarpeeksi pitkälle kehittyneitä tai laajasti käytössä.

Hajallaan olevaan RDF-muotoiseen tietoon voidaan tehdä kyselyjä eri tekniikoita käyttäen. Keskitetyssä kyselyssä kaikki tarvittava data ladataan ja yhdistetään ennalta yhdeksi isoksi tietojoukoksi, johon tehdään normaaleja SPARQL-kyselyjä. Federoidussa kyselyssä sen sijaan määritellään alikyselyitä etäällä sijaitseviin SPARQL-päätepisteisiin. Päätepisteet käsittelevät omat kyselynsä itsenäisesti ja palauttavat tulokset kyselijälle, jolloin saatu tieto yhdistetään. Mielenkiintoisin kyselytapa on etsiä tietoa datassa olevia linkkejä seuraamalla. Siinä käytettäviä tietojoukkoja ei tarvitse määritellä eikä tietää ennalta, koska uusia tietojoukkoja löydetään kyselyn aikana.

Hajallaan olevan datan kyselyn havainnollistamiseksi teimme kaksi koekyselyä. Ensimmäisessä haimme tietoa kahdesta eri SPARQL-päätepisteestä federoitua kyselyä käyttäen. Toisessa kyselyssä esittelimme linkkejä seuraamalla tapahtuvan kyselyn toimintaa.

Lähteet

ACH11 K. Alexander, R. Cyganiak, M. Hausenblas ja J. Zhao. Describing Linked Datasets with the VoID Vocabulary. World Wide Web Consortium Interest Group Note, 2011.

Ala11 F. Alahmari. Evaluating SPARQL Using Query Federation and Link Traversal. *ICDIM '11, Proceedings of the Sixth International Conference on Digital Information Management*, IEEE, 2011, sivut 79 – 84.

ArP11 M. Arenas ja J. Pérez. Querying Semantic Web Data with SPARQL. *PODS '11, Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ACM, 2011, sivut 305 – 316.

ABM15 B. Adida, M. Birbeck, S. McCarron ja I. Herman. RDFa 1.1. World Wide Web Consortium Recommendation, 2015.

BAC13 C. Buil-Aranda, M. Arenas, O. Corcho ja A. Polleres. Federating Queries in SPARQL 1.1: Syntax, Semantics and Evaluation. *Journal of Web Semantics*, vol 18, no 1, 2013, sivut 1 – 17.

BBH09 T. Berners-Lee, C. Bizer ja T. Heath. Linked Data – The Story So Far. *International Journal on Semantic Web and Information Systems*, vol 5, no 3, 2009, sivut 1 – 22.

BBP14 D. Beckett, T. Berners-Lee, E. Prud'hommeaux ja G. Carothers. RDF 1.1 Turtle, Terse RDF Triple Language. World Wide Web Consortium Recommendation, 2014.

BCC06 T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly ja kumppanit. Tabulator: Exploring and Analyzing Linked Data on the Semantic Web. *SWUI '06, Procedings of the 3rd International Semantic Web User Interaction Workshop*, 2006.

BeB09 C. Becker ja C. Bizer. Exploring The Geospatial Semantic Web With Dbpedia Mobile. *Journal of Web Semantics*, vol 7, no 4, 2009, sivut

278 – 286.

Ber06 T. Berners-Lee. Linked Data – Design Issues, 2006. http://www.w3.org/DesignIssues/LinkedData.html [28.1.2016]

BHL01 T. Berners-Lee, J. Hendler ja O. Lassila. The Semantic Web. *Scientific American*, vol 284, no 5, 2001, sivut 28 – 37.

Biz09 C. Bizer. The Emerging Web of Linked Data. *IEEE Intelligent Systems*, vol 24, no 5, 2009, sivut 87 – 92.

BKO11 B. Bishop, A. Kiryakov, D. Ognyanov ja kumppanit. Factforge: A Fast Track to the Web of Data. *Semantic Web Journal*, vol 2, no 2, 2011, sivut 157 – 166.

BLK09 C. Bizer, C. Lehmann, J. Kobilarov ja kumppanit. DBpedia - A crystallization point for the Web of Data. *Journal of Web Semantics*, vol 7, no 3, 2009, sivut 154 – 165.

BMM08 S. M. Benslimane, A. Merazi, M. Malki ja D. A. Bensaber. Ontology Mapping for Querying Heterogeneous Information Sources. *INFOCOMP Journal of Computer Science*, vol 7, no 3, 2008, sivut 52 – 59.

BrA03 J. Broekstra ja A. Kampman. SeRQL: A Second Generation RDF Query Language. *Proceedings of the SWAD-Europe Workshop on Semantic Web Storage and Retrieval*, 2003, sivut 13 – 14.

BrG14 D. Brickley ja R.V. Guha. RDF Schema 1.1. World Wide Web Consortium Recommendation, 2014.

BrM14 D. Brickley ja L. Miller. FOAF Vocabulary Specification. Namespace Document, 2004. http://xmlns.com/foaf/spec/ [9.2.2016]

ChD97 S. Chaudhuri ja U. Dayal. An Overview of Data Warehousing and OLAP Technology. *ACM Sigmod Record*, vol 26, no 1, 1997, sivut 65 – 74.

CJB99 B. Chandrasekaran, J.R. Josephson ja V.R. Benjamins. What Are Ontologies, And Why Do We Need Them? *IEEE Intelligent Systems and Their Applications*, vol 14, no 1, 1999, sivut 20 – 26.

CSM10 G. Correndo, M. Salvadores, I. Millard, H. Glaser ja N. Shadbolt. SPARQL Query Rewriting for Implementing Data Integration Over Linked Data. *EDBT '10*, *Proceedings of the 2010 EDBT/ICDT Workshops*, ACM, 2010, sivut 4.1 – 4.11.

CWL14 R. Cyganiak, D. Wood ja M. Lanthaner. RDF 1.1 Concepts and Abstract Syntax. World Wide Web Consortium Recommendation, 2014.

DBLCR Dublin Core Metadata Initiative Terms. http://dublincore.org/documents/dcmi-terms/ [20.1.2016]

DGH14 X. Dong, E. Gabrilovich, G. Heitz ja kumppanit. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2014, sivut 601 – 610.

DeS04 M. Dean, G. Schreiber ja kumppanit. OWL Web Ontology Language Reference. World Wide Web Consortium Recommendation, 2004.

DMM00 S. Decker, P. Mitra ja S. Melnik. Framework for the Semantic Web: an RDF tutorial. *IEEE Internet Computing*, vol 4, no 6, 2000, sivut 68 – 73.

DMV00 S. Decker, S. Melnik, F. Van Harmelen ja kumppanit. The Semantic Web: the Roles of XML and RDF. *IEEE Internet Computing*, vol 4, no 5, 2000, sivut 63 – 73.

DNRDF DotNetRDF, RDF-kirjasto .NET:lle. http://www.dotnetrdf.org/ [7.1.2016]

EARDF EasyRDF, RDF-kirjasto PHP:lle. http://www.easyrdf.org/ [7.1.2016]

ErM09 O. Erling ja I. Mikhailov. RDF Support in the Virtuoso DBMS. *Networked Knowledge – Networked Media*, Springer, 2009, sivut 7 – 24.

FCO12 A. Freitas, E. Curry, J.G. Oliveira ja S.O. Riain. Querying Heterogeneous Datasets on the Linked Data Web: Challenges, Approaches, and Trends. *IEEE Internet Computing*, vol 16, no 1, 2012, sivut 24 – 33.

FIDAT Avoindata.fi – Avoimen tiedon ja yhteentoimivuuden palvelu. http://www.avoindata.fi/ [14.1.2016]

FRAID Yhdysvaltain ja Iso-Britannian ulkomaanavun vertailusovellus. data-gov.tw.rpi.edu/demo/linked/aidviz-1554-10030.html [1.3.2016]

FVH01 D. Fensel, F. Van Harmelen, I. Horrocks ja kumppanit. OIL: an Ontology Infrastructure for the Semantic Web. *IEEE Intelligent Systems*, vol 16, no 2, 2001, sivut 38 – 45.

FWC13 L. Feigenbaum, G. Williams, K. Clark ja E. Torres. SPARQL 1.1 Protocol. World Wide Web Consortium Recommendation, 2013.

GaS14 F. Gandon ja G. Schreiber. RDF 1.1 XML Syntax. World Wide Web Consortium Recommendation, 2014.

GHM08 B.C. Grau, I. Horrocks, B. Motik ja kumppanit. OWL 2: The Next Step for OWL. *Journal of Web Semantic*s, vol 6, no 4, 2008, sivut 309 – 322.

GoC02 A. Gomez-Perez ja O. Corcho. Ontology Languages for the Semantic Web. *IEEE Intelligent Systems*, vol 17, no 1, 2002, sivut 54 – 60.

GrM12 J. Gracia ja E. Mena. Semantic Heterogeneity Issues on the Web. *IEEE Internet Computing*, vol 16, no 5, 2012, sivut 60 – 67.

HaF12 O. Hartig ja J.C. Freytag. Foundations of Traversal Based Query Execution Over Linked Data. *HT '12, Proceedings of the 23rd ACM Conference on Hypertext and Social Media*, ACM, 2012, sivut 43 – 52.

HaL10 O. Hartig ja A. Langegger. A Database Perspective on Consuming Linked Data on the Web. *Datenbank-Spektrum*, vol 10, no 2, 2010, sivut 57 – 66.

Har10 A. Harth. VisiNav: A System for Visual Search and Navigation on Web Data. *Journal of Web Semantic*s, vol 8, no 4, 2010, sivut 348 – 354.

Har11 O. Hartig. Zero-knowledge Query Planning for an Iterator Implementation of Link Traversal Based Query Execution. *The Semantic Web: Research and Applications*, 2011, Springer, sivut 154 – 169.

HaS13 S. Harris ja A. Seaborne. SPARQL 1.1 Query Language. World Wide

Web Consortium Recommendation, 2013.

Hau09 M. Hausenblas. Exploiting Linked Data to Build Web Applications. *IEEE Internet Computing*, vol 13, no 4, 2009, sivut 68 – 73.

HaZ12 O. Hartig ja J. Zhao. Provenance Vocabulary Core Ontology Specification. http://trdf.sourceforge.net/provenance/ns.html [6.4.2016]

HBF09 O. Hartig, C. Bizer ja J. Freytag. Executing SPARQL Queries over the Web of Linked Data. *ISWC '09, Proceedings of The 8th International Semantic Web Conference*, 2009, Springer, sivut 293 – 309.

HeB11 T. Heath ja C. Bizer. Linked Data: Evolving the Web Into a Global Data Space. Morgan & Claypool, 2011.

HeC06 B. He ja K. Chang. Automatic Complex Schema Matching Across Web Query Interfaces: A Correlation Mining Approach. *ACM Transactions on Database Systems*, vol 31, no 1, 2006, sivut 346-395.

HeH00 J. Heflin ja J. Hendler. Dynamic Ontologies on the Web. *AAAI '00, Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 2000, sivut 443 – 449.

HeM00 J. Hendler ja D.L. McGuinness. The DARPA Agent Markup Language, *IEEE Intelligent Systems*, vol 15, no 6, 2000, sivut 67 – 73.

Hen09 J. Hendler. Web 3.0 Emerging. *Computer*, vol 42, no 1, 2009, sivut 111 – 113.

HEP03 M. Hori, J. Euzenat ja P.F. Patel-Schneider. OWL Web Ontology Language XML Presentation Syntax. World Wide Web Consortium Recommendation, 2003.

HHM12 J. Hendler, J. Holm, C. Musialek ja G. Thomas. US Government Linked Open Data: Semantic.data.gov. *IEEE Intelligent Systems*, vol 27, no 3, 2012, sivut 25 – 31.

HKP12 P. Hitzler, M. Krötzsch, B. Parsia ja kumppanit. OWL 2 Web Ontology Language 2. World Wide Web Consortium Recommendation, 2012.

HoP12 M. Horridge ja P.F. Patel-Schneider. OWL 2 Web Ontology Language Manchester Syntax. World Wide Web Consortium Working Group Note, 2012.

HPV03 I. Horrocks, P. Patel-Schneider ja F. Van Harmelen. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. *Journal of Web Semantics*, vol 1, no 1, 2003, sivut 7 – 26.

HuQ08 W. Hu ja Y. Qu. Falcon-AO: A Practical Ontology Matching System. *Journal of Web Semantics*, vol 6, no 3, 2008, sivut 237 – 239.

JENA Apache Jena – Semanttisen webin ohjelmistokehys Javalle. http://jena.apache.org/ [21.1.2016]

JSON JSON – The JavaScript Object Notation Data Interchange Format. http://www.json.org/ [19.1.2016]

Kar02 G. Karvounarakis ja kumppanit. RQL: A Declarative Query Language for RDF. *Proceedings of the 11th International Conference on World Wide Web*, ACM, 2002, sivut 592 – 603.

KEGG KEGG – Bioinformatiikan tietopankki. http://www.kegg.jp/ [4.2.2016]

Kle01 M. Klein. XML, RDF, and Relatives. *IEEE Intelligent Systems*, vol 16, no 2, 2001, sivut 26 – 28.

LaH07 O. Lassila ja J. Hendler. Embracing Web 3.0. *IEEE Internet Computing*, vol 11, no 3, 2007, sivut 90 – 93.

LaS12 M. Lanzenberger ja J. Sampson. Making Ontologies Talk: Knowledge Interoperability in the Semantic Web. *IEEE Intelligent Systems*, vol 23, no 6, 2008, sivut 72 – 85.

Las98 O. Lassila. Web Metadata: A Matter of Semantics. *IEEE Internet Computing*, vol 2, no 4, 1998, sivut 30 – 37.

LaT06 P. Lambrix ja H. Tan. SAMBO – A System for Aligning and Merging Biomedical Ontologies. *Journal of Web Semantics*, vol 4, no 1, 2006, sivut 196 – 206.

LaT10 G. Ladwig ja T. Tran. Linked Data Query Processing Strategies. *ISWC '10, Proceedings of The 9th International Semantic Web Conference*, 2010, Springer, sivut 453 – 469.

LTL09 J. Li, J. Tang, Y. Li ja Q. Luo. RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. *IEEE Transactions on Knowledge and Data Engineering*, vol 21, no 8, 2009, sivut 1218 – 1232.

MaS01 A. Maedche ja S. Staab. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, vol 16, no 2, 2001, sivut 72 – 79.

MFH02 D.L. McGuinness, R. Fikes, J. Hendler ja L.A. Stein. DAML+OIL: An Ontology Language for the Semantic Web. *IEEE Intelligent Systems*, vol 17, no 5, 2002, sivut 72 – 80.

MLR10 V. Mascardi, A. Locoro ja P. Rosso. Automatic Ontology Matching via Upper Ontologies: A Systematic Evaluation. *IEEE Transactions on Knowledge and Data Engineering*, vol 22, no 5, 2010, sivut 609 – 623.

MPP12 B. Motik, P.F. Patel-Schneider ja B. Parsia. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. World Wide Web Consortium Recommendation, 2012.

MUONT The Music Ontology – Musiikin alan ontologia. http://musicontology.com/ [22.2.2016]

NCL06 N.F. Noy, A. Chugh, W. Liu ja M.A. Musen. A Framework for Ontology Evolution in Collaborative Environments. *ISWC '06, Proceedings of the Fifth International Semantic Web Conference*, 2006, Springer, sivut 544 – 558.

NiP01 I. Niles ja A. Pease. Towards A Standard Upper Ontology. *FOIS '01, Proceedings of the 2nd International Conference on Formal Ontology in Information Systems*, ACM, 2001, sivut 2 – 9.

OCD08 E. Oren, R. Delbru, M. Catasta, R. Cyganiak ja kumppanit. Sindice.com: A Document-Oriented Lookup Index For Open Linked Data. *International Journal of Metadata, Semantics and Ontologies*, vol 3, no 1, 2008, sivut 37 – 52.

PKDAT Hri.fi – Helsinki Region Infoshare – Pääkaupunkiseudun avoin data. http://www.hri.fi/ [14.1.2016]

PrB13 E. Prud'hommeaux, C. Buil-Aranda ja kumppanit. SPARQL 1.1 Federated Query. World Wide Web Consortium Recommendation, 2013.

PURL Purl.org – Uudelleenohjauspalvelu web-resursseille https://purl.org/ [31.3.2016]

QuL08 B. Quilitz ja U. Leser. Querying Distributed RDF Data Sources with SPARQL. Springer, 2008.

RASQL Rasqal, SPARQL-kyselykirjasto C:lle. http://librdf.org/rasqal/ [7.1.2016]

SBJ14 M. Schmachtenberg, C. Bizer, A. Jentzsch ja R. Cyganiak. Linking Open Data Diagram. http://lod-cloud.net/ [27.2.2016]

SBP14 M. Schmachtenberg, C. Bizer ja H. Paulheim. Adoption of the Linked Data Best Practices in Different Topical Domains. *ISWC '14, Proceedings of the 13th International Semantic Web Conference*, 2014, Springer, sivut 245 – 260.

Sch11 F. Schmedding. Incremental SPARQL Evaluation for Query Answering on Linked Data. *Proceedings of the 2nd International Workshop on Consuming Linked Data at ISWC*, 2011.

SESAM Sesame, RDF-ohjelmistokehys Javalle. http://rdf4j.org/ [7.1.2016]

SHB06 N. Shadbolt, W. Hall ja T. Berners-Lee. The Semantic Web Revisited. *IEEE Intelligent Systems*, vol 21, no 3, 2006, sivut 96 – 101.

ShE13 P. Shvaiko ja J. Euzenat. Ontology Matching: State of the Art and Future Challenges. *IEEE Transactions on Knowledge and Data Engineering*, vol 25, no 1, 2013, sivut 158 – 176.

ShL90 A. Sheth ja J. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, vol 22, no 3, 1990, sivut 183 – 236.

ShO13 N. Shadbolt ja K. O'Hara. Linked Data in Government. *IEEE Internet Computing*, vol 17, no 4, 2013, sivut 72 – 77.

SKL14 M. Sporny, G. Kellogg, M. Lanthaner ja kumppanit. JSON-LD 1.0, A JSON-based Serialization for Linked Data. World Wide Web Consortium Recommendation, 2014.

SKW07 F. M. Suchanek, G. Kasneci ja G. Weikum. Yago: A Core of Semantic Knowledge. *Proceedings of the 16th International Conference on World Wide Web*, ACM, 2007, sivut 697 – 706.

SNOMD SNOMED CT – Lääketieteellisten termien tietopankki. http://www.nlm.nih.gov/snomed/ [28.1.2016]

SOB12 N. Shadbolt, K. O'Hara, T. Berners-Lee ja kumppanit. Linked Open Government Data: Lessons from Data.gov.uk. *IEEE Intelligent Systems,* vol 27, no 3, 2012, sivut 16 – 24.

SPG07 E. Sirin, B. Parsia, B. Grau ja kumppanit. Pellet: A Practical OWL-DL Reasoner. *Journal of Web Semantics*, vol 5, no 2, 2007, sivut 51 – 53.

UKDAT Data.gov.uk – Iso-Britannian hallituksen julkaisema avoin data. http://data.gov.uk/ [14.1.2016]

UKH12 J. Umbrich, M. Karnstedt, A. Hogan ja J. Parreira. Hybrid SPARQL Queries: Fresh vs. Fast Results. *ISWC '12, Proceedings of The 11th International Semantic Web Conference*, 2012, Springer, sivut 608 – 624.

USDAT Data.gov – Yhdysvaltain hallituksen julkaisema avoin data. http://www.data.gov/ [14.1.2016]

WKG06 D.S. Wishart, C. Knox, A.C. Guo ja kumppanit. DrugBank: A Comprehensive Resource for In Silico Drug Discovery and Exploration. *Nucleic Acids Research*, vol 34, 2006, sivut 668 – 672.

Woo14 D. Wood. What's New in RDF 1.1. World Wide Web Consortium Working Group Note, 2014.

WRP05 A. Wong, P. Ray, N. Parameswaran ja J. Strassner. Ontology Mapping for the Interoperability Problem in Network Management. *IEEE Journal*

on Selected Areas in Communications, vol 23, no 10, 2005, sivut 2058 – 2068.

XMLNS Xmlns.com – URL-tunnuspalvelu semanttisen webin sanastoille http://xmlns.com/ [31.3.2016]

YSO YSO – Yleinen suomalainen ontologia. http://finto.fi/yso/fi/ [9.2.2016]

ZhM03 Z. Zhihong ja Z. Mingtian. Web Ontology Language OWL and Its Description Logic Foundation. *PDCAT'03, Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, IEEE, 2003, sivut 157 – 160.