Semanttinen web:
linkitetyn avoimen datan käsikirja

Eero Hyvönen ©
Aalto-yliopisto ja Helsingin yliopisto
Semantic Computing Research Group (SeCo)
Helsinki Centre for Digital Humanities (HELDIG)
http://seco.cs.aalto.fi ja http://heldig.fi

Versio 11.11.2017

Sisältö

Esipuhe

Lähtölaukaus tämän teoksen kirjoittamiselle paukahti helmikuussa 2001, kun webin infrastruktuurin kehitystyötä maailmanlaajuisesti koordinoiva W3C-järjestö[1] käynnisti laajamittaisen semanttisen webin kehitysohjelmansa "Semantic Web Activity". Sen visiona oli luoda WWW:n perustaksi tietoinfrastruktuuri, joka yhdistäisi verkossa olevan tiedon maailmanlaajuiseksi semanttiseksi, tietokoneiden "ymmärtämäksi" verkoksi, Web of Data. Idea lannoitti kaltaiseni tekoälytutkijan mieltä siinä määrin, että heti samana syksynä järjestin Helsingin yliopistolla yhteistyökumppaneiden ja W3C:n väen kanssa tilaisuuden "Semantic Web Kick-off in Finland"[2], johon saapui yllättäen yli 200 kuulijaa. Aihe selvästi kiinnosti monia eri tahoja ja sen piiriin syntyikin nopeasti kansainvälinen tutkimustraditio omine konferenssisarjoineen ja journaaleineen. Itse innostuin saman tien perustamaan aihepiiriä tutkivan Semanttisen laskennan tutkimusryhmän (SeCo)[3] Helsingin yliopiston tietojenkäsittelytieteen laitokselle ja juuri toimintansa aloittaneeseen, Teknillisen korkeakoulun (nykyisen Aalto-yliopiston) ja Helsingin yliopiston yhteiseen Helsinki Institute for Information Technology (HIIT) -tutkimuskeskukseen[4].

Uuden tutkimus- ja kehitysalueen yhtenä kotimaisena haasteena oli opetuksen käynnistäminen uusista semanttisista web-teknologioista ja suomen kielen terminologian luominen alalle. Alussa asiaa hoidettiin satunnaisten projektikurssien ja tutkimusseminaarien kautta, mutta alan teknologioiden ja W3C:n standardien vakiintuessa kasvoi tarve alan peruskurssin kehittämiseksi. Tähän päästiin kuitenkin vasta Teknillisen kor-

[1] https://www.w3.org/

[2] Tilaisuuden esitelmät julkaisiin HIIT-tutkimuskeskuksen julkaisusarjan ensimmäisenä niteenä [27].

[3] http://seco.cs.aalto.fi/

[4] http://www.hiit.fi

keakoulun viestintätekniikan laboratoriossa keväällä 2005, jonne oli perustettu juuri tähän aihepiiriin fokusoitu uusi semanttisen viestintätekniikan professuuri. Minut nimitettiin virkaan ja siirryimme tutkimusryhmäni kanssa Otaniemeen, mutta yhteistyö Helsingin yliopiston kanssa on jatkunut näihin päiviin.

Tarve suomen kieliselle oppikirjalle tuli vastaan heti ensimmäisellä luentokerralla: voiko tämä ala olla uskottava, jos siitä ei ole saatavilla suomen kielistä kirjallisuutta? Kesti kuitenkin peräti kymmenen vuotta tarttua tehtävään. Ratkaiseva taitekohta oli apurahahakemuksen jättäminen WSOY:n kirjallisuusäätiölle, jonka myönteisen päätöksen jälkeen pakotie oli tukossa, sekä Aalto-yliopiston myöntämä "sapattivapaa", jonka tavoitteisiin kirjan kirjoittaminen kirjattiin 2015–2016. Kymmenen vuoden aikana yliopisto-opetuksen kansainvälistyminen oli tosin edennyt siinä määrin, että teoksen pohjana olevaa Semanttinen web -kurssia Aalto-yliopistossa alettiin luennoida englanniksi, mutta toisaalta digitalisaation edetessä yhä uusille aloille ohjelmointia on alettu opettamaan enenemässä määrin eri sovellusaloilla ja yhä nuoremmille, kuten lukioissa. Lisäksi digitaalisten ihmistieteiden ja Helsingin yliopiston uuden aihepiiriä edistävän HELDIG-keskuksen[5] piirissä on syntynyt uutta tarvetta suomenkieliselle oppimateriaalille. Oman kohderyhmänsä teokselle muodostavat tutkijat ja kehittäjät, joiden nuoruudessa webiä ei ollut tai sen teknologioita opetettu, mutta jotka jälkeenpäin omaehtoisesti haluavat täydentää osaamistaan web-ilmiöstä. Web ja internet ovat ennennäkemättömällä tavalla muuttaneet maailmaamme vain parinkymmenen vuoden kuluessa ja voidaan lukea jo nyt ihmiskunnan suurimpien innovaatioiden joukkoon[6].

Tämän teoksen tavoitteena on luoda konkreettinen yleiskuva kehityssuunnasta kohti tekoälyyn perustuvaa, "älykästä" semanttista webiä. Tavoitteena on, että kirjan lukenut pystyisi arvioimaan semanttisen webin ja linkitetyn datan mahdollisuuksia omalla sovellusalueellaan ja ymmärtämään myös teknologian haasteita ja rajoituksia. Toivon mukaan kirjasta voi myös saada kipinän opiskella ja kokeilla web-sovellusten ohjelmointia. Esimerkiksi digitaalisten ihmistieteiden tutkijoille ja soveltajille perustietojen ja taitojen hankkiminen laskennallista menetelmistä ja oh-

[5] http://heldig.fi/

[6] Ks. esimerkiksi
https://www.famousscientists.org/5-most-important-inventions-of-all-time/

jelmoinnista on erittäin hyödyllistä.

Teos jakaantuu neljään pääosaan. Ensimmäisessä osassa luodaan laajempi katsaus webin erilaisiin kehityssuuntiin ja megatrendeihin. Näistä tiedon haun ja yhdistämisen haasteisiin sekä "älykkäiden" verkkopalveluiden kehittämiseen keskittyvä semanttinen web on yksi. Käytännön sovellusten tasolla semanttinen web realisoituu nykyisin linkitettynä datana (Linked Data), jonka perusteet esitellään toisessa osassa. Linkitetty data perustuu semanttisen webin standardipinon pohjalla olevaan yksinkertaiseen RDF-tietomalliin ja kieleen, SPARQL-kyselykieleen sekä linkitetyn datan julkaisuperiaatteisiin ja -käytäntöihin.

Kirjan kolmannen osan aiheena ovat RDF-mallin päälle kehitetyt tietämyksen esittämisen mallit ja standardit, joita käytetään eri sovellusalueiden tietosisältöjen kuvaamiseen ns. ontologioina. Semanttisen webin tietämyksen esitysmuodot perustuvat logiikkaan, jonka avulla tietokone pystyy tekemään automaattisesti päätelmiä. Päättelyn avulla voidaan rikastaa automaattisesti verkon tietoa ja toteuttaa älykkäästi toimivia järjestelmiä. Teoksen viimeisessä osassa tarkastellaan sovellusten, erityisesti semanttisten portaalien kehittämistä, verkkomuotoisen datan sisällöntuotantoon liittyviä kysymyksiä ja kaikessa tässä tarvittavaa tietoinfrastruktuuria.

Tämä kirja perustuu pitkäaikaiseen yhteistyöhön opiskelijoiden ja tutkijoiden kanssa Aalto- ja Helsingin yliopiston Semanttisen laskennan tutkimusryhmässä. Sen työtä eri aikoina on rahoittanut ja ohjannut puolensataa eri organisaatiota ja yritystä. Lämmin kiitos kuuluu kaikille teille mukana matkanneille.

Espoossa 6.12.2017

Eero Hyvönen

Luku 1

Johdanto

World Wide Webin (WWW) idean koko maailman kattavasta hypertekstijärjestelmästä esitti Tim Berners-Lee vuonna 1989. Ajatus syntyi tarpeesta tukea CERN-tutkimuskeskuksen fyysikkojen työtä yhteisöllisen, hajautetun julkaisukanavan kautta internetissä. Internetiä oli kehitetty jo aiemmin 60-luvulta saakka kylmän sodan aikaisena tiedonvälityskanavana, jossa solmukohtien tuhoutuminen vaikkapa ydinhyökkäyksen seurauksena voitaisiin korjata reitittämällä tietoliikenne joustavasti kulkemaan edelleen toimivien verkon osien kautta.

Webin keskeinen innovaatio aiempiin viestintäjärjestelmiin verrattuna on lähes ilmainen reaaliaikainen monelta-monelle-julkaiseminen: periaatteessa kuka tahansa voi julkaista webissä sisältöjä kustannustehokkaasti ja muista riippumatta kaikkien käytettäväksi. Perinteisesti tiedon julkaiseminen ja levitys oli ollut vain harvojen käsissä, esimerkiksi lehtitalojen, TV-asemien tai kirjankustantajien. Toki vieläkin tarvitaan palvelun tarjoaja kotisivuille, blogeille tai erityinen yhteisöllinen tiedonjakamisen alusta, kuten Facebook tai Twitter, mutta näitä webissä riittää, eikä julkaisemisen hinta enää ole (yleensä) merkittävä este ainakaan pienimuotoisemmille julkaisuille.

WWW:n uusi tekninen oivallus oli maailmanlaajuinen, *URL-tunnisteisiin* (Uniform Resource Locator) perustuva osoitejärjestelmä ja sen yhdistäminen HTTP-protokollaan, jolla WWW-sivut ja muut dokumentit saadaan haettua URL-osoitteista, ja HTML-kieleen, jonka avulla WWW-sivut ja niiden välinen linkitys esitetään. Tulemme näkemään, että osoitteina käytettävät tunnisteet ovat keskeisessä asemassa myös semantti-

URL-tunniste

sessa webissä, jossa linkitetään *tietoa* WWW-sivujen ohella. Itse asiassa ajatus semanttisesta webistä, jossa yhdistetään tietoja eikä vain webin sivuja toisiinsa, oli mukana jo webin alkuperäisissä hahmotelmissa[1]. WWW toteutui kuitenkin ensin dokumentteja yhdistävänä hypertekstijärjestelmänä, koska sellaisen luominen oli helpompaa ja sellainen joka tapauksessa tarvitaan ihmiskäyttäjiä varten.

Webin merkitys ymmärrettiin nopeasti ja sen käytön lisääntyminen on ollut räjähdysmäistä. Tuskinpa mikään muu tekninen innovaatio on muuttanut maailmaa niin paljon yhtä nopeasti. Vuonna 2006 rikkoontui miljardin ja vuonna 2012 kahden miljardin käyttäjän raja. Tätä kirjoitettaessa webissä arvioidaan olevan ainakin 47 miljardia indeksoitua sivua[2]. Lisäksi verkossa on valtavia määriä hakukoneiden tavoittamattomissa olevia *syvän webin* (deep web, hidden web) tietokantoja, joita kuitenkin käytetään webin kautta. Samalla webin teknologiat ovat sekä kehittyneet että lisääntyneet, mutta perusta on pysynyt paljolti samana.

syvä web

Webin kehitystä voi tarkastella sen piirissä syntyneiden erilaisten megatrendien kautta. Tarkastelemme seuraavassa ensin webin isoja kehityssuuntia, joista tämän teoksen aiheena oleva semanttinen web on yksi keskeisimmistä.

[1] Webin historiaa on muisteltu ja käsitelty mm. teoksessa [6].

[2] Tämä arvio perustuu arvioihin Google-, Bing- ja Yahoo-hakukoneiden indekseistä, ks. http://www.worldwidewebsize.com/

Osa I

Kohti semanttista webiä

Luku 2

Webin kehityssuuntia

Tässä luvussa luodaan katsaus webin kehitykseen tiedon julkaisemisen ja käytön erilaisista näkökulmista. Tämän jälkeen rajataan tämän teoksen näkökulmaksi semanttinen web.

2.1 Web julkaisukanavana

Web tarjoaa kanavan tiedonhankintaa ja sosiaalista kassakäymistä varten. Fyysistä kokemusta näyttöpääte tai mobiililaite ei korvaa, mutta mahdollistaa uusia tapoja tiedon saamiselle ja yhteydenpidolle. Tässä teoksessa keskitytään erityisesti webin tarjoamiin mahdollisuuksiin tietojen yhdistämisen, julkaisemisen ja haun kannalta. Näitä ovat mm.:

1. *Ajasta ja paikasta riippumaton tiedonvälitys.* Tietosisällöt saadaan joustavasti tutkijoiden ja suuren yleisön käyttöön. Tietoon tutustuminen ei edellytä fyysistä läsnäoloa, esimerkiksi kirjastoon tai museoon tulemista tiettyyn aukioloaikaan.

2. *Tietosisällön määrän lisääminen.* Laajat kokoelmat, joiden fyysinen näyttäminen ja tiedon saataville asettaminen olisi mahdotonta, voidaan avata yleisölle WWW:n välityksellä.

3. *Tietosisällön laadun parantaminen.* Tietotekniikka mahdollistaa tietojen haun, yhdistämisen ja esittämisen tavoilla, jotka eivät ole teknisesti tai taloudellisesti mahdollisia fyysisissä tiloissa.

vuorovaikutteisuus
sopeutuvuus
oppiminen

4. *Vuorovaikutteisuuden hyödyntäminen.* Tietojärjestelmien *vuorovaikutteisuus* (interactivity) ja *sopeutuvuus* (adaptivity) ja tarjoavat aiempaa joustavampia ja henkilökohtaisempia tapoja tutustua aineistoihin.

5. *Audiovisualisuuden hyödyntäminen.* Sisältöjä voidaan esitellä audiovisuaalisilla keinoilla, kuten kuva-, musiikki- ja videotallenteilla.

6. *Uudet sisältökohteet ja palvelut.* Uudentyyppisten aineistojen tallentaminen ja julkaiseminen on tullut mahdolliseksi. Esimerkiksi sähköinen asiointi pankeissa ja verkkokaupoissa, sosiaalinen media ja erilaiset pelit ovat verkon käytetyimpiä sisältöjä. Internetsivustoja on määrätietoisesti ryhdytty tallentamaan uudenlaisina historiallisina dokumentteina[1].

Web-sovellusten kehittämisessä on runsaasti käytännöllisiä teknisiä haasteita. Suurimmat haasteet liittyvät kuitenkin tietosisältöjen esittämiseen eikä tekniikkaan. Avainkysymys on, miten web-sivustoista saadaan kävijöiden kannalta aidosti mielenkiintoisia, hyödyllisiä ja helppokäyttöisiä. Tähän ei pelkkä perinteinen tietojen hakumahdollisuus tietokannasta välttämättä riitä. Siinä ei hyödynnetä kuin pieni osa niistä mahdollisuuksista, joita verkon rikas sisältö voi tarjota.

Webin kehittymistä ja haasteita voidaan tarkastella monista tietoon liittyvistä näkökulmista, joihin kuhunkin liittyy omat hype-terminsä sekä poliittisia ja teknologisia kehityspolkuja:

Open Data

1. *Tiedon avoimuus (Open Data).* Tieto on tietoyhteiskunnan toiminnan keskeisiä tukipilareita. Mutta missä määrin sen pitäisi olla avointa (open data) ja suljettua? Erilaisia näkökohtia ja argumentteja voidaan esittää mm. demokratian, liiketoiminnan, tietoturvan ja yksityisyyden suojan kannalta.

Web 2.0

2. *Tiedon yhteisöllinen tuotanto (Web 2.0).* Miten webin tietoa voidaan tuottaa ja julkaista yhteistyössä sen käyttäjien kanssa mahdollisimman helposti?

[1] Ks. esimerkiksi Internet Archive https://archive.org/ ja Kansalliskirjaston Suomalainen verkkoarkisto http://verkkoarkisto.kansalliskirjasto.fi/va/

3. *Tiedon määrä (Big Data).* Miten voidaan tuottaa ja hallita webin yhä suurempia ja dynaamisempia suuraineistoja? *Big Data*

4. *Tieto palveluina (Web Services).* Millaisina toiminnallisina palveluina tieto kannattaa julkaista toisaalta ihmiskäyttäjille toisaalta toisten verkkopalveluiden käytettäviksi? *Web Services*

5. *Tiedon verkko (Web of Data, Linked Data, Semantic Web).* Perinteiset web-sisällöt on tarkoitettu ihmisen luettaviksi toisiinsa linkitettyinä dokumentteina. Webin sisään on kuitenkin rakentumassa myös tietokoneiden käytettäväksi tarkoitettu, konetulkittavan tiedon verkko, joka yhdistää web-sivujen sijasta käsitteitä ja tietoja toisiinsa. Koneellisesti tulkittavan datan avulla voidaan rikastaa tietoja toistensa avulla, päätellä uutta tiedoa ja kehittää älykkäitä sovelluksia (tekoäly). *Web of Data*

Tarkastelemme seuraavassa lyhyesti näitä webin tietoon liittyviä kehityssuuntia.

2.2 Tiedon avoimuus – Open Data

Tietoyhteiskunnan keskeisin tuote tieto poikkeaa perinteisistä teollisuuden tuotteista siinä, että siitä voidaan sekä valmistaa kopioita (monistaa) että jakaa näitä (sähköisesti) lähes ilmaiseksi. Teoriassa tietoyhteiskunnan kansalainen voisi siis elää tiedon osalta loputtomassa yltäkylläisyydessä ja me kaikki olla tiedon miljonäärejä. Sitä ennen on kuitenkin ratkaistava joitain perustavaa laatua olevia kysymyksiä kuten: kuka kustantaisi tietotuotteen ensimmäisen kopion valmistamisen, jos kaikki muut saavat sen ilmaiseksi käyttöönsä? Hyöty korjataan silloin jokin muun tahon kuin tiedon tuottajan toimesta. Yritysten liiketoiminnan osalta avoimen ilmaisen datan haaste on ilmeinen ja edellyttää sitä, että varsinainen liiketoiminta syntyy datan myymisen sijasta siihen liittyvien palveluiden tai muiden tuotteiden kautta.

Julkisen tiedon tuotannon kustannamme me veronmaksajat. Ensimmäisen kopion tuottamisen ja tiedon jakamisen ei siksi pitäisi olla ongelma, jos tiedon avulla voidaan tuottaa kansalaisille hyödylliseksi koettuja palveluita. Julkinen sektori onkin ryhtynyt avaamaan tietokantojaan määrätietoisesti, vaikka julkisen sektorinkin dataan kuten esimerkiksi paikka-

ja säätietoon liittyy taloudellisia intressejä. Ajatuksena on, että laajemman kokonaisuuden kuten Suomi Oy Ab:n kannalta datan avaamisesta saatava hyöty on suurempi kuin yksittäisen julkisen organisaation menettämät myyntituotot.

Toinen perustavaa laatua oleva kysymys on: Eikö tiedon arvo vähene sitä levitettäessä sitä mukaa, kun yhä harvemmalla on tarve ostaa tietoa itselleen? Avoimen datan idea haastaa tämän perinteisen ajattelutavan. Ideana on, että tiedon arvo ei vähene vaan päinvastoin kasvaa sitä avoimesti jakamalla, jos tietoa tuotetaan ja sitä voidaan hyödyntää yhteisöllisesti. Webin kehitys ja Wikipedian, YouTuben ja Facebookin kaltaiset sovellukset ovat osoittaneet, mikä voima yhteisöllisessä avoimessa tiedontuotannossa voi piillä. Sopivasti suunnattu ilmaisuus ja avoimuus tarjoavat myös uusia mahdollisuuksia liiketoiminnallisiin innovaatioihin, kuten vaikkapa Googlen eri palvelut ovat osoittaneet.

Julkisen tiedon avoimeen maksuttomaan avaamiseen sitä tarvitseville tahoille on useita syitä.

- *Demokratian edistäminen.* Kansalaisilla on oltava demokraattinen oikeus saada ja käyttää julkisin varoin tuotettua tietoa maksutta, koska he ovat sen viime kädessä maksaneetkin veroina yms. Tämä on jo pidempään ollut yleisperiaatteena mm. Yhdysvalloissa.

- *Tiedon saatavuuden parantaminen.* Paikallinen tietoon liittyvä eduntavoittelu saattaa estää tiedon hyödyntämisen laajemmassa yhteydessä. Esimerkiksi kartta- ja paikkatiedon osalta helposti saatavilla olevat aineistot kuten Google Maps ja alun perin sotilaskäyttöön kehitetyn, satelliittien kautta saatavan GPS-paikannustiedon avaaminen ovat käynnistäneet todellisen sovellusten ja innovaatioiden aallon. Suomessa julkisia organisaatioita on kuitenkin suorastaan edellytetty mm. maksuperustelailla sulkemaan tietoa ja tekemään taloudellista tulosta aineistoillaan tai ainakin kattamaan datan tuotannon kuluja. Yhden organisaation pieni etu tukahduttaa kuitenkin helposti innovaatioita ja kehitystä laajemmassa yhteydessä.

- *Sisällön yhteisöllinen rikastaminen.* Yhdistämällä oma aineisto muiden toimijoiden aineistoihin jokaisen tiedontuottajan tiedon arvo kasvaa yhteisöllisesti. Esimerkiksi Wikipedian artikkelit erillisinä

eivät ole kovin arvokkaita, vaan arvo syntyy näiden muodostamasta kokonaisuudesta.

- *Yhteentoimivuuden edistäminen.* Yhteiset avoimet pelisäännöt tiedon julkaisemisessa, yhdistämisessä ja jakamisessa edistävät tietojärjestelmien ja palveluiden *yhteentoimivuutta* (interoperability), joka on keskeinen este mm. julkisen sektorien eri toimijoiden yhteistyössä. *yhteentoimivuus*

- *Tuottavuuden tehostaminen.* Tietoja ja ratkaisuja jakamalla voidaan tehostaa työnjako, eliminoida tarpeetonta päällekkäistä työtä ja lisätä näin työn tuottavuutta. Esimerkiksi Suomen kuntien verkkopalvelulla on paljon samoja toiminnallisuuksia, mikä mahdollistaa yhteistyötä tietojärjestelmien kehitystyössä. Asiakkaat ja heidän tarpeensa eri kunnissa ovat paljolti samanlaisia.

- *Uusien innovaatioiden mahdollistaminen.* Tietoa avoimesti julkaisemalla mahdollistetaan ennalta arvaamattomia innovaatioita. Joku voi keksiä ja toteuttaa tiedolle yllättäviä sovelluksia ja hyödyntämistapoja, jos vain tieto on helposti saatavilla. Esimerkkinä mainittakoon vaikkapa asunnonhakupalvelu, jossa voidaan etsiä keskustasta tietyllä ajallisella etäisyydellä olevia kohteita julkisen liikenteen avointen aikataulutietojen avulla.

Tiedon avoimuuden edistämiseksi on perustettu useita kansallisia ja kansainvälisiä ohjelmia ja järjestöjä. Esimerkiksi Open Knowledge International[2] -yhteisö, jolla on aktiivinen alajärjestö Suomessakin, on toimittanut verkossa olevan monikielisen Open Data Handbook -sivuston[3]. Sen kautta kautta voi tutustua tarkemmin avoimeen dataan liittyviin juridisiin, sosiaalisiin ja teknisiin kysymyksiin, avoimen datan menestystarinoihin sekä käytettävissä oleviin aineistoihin ja muihin resursseihin.

2.3 Tiedon yhteisöllisyys – Web 2.0

Webin voima perustuu paljolti yhteisöllisesti julkaistaviin aineistoihin. Webin alkuaikoina käyttäjien tiedontuotanto ja -jakelu, ts. web-

[2] https://okfn.org/
[3] http://opendatahandbook.org/

sivustojen luominen ja omien aineistojen julkaiseminen, ei kuitenkaan ollut kovin yksinkertaista, vaan vaati teknistä osaamista. Web 2.0 -termi lanseerattiin v. 2004 erityisesti Tim O'Reillyn and Dale Doughertyn toimesta viittaamaan uudentyyppisiin, helppokäyttöisiin, yhteentoimiviin web-järjestelmiin, jotka perustuvat käyttäjien itse tuottamaan sisältöön.[4] Monet eniten käytetyistä nykyistä web-palveluista ovat tämän tyyppisiä sovelluksia, esimerkiksi YouTube ja Facebook.

2.4 Tiedon määrä – Big Data

suurdata

skaalautuvuus

Big Data -käsitteellä eli *suurdatalla* viitataan isoihin tietojoukkoihin, joiden tallentaminen, siirtäminen, käsitteleminen ja hallinta nykyisillä relaatiotietokanta- ja tietoteknisillä ratkaisuilla on haasteellista. Se koetaanko tietojoukko isoksi, riippuu siis kulloinkin käytettävissä olevan teknologian mahdollisuuksista ja rajoituksista käsitellä laajoja tietoaineistoja eli *skaalautuvuudesta*.

Dough Laneyn klassisen vuonna 2001 esittämän ns. kolmen V:n mallin mukaan[5] tiedon määrällisen kasvun haasteita ovat:

1. *Volume (määrä)*. Perinteinen tietokantatekniikka ei riitä hyvin suurien datajoukkojen tallentamiseen, siirtoon ja käsittelemiseen. Ratkaisumallina on datan palasteleminen osiin.

2. *Velocity (nopeus)*. Velocity viittaa tässä yhteydessä laskennan nopeuteen. Riittävän tehokkuuden aikaansaamiseksi laskenta tyypillisesti hajautetaan kymmenille tai jopa tuhansille rinnakkain samanaikaisesti toimiville laitteistoilla, joiden tulokset kootaan lopuksi yhteen.

3. *Variety (monimuotoisuus)*. Tieto tai sen osat voivat olla esitettyinä eri tavoin, jolloin niiden käsitteleminen kokonaisuutena hankaloituu. Käytössä voi olla esimerkiksi vaihtoehtoisia tai epäyhtenäisiä tapoja nimien, osoitteiden, numeeristen arvojen yms. esittämiseksi tai näiden välisten yhteyksien kuvaamiseksi.

[4] Lisätietoa Web 2.0 -ilmiöstä ja tiedon yhteisöllisyydestä löytyy esimerkiksi teoksesta [3].

[5] Ks. [32]

Mallia on sittemmin täydennetty uusilla V-ulottuvuuksilla, kuten:

- *Veracity (totuudenmukaisuus)*. Data voi olla epätäsmällistä, puutteellista tai suorastaan virheellistä.
- *Variability (vaihtelevuus)*. Vaihtelevuudella viitataan datan mahdollisten arvojen lisääntymiseen isoissa datajoukoissa.
- *Value (arvo)*. Datan arvo sovelluksissa on keskeinen datan ominaisuus.

Suurdatan käsittelyssä tarvitaan tehokkaita tietojenkäsittelyn menetelmiä ja laitteistoja, joissa tyypillisesti hyödynnetään rinnakkaislaskentaa.[6] Toisaalta suurdatan tulkinnassa ja hyödyntämisessä tarvitaan *data-analyysin* ja *datatieteen* menetelmiä[7] sekä visualisointeja[8]. *data-analyysi* *datatiede*

2.5 Tieto palveluina – Web Services

Passiivisen tiedon julkaisemisen ohella toinen webin kehittymisen päähaaroista on ollut toiminallisuuksien julkaiseminen *verkkopalveluina* (Web Service).[9] Verkkopalvelulla tarkoitetaan tässä yhteydessä palvelua, jossa kone suorittaa laskentatehtävän antamalla sen verkon kautta jollekin sen palvelimelle suoritettavaksi kommunikaatioprotokollaa käyttäen. Web Service -termi isolla kirjoitettuna ei siis viittaa ihmiskäyttäjille tarkoitettuihin verkkosovelluksiin, vaan verkossa olevien koneiden väliseen työnjakoon ja kommunikointiin. Kantavana ideana on WWW:n muuttaminen ikään kuin jättimäiseksi hajautetuksi tietokoneeksi, jonka eri osat erikoistuvat erilaisten tehtävien suorittamiseen ja toistensa auttamiseen. Näin kaikkien verkon toimijoiden ei tarvitse toteuttaa samaa toiminnallisuutta moneen kertaan – keskitetyn yhteisen palvelun kutsuminen yhteisesti sovitun protokollan avulla riittää. Esimerkiksi Google Maps on *verkkopalvelu*

[6] Uusia menetelmiä ja työkaluja suurdatan käsittelyyn ovat mm. Hadoop MapReduce [45] ja Apache Spark [30].

[7] Datatieteen menetelmiä on esitelty esimerkiksi teoksessa [45].

[8] Suosituksi työkaluksi tässä on noussut mm. tilastollisiä menetelmiä ja esitysgrafiikoita hyödyntävä R [14].

[9] Verkkopalvelutekniikoita on esitelty esimerkiksi teoksessa [2].

monipuolinen verkkopalvelu, jonka avulla toiset verkkosovellukset saavat vaivattomasti käyttöönsä paikkatietoa, karttoja ja satelliittikuvia.

Web Services standardeista tunnetuimpia ovat:

- SOAP (Simple Open Access Protocol). SOAP-kielen ja protokollan avulla palvelun käyttäjä lähettää komennon palvelun tarjoajalle ja palvelun tarjoaja antaa komentoon vastauksen.
- WSDL (Web Service Description Language). XML-kieli, jonka avulla palvelun rajapinta määritellään koneluettavalla tavalla. Kuvaus kertoo, miten palvelua kutsutaan, millaisia parametreja sille voidaan antaa ja millaisia vastauksia palvelu palauttaa.
- UDDI (Universal Description Discovery and Integration). UDDI on standardoitu tapa julkaista palvelukuvauksia ja etsiä niitä verkosta. Standardin käyttö on kuitenkin jäänyt vähäiseksi ainakin toistaiseksi.

WWW:n keskeisin protokolla on HTTP, jonka avulla esimerkiksi selaimet lukevat eri palvelimilla olevia verkkosivuja. Yhä keskeisemmäksi verkkopalveluarkkitehtuuriksi on muodostunut HTTP-protokollaan käyttävä

REST

REST (Representational State Transfer). Sen ideana on tarjota yksinkertainen tapa toteuttaa verkkopalveluita, joissa *asiakasjärjestelmä* (client)

asiakasjärjestelmä

antaa HTTP-protokollan mukaisen palvelupyynnön URL-osoitteena *palvelimelle* (server) ja saa siihen vastauksen. Kommunikointi tapahtuu

palvelin

usein JSON-muotoisen datan (JavaScript Object Notation) avulla, joka on kompakti tiedon esitysmuoto ja yksinkertaista käyttää verkkoselainten JavaScript-ohjelmissa.

Yhä useammat sovellukset perustuvat nykyään monipuoliseen selaimen

RIA

ohjelmointiin (Rich Internet Application, RIA), jossa palvelimia käytetään lähinnä datan varastoina ja sovelluksen toiminnallisuus on toteutettu selaimessa. Perinteisten palvelinperustainen sovellusten yksi keskeinen haaste interaktiivisten käyttöliittymien teossa on, ettei käyttöliittymäsivua voida päivittää nopeasti osittain vaan ainoastaan lataamalla aina koko sivu uudelleen palvelimelta, mikä hidastaa merkittävästi interaktiivisuutta. Selainta ohjelmoimalla käyttöliittymää voidaan manipuloida suoraan ja saada aikaan vastaavanlainen nopea toiminnallisuus verkossa kuin *itsenäisissä sovelluksissa* (stand-alone application). Tästä teknologiasta,

jossa selaimen JavaScript-ohjelma hakee tietoa HTTP-protokollalla ja päivittää sivun HTML-rakenteen määrittävää tietorakennetta (joka kuvataan DOM Domain Object Model standardin avulla) käytetään nimitystä AJAX (Asynchronous JavaScript And XML). Nimitys on sikäli harhaanjohtava ja vanhentunut, että AJAX ei ole sidottu vain XML:n käyttöön, vaan tiedonsiirrossa käytetään yhä useammin JSON-muotoista dataa. AJAX:a käyttämällä syntyy yhteen interaktiiviseen verkkosivuun perustuvia sovelluksia, joista käytetään nimitystä *yhden sivun sovellus* (Single Page Application, SPA).

DOM

AJAX

yhden sivun sovellus

2.6 Tiedon verkko – Web of Data

Avoimen tiedon idea ei ota kantaa siihen, miten tieto julkaistaan. Tämä voidaan tehdä teknisesti esimerkiksi Excel-taulukkona, tietokantakopiona tai toiminnallisesti palvelurajapintojen kautta. Haasteeksi jää kysymys siitä, miten tiedot oikeastaan kannattaisi jakaa, jotta ne voitaisiin vaivattomasti ottaa käyttöön eri organisaatioissa, ja miten eri tahojen tiedot voidaan yhdistää sisällöllisesti eli semanttisesti toisiinsa. Ratkaisumallin tähän tarjoaa yhdistetyn verkkotiedon (Web of Data, Linked Data) idea ja kansainvälinen Linked Data -liike. Sen ideana on esittää verkon datat ja niiden kuvailuissa (metadata) käytetyt sanastot yhteismitallisesti ns. *semanttisena verkkona* (semantic net). Tuloksena syntyy *semanttinen web* (Semantic Web), jonka tietosisällöt on esitetty tavalla, jota koneetkin ymmärtävät. Tämä mahdollistaa älykkäiden sovellusten kehittämisen laskennallisen logiikan avulla, johon semanttisen webin tiedon esittämismuodot perustuvat.[10]

semanttinen verkko

semanttinen web

Verkkomuoto on osoittautunut joustavaksi ja käytännölliseksi tavaksi esittää tietoja ja yhdistää niitä laajemmiksi kokonaisuuksiksi. Tätä varten on webin kehitystä koordinoiva W3C-järjestö standardoinut RDF-tietomallin (Resource Description Framework) ja kielen, joka on dokumenttien rakenteen kuvailuun tarkoitetun XML:n jälkeen seuraava merkittävä standardi webin standardipinossa.

RDF

[10] Semanttisen webin englannin kielisiä oppikirjoja ovat esimerkiksi [4] ja tiedon kuvaamiseen tarkemmin paneutuva [1]. W3C:n sivulta https://www.w3.org/2001/sw/wiki/Books löytyy kokoelma eri vuosina ilmestyneitä alan oppikirjoja.

RDF-malliin perustuen on kehitetty lukuisia kieliä ja standardeja *tietämyksen esittämiseen* (Knowledge Representation) ja *tietämyksen muodostamiseen* (Knowledge Discovery) webissä. Näistä tärkeimpiä ovat SPARQL kyselykieli RDF-muotoiselle datalle, SKOS (Simple Knowledge Organization System) ja OWL (Web Ontology Language) sanastojen ja ontologisten käsitejärjestelmien esittämiseen ja RIF (Rule Interchange Format) logiikan päättelysääntöjen kuvaamiseen. Semanttisesta webistä käytetään joskus nimitystä Web 3.0, mikä korostaa verkon luonteessa ja teknologisessa perustassa tapahtumassa olevaa syvällistä muutosta alkuperäisen webin (Web 1.0) ja yhteisöllisen webin (Web 2.0) jälkeen.

SPARQL
SKOS
OWL
RIF
Web 3.0

Semanttisen webin tunnetuimpana edustajana on toiminut W3C-järjestön vetäjä ja "webin isä" Sir Tim Berners-Lee. Maailmanlaajuisen yhteisöllisen työn seurauksena verkossa on nyt vapaasti saatavilla valtavan kokoinen tietoresurssien ja näitä yhdistävien linkkien semanttinen verkko. Se sisältää tuhansia toisiinsa sillattuja verkkomuotoisia aineistoja, kuten koneellisesti louhitut Wikipedian erikieliset versiot (DBpedia) ja Wikidata, Geonames-palvelun miljoonat paikkatiedot, EU:n Eurostat-tilastot, US Census Data -väestötiedot ja kirjastojen viitetietokantoja kuten Ruotsin kansalliskirjaston LIBRIS. Kun perinteinen WWW yhdistää toisiinsa web-sivuja ja dokumentteja hypertekstilinkkien avulla verkoksi (Web of Documents), tiedon semanttinen verkko (Web of Data) linkittää toisiinsa käsitteitä ja tietoja näiden välisten suhteiden kautta, esimerkiksi että Maa on planeetta ja että planeetat ovat taivaankappaleita. Globaalista tiedon verkosta käytetään nimitystä Giant Global Graph (GGG). Googlen lanseeraama nimitys dataverkosta on Knowledge Graph.

GGG

GGG:n ytimessä on joukko keskeisiä, toisiinsa yhdistettyjä avoimia *datajoukkoja* (dataset), jotka muodostavat eräänlaisen "virallisen" linkitetyn avoimen datapilven, *LOD-pilven* (Linked Open Data Cloud, LOD Cloud). Kuvassa 2.1 on esitetty LOD-pilven rakenne[11] keväällä 2017. Pilvi koostuu 1139 solmusta, jotka ovat datajoukkoja jaoteltuna sisällöllisesti muutamaan eri kategoriaan, jotka on grafiikassa ryhmitelty ja esitetty eri väreillä havainnollisuuden vuoksi. Datajoukkoja yhdistävät kaaret kertovat, mitkä datajoukot on sillattu toisiinsa. *Siltaus* (mapping) tarkoittaa kuvausta, joka kertoo, mitkä käsitteet yhdistetyissä datajoukoissa vastaavat toisiaan. Esimerkiksi maailmassa on useita London-

datajoukko
LOD-pilvi
siltaus

[11] Tietoa pilvestä ylläpidetään osoitteessa http://lod-cloud.net/, jossa kerrotaan myös pilven ylläpidosta kansainvälisen yhteisön voimin.

nimisiä paikkoja, joita löytyy LOD-pilveen kuuluvasta kansainvälisestä Geonames-paikkatietorekisteristä, mutta näistä vain yksi vastaa Wikipediassa (DBpediassa) olevaa Iso-Britannian pääkaupungin käsitettä. Tämä selviää Geonames- ja DBpedia-solmujen siltauksesta, jossa asia on esitetty samat käsitteet yhdistävillä kaarilla.

Wikipedioista automaattisesti louhittu RDF-muunnos DBpedia on LOD-pilven kaikkein keskeisin ja linkitetyin aineisto. LOD-pilven datajoukot ja siltaukset muodostavat jättiläismäisen, miljardeista soluista ja kaarista muodostuvan semanttisen RDF-verkon, josta on muodostunut maailmanlaajuisen semanttisen webin ydin. Haastajaksi DBpedialle on muodostumassa Wikipedia-järjestön itsensä koordinoima Wikidata-ohjelma[12], jossa luodaan semanttisen webin tekniikoilla kaikkien Wikipedioiden yhteisestä datasta vastaava linkitetyn datan palvelu koneille (GGG) kuin Wikipediat ovat ihmiskäyttäjille (WWW). Wikidataa kehitetään tätä kirjoitettaessa yhteisöllisesti yli 17000 kehittäjän voimin ja sen dataan sisältyy lähes 40 miljoonaa tietokohdetta, kuten *Helsinki* kaupunkina tai *Kalevala* sanan eri merkityksissä, esimerkiksi runoteoksena, asteroidina, orkesterina, musiikkilevynä, yrityksenä ja venäläisenä kylänä.

Avointen tietoaineistojen ja niiden yhdistämisen perustalle on alkanut nopeasti syntyä uusia innovatiivisia toimintamalleja ja käytännön sovelluksia, joita ollaan ottamassa käyttöön tietoyhteiskunnassa ja julkishallinnossa eri puolilla maailmaa. Monissa maissa, kuten Iso-Britanniassa, on avattu julkisen sektorin avoimen datan portaaleja[13], joiden kautta löytyy sekä avoimia tietovarantoja että näihin perustuvia sovelluksia, vaikkapa lähellä olevien koulutusmahdollisuuksien tai hoitokotien löytämiseen tai liikennetietojen seuraamiseen. Yksi tunnettu sovellus on BBC:n kotisivujen verkkopalvelut, joissa organisaation eri tahojen mediasisältöjen yhdistämiseen käytetään Wikipedian semanttisen webin muunnosta DBpediaa.[14]

Suomessa semanttista webiä on kehitetty laajimmin kansallisessa semanttisen webin FinnONTO-hankesarjassa[15] (2003–2012), jossa kehitettiin *FinnONTO*

[12]Hankkeen kotisivu on https://www.wikidata.org/ ja kerätty data on avoimesti käytettävissä rajapinnan kautta osoitteessa https://query.wikidata.org/

[13]https://data.gov.uk/

[14]http://www.bbc.co.uk/academy/technology/software-engineering/semantic-web

[15]http://seco.cs.aalto.fi/projects/finnonto/

kansallisen semanttisen webin sisältöinfrastruktuurin prototyyppi ONKI.fi ontologiapalveluineen[16] ja useita hajautettuun sisällöntuotantoon perustuvia semanttisia portaaleja, kuten MuseoSuomi[17] (2004), TerveSuomi[18] (2008), Kulttuurisampo[19] (2008) ja Kirjasampo[20] (2011). Aaltoyliopiston ja Helsingin yliopiston kehittämä ONKI.fi-prototyyppi otettiin koekäyttöön mm. monissa maamme museoissa yhdistämällä palvelu osaksi kokoelmanhallinnan järjestelmiä. Valtiovarain- ja opetus- ja kulttuuriministeriön rahoituksella ONKI:n keskeisiä osia tuotteistettiin ja otettiin käyttöön 2014 Kansalliskirjaston ylläpitämänä Finto.fi-palveluna[21] (2014). Kirjasampoa ylläpitävät ja kehittävät nykyisin maamme yleiset kirjastot (Kirjastot.fi). Vuonna 2016 Kirjasammossa kävi 1,6 miljoonaa vierailijaa ja se on Finto.fi-palvelun ohella ollut käytetyin suomalainen semanttisen webin sovellus.

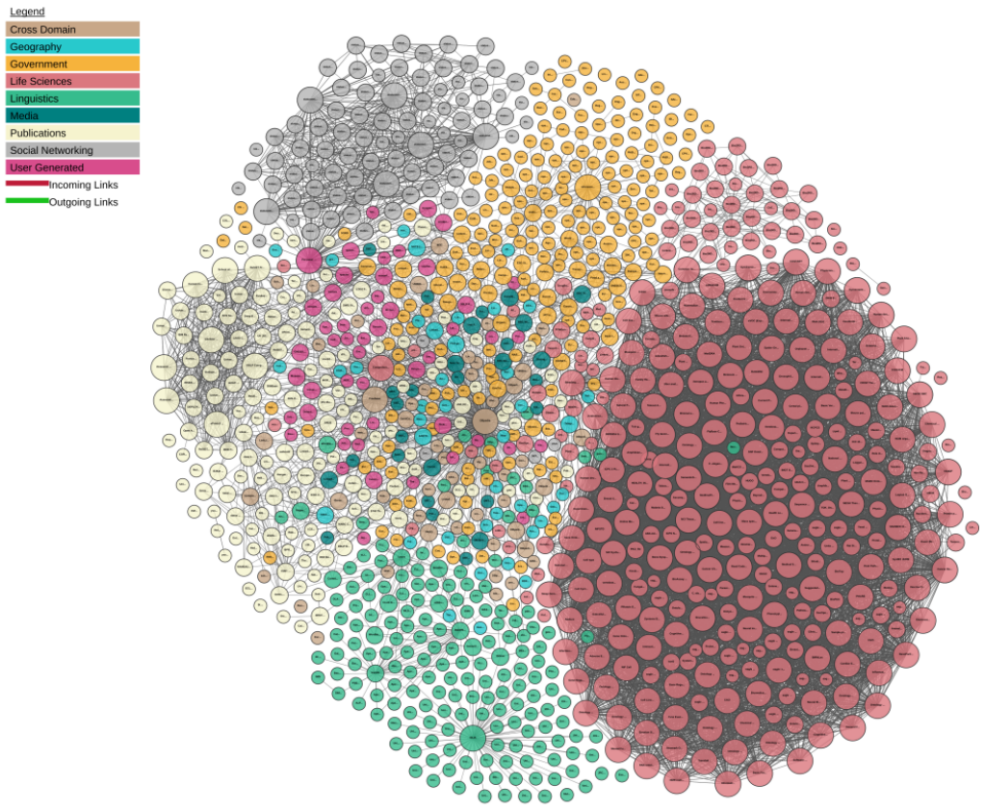
FinnONTO:ssa käynnistynyt tutkimustyö on jatkunut avoimen Linked Data Finland -datajulkaisualustan[22] kehittämisellä ja siihen perustuviin sovelluksiin liittyen. Näitä ovat mm. talvi- ja jatkosodan aineistoja julkaiseva Sotasampo.fi[23] (2015) ja oikeusministeriön Finlex-lakiaineistoihin perustuva Semanttinen Finlex[24] (2016), joka julkaisee keskeisimmät osat lainsäädännöstämme ja oikeustapauksia avoimena linkitettynä datana. Molempien sovellusten semanttiseen verkkoon kuuluu nykyisin miljoonittain käsitteitä ja näiden välisiä yhteyksiä, ja niiden data on saatavilla avoimesti Linked Data Finland -palvelusta. Sotasammossa on esimerkiksi sekä tietoa tuhansista sodanajan tapahtumista ja kymmenistä tuhansista luovutetun alueen paikoista historiallisilla kartoilla. Järjestelmään on yhdistetty Kansallisarkiston tuottamat tiedot kaikista viime sodissamme menehtyneistä n. 95 000 sotilaasta ja tuotu eri lähteistä tietoja tuhansista muista sodasta selvinneistä tunnetuista sotilaista. Puolustusvoimien SA-Kuva-arkistosta on saatu n. 160 000 autenttisen sota-ajan valokuvan kokoelma. Aineistoja on sillattu toisiinsa ja ulkoisiin aineistoihin, kuten Suomen Sotahistoriallisen Seuran verkossa julkaisemien Kansa Taisteli

[16] http://onki.fi
[17] http://www.museosuomi.fi
[18] https://seco.cs.aalto.fi/applications/tervesuomi/
[19] http://www.kulttuurisampo.fi
[20] http://www.kirjasampo.fi
[21] http://finto.fi
[22] http://ldf.fi
[23] http://sotasampo.fi
[24] http://data.finlex.fi

-lehtien (1957–1986) tuhansiin muisteluartikkeleihin.

Yksi kiinnostava kehityssuunta jatkossa on kieliteknologian yhdistäminen semanttisen webin menetelmiin ajatuksena kehrätä webin semanttista verkkoa automaattisesti tekstiaineistosta. Yhtenä esimerkkinä tästä on elämäkertoihin liittyvä sovellus Vanhat norssit semanttisessa webissä (2017)[25] ja Semanttinen Kansallisbiografia, jossa Suomalaisen Kirjallisuuden Seuran toimittamat yli 13 000 elämäkertaa rakenteistetaan ja rikastetaan semanttisesti verkon data-aineistojen kautta[26].

[25] http://www.norssit./semweb/
[26] Ks. artikkeli [24]



Kuva 2.1: Linked Open Data -pilven ydin koostui keväällä 2017 1139 datajoukosta, jotka näkyvät kuvassa erivärisinä palloina sen mukaan, minkä aihealueen datasta on kysymys (esimerkiksi “geography”, “government”, “life sciences”, “linguistics” jne.; ks. vasemman ylänurkan selite). Datajoukkojen väliset kaaret kuvaavat siltauksia. Pilven keskellä on erikielisistä Wikipedioista louhittu semanttinen verkko DBpedia.

Luku 3

Tiedonhaun haasteita

Ihmiskäyttäjän näkökulmasta webillä on kaksi pääasiallista käyttötapaa tiedon etsimisessä:

1. *Haussa* (search) käyttäjä muotoilee tietotarpeensa *kyselyksi* (query) interaktiivisen hakujärjestelmän haluamaan muotoon ja saa kyselyn vastauksena joukon tietosisältöjä, kuten linkkejä web-sivuille. *haku* *kysely*

2. *Selailussa* (browsing) webin sisältöihin tutustutaan assosiatiivisesti sivuja yhdistäviä linkkejä seuraten. *selailu*

Tiedonhaku on tietojenkäsittelytieteen ja informaatiotutkimuksen osa-alue, jossa tutkitaan tiedon esittämistä, tallettamista ja etsimistä. Tässä luvussa esitellään ensin perusteita tekstihausta, jota käytetään perinteisissä hakujärjestelmissä ja webin hakukoneissa. Tämän jälkeen tarkastellaan sekä tekstihaun että selailun haasteita käyttäjän kannalta. Ratkaisumalliksi moniin ongelmiin on tullut tiedon esittäminen semanttisissa muodoissa, joita tietokone kykenee tulkitsemaan tekstiä paremmin. Älykkäiden web-palveluiden kehittämisen edellytyksenä voidaan pitää sitä, että kone osaa tulkita tai ”ymmärtää” ainakin jossain mielessä niitä sisältöjä, joita se käsittelee. [1] *tiedonhaku*

[1] Tiedonhaun tutkimusalaa ja menetelmiä on esitelty tarkemmin esimerkiksi teoksessa [38], josta on saatavilla oppimateriaalia myös vapaasti verkossa: https://nlp.stanford.edu/IR-book/

3.1 Perinteinen tekstihaku

Käytetyin hakutapa verkossa on edelleen Google-tyyppinen tekstikenttä, johon kysely kirjoitetaan joukkona hakutermejä. Haun tuloksena palautetaan joukko dokumentteja (linkkejä web-sivuihin) järjestettynä siten, että *olennaisimmat* (relavance) ts. parhaat osumat tulevat ensin.

olennaisuus

tekstihaku

Perinteinen *tekstihaku* (text search) toteutetaan niin, että haun kohteena olevista dokumenteista, kuten web-sivuista, muodostaan *käänteinen hakemisto* (inverted index), jossa jokaisen hakusanan kohdalla on lueteltu, missä dokumenteissa sana esiintyy. Hakusanaa vastaavat dokumentit voidaan silloin palauttaa nopeasti vain hakemistoa katsomalla.

käänteinen hakemisto

Tarkastellaan esimerkkinä kolmea tekstidokumenttia:

Dokumentti	**Teksti**
D1	Kissalla on kala.
D2	Kissa syö katolla kalaa.
D3	”Kuuma kissa katolla”on elokuva.

Taulukko 3.1: Haun kohteena oleva dokumenttijoukko

Välimerkit poistamalla ja sanat perusmuotoistamalla näistä syntyy talukon 3.2 käänteisindeksi.

Termi	**Esiintyy dokumentissa**
kissa	D1, D2, D3
olla	D1, D3
kala	D1, D2
syödä	D2
katto	D2, D3
kuuma	D3
elokuva	D3

Taulukko 3.2: Taulukon 3.1 dokumenttien kääteisindeksi

Useamman hakutermin kyselyn tulos voidaan laskea termejä vastaavien dokumenttijoukkojen leikkauksen avulla. Haettaessa esimerkiksi dokumentteja, joiden aiheena on sekä *kissa* ja *katto*, saadaan tulokseksi dokumentit D2 ja D3:

$$\{D1, D2, D3\} \cap \{D2, D3\} = \{D2, D3\} \tag{3.1}$$

Tuloksen dokumenttien järjestämiseen jonkun paremmuuskriteerin mukaan on kehitetty lukuisia menetelmiä. Esimerkiksi paljon käytetyssä TF-IDF-menetelmässä painotetaan dokumentteja, joissa hakusana esiintyy useaan kertaan ja on harvinainen koko dokumenttijoukossa. Tässä menetelmässä käänteisindeksiin lisätään tietoa myös sanojen esiintymismääristä eri dokumenteissa. Googlen lanseeraama Page-Rank-algoritmi taas hyödyntää web-sivujen välisiä linkkejä ja antaa korkean arvon sellaisille sivuille, joihin viitataan monilta muilta sivuilta ja joiden oma arvo on myös korkea.

Haun laatua mitataan yleensä *tarkkuudella* (precision) ja *saannilla* (recall). Oletetaan, että oikea hakutulos ts. kyselyä vastaava dokumenttijoukko (relevant documents) on *Oikein* ja haun tuloksena palautuu dokumenttijoukko *Tulos* (retrieved documents). Tarkkuus kertoo, miten suuri osa tuloksesta meni oikein

tarkkuus

saanti

$$Tarkkuus = |(Tulos \cap Oikein)|/|Tulos| \tag{3.2}$$

ja saanti, kuinka suuri osuus kaikista oikeista vastauksesta jäi haaviin:

$$Saanti = |(Tulos \cap Oikein)|/|Oikein| \tag{3.3}$$

Merkintä $|X|$ tarkoittaa joukon X kokoa.

Haun kannalta keskeinen haaste on, miten muotoilla omaa tiedon tarvetta vastaava kysely niin, että tuloksen tarkkuus ja saanti olisivat mahdollisimman hyviä *samanaikaisesti*. Esimerkiksi 100% saanti olisi helppoa toteuttaa hakemalla aina tulokseen kaikki web-sivut, mutta silloin tarkkuus olisi tietysti surkea. Hakua rajaamalla tarkkuus yleensä paranee, mutta saanti tyypillisesti huononee, sillä tarkka kysely rajaa helposti vahingossa pois myös oikeita hakutuloksia.

3.2 Haun haasteita

Edellä hahmoteltuun perinteiseen tekstihakuun liittyy monia haasteita ja kehitysmahdollisuuksia. Seuraavassa on lueteltu joitain perustavaa laatua

olevia haasteita, joita hakujärjestelmissä joko hoidetaan jollain tavalla tai jätetään huomiotta:

- *Synonyymit.* Dokumentissa voidaan viitata samaan käsitteeseen monella eri tavalla, joista vain yksi vastaa kyselyä. Esimerkiksi Venus-planeetasta voidaan puhua myös Ilta- tai Aamutähtenä. Synonyymiset ilmaisut huonontavat haun saantia.

homonymia

- *Homonyymit. Homonymialla* (homonymy) tarkoitetaan termin monimerkityksisyyttä: samalla ilmaisulla voi olla samanaikaisesti useita eri merkityksiä. Esimerkiksi ”varkaudella” voidaan viitata rikokseen tai kaupunkiin, ”nokialla” taas yritykseen, kaupunkiin, nokinäätään eli soopeliin tai henkilöön F. E. Sillanpään romaanissa *Ihmiset suviyössä*. Nimet, esimerkiksi paikan ja henkilöiden nimet ovat usein homonymisia. Suomessa on esimerkiksi 49 eri Pyhäjärveä Maanmittauslaitoksen Paikannimirekisterin mukaan, ja Pyhäjärvi on myös sukunimi. Homonymista termiä, jonka merkitykset liittyvät toisiinsa, kutsutaan *polysemiseksi* (polysemy). Esimerkiksi ”päällä” voidaan viitata nuolen tai ihmisen päähän. Homonymia huonontaa haun tarkkuutta.

polysemia

- *Monikielisyys.* Erikielisissä dokumenteissa samoihin sisältöihin viitataan erikielisin termein. Lisäksi nimet kirjoitetaan eri kielissä käyttäen hyväksi erilaisia translitterointisääntöjä ja merkistöjä. Esimerkiksi säveltäjä Pjotr Tšaikovskin nimi kirjoitetaan englannissa muodossa Pyotr Tchaikovsky ja alkukielessä kyrillisin kirjaimin. Edellä kuvattu tekstihaku ei sellaisenaan huomioi monikielisyyteen liittyviä haasteita.

- *Taivutusmuodot.* Englannissa sanojen taivutus on yksinkertaista ja voidaan huomioida haussa helpohkosti, mutta monissa kielissä kuten suomessa sanat taipuvat rikkaasti ja niistä voidaan johtaa uusia sanoja. Esimerkiksi hakusanalla ”yö” ei löydy dokumentti, jossa puhutaan ”öisestä kuhanuistelusta”. Hypätä-verbi voi esiintyä paitsi taivutusmuodoissaan, myös lukuisina johdoksina ja näiden taivutusmuotoina: hypyttää, hypäytin, hypellä jne. Lisäksi tulevat vielä päätteet (esimerkiksi hypyttääköhän). Kelvollinen tekstihaku edellyttää sekä dokumenteissa että kyselyissä esiintyvien sanojen palauttamista perusmuotoon eli *lemmausta* (lemmatization) tai vä-

lemmaus

hintään taipuneiden päätteiden poistamista eli *stemmausta* (stemming). *stemmaus*

- *Käsitteiden väliset suhteet.* Hakusanojen välillä on semanttisia suhteita, joita perinteinen tekstihaku ei huomioi. Töölössä oleva pizzeria ei löydy haettaessa helsinkiläisiä ravintoloita, vaikka tiedämme Töölön olevan Helsingin osa ja pizzerian eräänlainen ravintola. Savu-sanalla ei löydy tulesta kertovia sivuja (ellei sivulla esiinny myös sana tuli), vaikka tiedämme, ettei ole savua ilman tulta.

- *Erilaiset hakukohteet.* Perinteisen haun kohteena ovat dokumentit, mutta monessa tapauksessa käyttäjän tiedontarpeena ei ole dokumenttien hakeminen sinänsä. Tavoitteena voi olla vaikkapa käsitteiden välisten yhteyksien hakeminen, jolloin puhutaan *yhteyshausta* (relational search). Halutaan esimerkiksi tietää, millä tavoilla Jean Sibelius liittyy Hämeenlinnaan[2] tai Akseli Gallen-Kallela Mannerheimiin[3]. *yhteyshaku*

- *Ongelmanratkonta.* Viimekädessä tiedon haussa on yleensä kysymys ongelmanratkonnasta, johon dokumenttien haku on vain yksi epäsuora väline. Halutaan esimerkiksi tietää, paljonko painaa kilo höyheniä kuussa, paljonko asukkaita on Togossa tai että kuka on presidentti Halosen tyttären isä. Painopiste hakukoneiden kehittämisessä on siirtymässä dokumenttien hausta kohti käyttäjän tiedontarpeeseen liittyvän ongelman ratkontaa. Askeleita tähän suuntaan ovat esimerkiksi hakukoneet `www.wolframalpha.com` ja `www.ask.com`.

- *Personointi ja käyttökonteksti.* Hakukoneen olisi myös syytä ottaa huomioon käyttäjänsä erityistarpeet ja toiveet eikä välttämättä toimia samalla tavalla kaikille eri käyttäjille vauvasta vaariin. Samaten käyttöympäristöllä ja ajalla on vaikutusta hakuun. Jos esimerkiksi etsitään ajanviettomahdollisuuksia Helsingissä, pitäisi uimarantojen relevanssin olla pienempi talvella kuin kesällä. Hakukoneet käyttävät hyväksi tietoa käyttäjistään ja heidän aiemmasta *personononointi* *käyttökonteksti*

[2]Sibelius syntyi vuonna 1865 Hämeenlinnassa, ja siellä on monia häneen liittyviä kulttuurikohteita ja tapahtumia.

[3]Gallen-Kallela esimerkiksi nimitettiin Mannerheimin toimesta pääesikunnan kirjapainon esimieheksi ja hän toimi Mannerheimin adjutanttina.

toiminnastaan hakukoneen personointiin ja kohdennettuun markkinointiin. Samoin käyttöpaikka kuten maa ja sen kieli voidaan huomioida haussa ja hakutuloksia järjestettäessä.

Kuva 3.1: Museoesineen tietoja kokoelmassa

Tarkastellaan esimerkkinä hakua museokokoelmasta, johon on talletettu mm. kuvan 3.1 mukainen esine Kansallismuseosta. Tehdään kokoelmaan seuraavat kyselyt:

1. Hae kaikki esineet.
2. Hae kaikki keramiikka-astiat.
3. Hae esineet, jotka on valmistettu Euroopassa.
4. Onko tämä kuppi ja lautanen Meissenin valmistama?

Tällaiset kyselyt tehdään webissä tyypillisesti lomakkeella, jossa eri kentille, tässä *museo*, *objekti*, *materiaali*, *valmistuspaikka* ja *valmistaja*, on oma tekstikenttänsä hakua varten. Kysely (2) esimerkiksi tehdään niin, että objekti-kenttää kirjoitetaan ”astia” ja materiaali-kenttään ”keramiikka”. Perinteistä tekstihaku käyttämällä ei mihinkään kyselyyn 1–3 kuitenkaan saada vastuksena kuvan esinettä, vaikka ihmishakija niin luulee jokaisessa tapauksessa: Kyselyssä (1) kone ei ymmärrä kohteen olevan esine, vaikka se on kuppi ja lautanen; kyselyssä (2) kone ei älyä posliinin olevan keramiikkaa eikä kupin tai lautasen olevan astioita; kohdassa (3) kone ei ymmärrä Saksan kuuluvan Eurooppaan. Kyselyyn (4) oikein vastaaminen edellyttäisi hakukoneelta tietoa siitä, että valmistaja Meissen ei viittaa Meissenin kaupunkiin vaan samannimiseen tehtaaseen, joka tässä tapauksessa kyllä sijaitsee Meissenin kaupungissa.

Oma ongelma-alueensa hakumenetelmissä on hakukyselyjen muodostaminen. Erillisten hakusanojen kirjoittaminen hakukenttään on yksi usein

toimiva yksinkertainen ratkaisu, kuten esimerkiksi Googlen menestyksekäs käyttöliittymä on osoittanut. Moniin käyttötapauksiin on kuitenkin kehitetty parempiakin ratkaisuja, kuten hakujen muotoilu hierarkkisista hakufaseteista valintoja tehden ns. *fasettihaussa* (faceted search) ja erilaiset graafiset hakukäyttöliittymät, kuten karttakäyttöliittymät paikkatietoa sisältävän tiedon haussa.

fasettihaku

Myös hakutulosten esittäminen voidaan tehdä eri tavoin. Tuloslista ei aina ole tarkoituksenmukaisin tapa, vaan tilanteesta riippuen on tarpeen käyttää muitakin tapoja tulosjoukon hahmottamiseen käyttäjälle. Voidaan esimerkiksi käyttää karttoja paikkatietoa haettaessa, aikajanoja ajasta riippuvan tiedon esittämiseen, tuloksia voidaan ryhmitellä aineistotyypin mukaan tai käyttää bisnesgrafiikan visualisointeja.

3.3 Selailun haasteita

Myös web-sivujen selailussa törmätään usein vaikeuksiin:

- *Eksyminen.* Webin linkkien valtamereen eksyy helposti (“lost in hyperspace”). Kokonaisuuksien hahmottaminen on vaikeaa pirstaleisessa laajassa tietomassassa linkkien kautta hyppelemällä, vaikeampaa kuin perinteisissä lineaariseen rakenteeseen perustuvissa kirjoissa.[4]

- *Linkkien vanheneminen.* Linkit vanhentuvat ja tuhoutuvat, sillä tiedon päivittyessä linkit eivät välttämättä päivity, vaan jäävät osoittamaan vanhaan tietoon. Esimerkiksi valtion tai yrityksen presidenttiin viittaava linkki voi unohtua osoittamaan vaalien jälkeen virastaan jo väistyneen henkilön sivulle.

- *Sivujen vanheneminen.* Linkitetyt kohdesivut vanhentuvat tai poistuvat kokonaan. Uudet sivut eivät linkity vanhoihin sivuihin ja vanhat sivut eivät linkity tai päivity uusiin automaattisesti.

- *Tiedon ja toimijoiden luotettavuus.* Luottamus (web of trust) tai pikemminkin sen puute on yhä keskeisempi ongelma webissä. Litteä maa -järjestön sivusto[5] ei sisältäne yhtä luotettavaa tietoa kuin

[4] Aihetta on tutkittu mm. hypertekstijärjestelmien yhteydessä, ks. esimerkiksi [17].
[5] http://www.theflatearthsociety.org/cms/

vaikkapa Nature-lehden tiedesivut. Paljon on keskusteltu ja verrattu yhteisöllisesti tuotetun Wikipedian ja toimituksellisten tietosanakirjojen kuten Encyclopedia Britannican luotettavuutta.

Hakua ja selailua joudutaan käyttämään paitsi erikseen myös samanaikaisesti sellaisessa käyttötilanteessa, jossa ei tarkkaan tiedetä, mitä ollaan etsimässä, vaan ollaan pikemminkin perehtymässä johonkin asiakokonaisuuteen. Jos käyttäjällä ei esimerkiksi ole hyvää käsitystä verkossa olevan museon kokoelmatietokannan sisällöstä tai täsmällistä tiedonhaun tavoitetta, järkevien hakusanojen keksiminen Googlen kaltaiselle hakukoneelle voi olla vaikeaa. Silloin olisi voitava löytää kokoelmia selailemalla paitsi yksittäisiä hakukohteita, myös hakea ja hahmottaa näistä muodostuvia kokonaisuuksia.

3.4 Ratkaisuna tiedon semantiikka

Semanttisen webin ideana on esittää verkon tieto systemaattisessa rakenteisessa muodossa, joka kuvaa verkon sisältöjä niin, että tietokone pystyy niitä tulkitsemaan algoritmisesti. ”Semanttinen web” tarkoittaa siis koneellisesti tulkittavissa olevaa webiä tai webiin upotettua datakerrosta. Tässä yhteydessä puhutaan usein *metatiedosta* eli *metadatasta* (metadata), joka on tietoa tiedosta, mutta rajanveto datan ja metadatan välillä ei aina ole selkeää, sillä metadatallakin viitataan web-ympäristössä koneellisesti tulkittavaan dataan. Termillä metadata viitataan yleensä koneellisessa muodossa olevaan tietoon, metatiedolla taas laaja-alaisemmin myös ihmisen ymmärrykseen.

metadata

Tiedon merkitystä tutkitaan *merkitysopissa* eli *semantiikassa* (semantics), jossa selvitetään *symbolisten ilmausten* (signifier) ja niillä viitattavien *tarkoitteiden* (denotation) välisiä suhteita. Semantiikkaa tutkitaan erilaisilla lähtökohdilla, menetelmillä ja tavoitteilla monilla eri aloilla, kuten kielitieteessä, filosofiassa, matemaattisessa logiikassa, semiotiikassa ja tietojenkäsittelytieteessä.

merkitysoppi

tarkoite

Webissä olevan tiedon semanttinen esittäminen mahdollistaa tarttumisen edellä kuvattuihin tekstihaun haasteisiin. Synonyymien, homonyymien ja monikielisyyden ongelman ratkaisumallina on erottaa toisistaan merkitykset eli käsitteet ja niihin liittyvät erilaiset kielelliset nimikkeet ja esittää ne tietorakenteina. Lisäksi käsitteiden merkitystä voidaan määritellä

ja kuvata toisten käsitteiden avulla. Tällaisia käsitteitä toistensa avulla määritteleviä rakennelmia kutsutaan semanttisessa webissä ja tietojenkäsittelyssä *ontologioiksi* (ontology). Yleisemmin ontologialla tarkoitetaan filosofian haaraa, joka tutkii olemisen olemusta. Filosofian ontologiassa voidaan esimerkiksi yrittää todistaa Jumalan olemassaoloa.

ontologia

Suomessa semanttisen webin ontologioiden systemaattinen kehittäminen aloitettiin v. 2003 kansallisen FinnONTO-hankesarjan (2003–2012) puitteissa. Työn tuloksena syntyi mm. Kansalliskirjaston Yleisestä suomalaisesta asiasanastosta (YSA) ja 14 muusta eri aloilla käytetystä asiasanastosta muokattu KOKO-ontologia, joka sisältää kymmeniä tuhansia käsitteitä[6]. Esimerkiksi kissan käsitteelle on määritelty tunniste p37252 ja nimikkeet suomeksi (”kissa”), ruotsiksi (”katt”), englanniksi (”cat”) ja latinaksi (”Felis silvestris catus”). Lisäksi on vielä vaihtoehtoisia nimikkeitä kuten ”kotikissa” ja ”kesykissa” suomeksi ja ”Felis catus” latinaksi. Alla oleva kuvankaappaus esittää kissa-käsitteen nimiketiedot ONKI-palvelussa, joka toteutettiin ontologisten käsitteiden hakua, selailua ja koneellista käyttöä varten:

Kuva 3.2: Kissa-käsitteen nimiketiedot KOKO-ontologiassa ONKI-ontologiapalvelussa

Kun käytettävissä on ontologia, voidaan tietoa indeksoida sen käsitteiden kieliriippumattomilla tunnisteilla. Kun tietoa myöhemmin haetaan, voidaan hakutermit muuttaa niitä vastaaviksi käsitteiksi käytetystä kielestä riippuen ja tehdä näin erikieliset haut samalla tavalla tunnisteiden avulla. Kieliriippumattoman semantiikan avulla voidaan näin ylittää kielimuureja.

[6] Asiasanastojen muuttamista ontologiaksi on käsitelty suomeksi julkaisussa [42]: https://www.doria.fi/handle/10024/96825.

kyselyn laajentaminen

Keskeinen ontologioiden etu on, että kyselyä voidaan *laajentaa* (query expansion) käsitteeseen liittyvien synonyymien ja semanttisten suhteiden kautta. Esimerkiksi ”kesykissa” hakusanalla voidaan hakea samalla dokumentteja, joissa esiintyy termi ”kissa”, ”kotikissa”, tai englanninkielisten tekstien osalta ”cat”. Homonyymien osalta menettely mahdollistaa monimerkityksisten termien tunnistamisen sellaisina käsitteinä, joilla on yhteinen samankielinen nimike. Esimerkiksi ”johtaminen” voi tarkoittaa sähkön johtamista, yrityksen johtamista, matemaattisen kaavan johtamista tai tien johtamista jonnekin. Kriittinen kysymys on, voidaanko monimerkityksisen termin eri merkitykset eri konteksteissa – esimerkiksi sähkötekniikkaa käsittelevässä artikkelissa tai museoesineen metatietojen tietyn kentän arvona – erottaa eli *disambiguoida* (disambiguate).

disambiguointi

Esimerkiksi kuvassa 3.1 termi ”Meissen” voi viitata joko keramiikkatehtaaseen tai kaupunkiin Saksassa. Koska termi esiintyy tässä tapauksessa valmistaja-kentässä, täytyy kyseessä olla tehdas eikä kaupunki. Vapaassa tekstissä Meissen-sanan disambiguointi olisi lähtökohtaisesti vaikeampaa, koska se edellyttäisi käyttökontekstin analysointia.

Kuva 3.3: MuseoSuomi disambiguoi haketermin Nokia merkitykset: Nokia valmistajana (yrityksenä), valmistus- tai käyttöpaikkana (kaupunkina) sekä eri valmistajina, joiden nimen osana on merkkijono ”Nokia”.

Disambiguointia tarvitaan toisaalta dokumenttien indeksoinnissa, toisaalta kyselytermien merkityksen selvittämisessä. Jos jompikumpi tai molemmat jää tekemättä, hakutuloksen tarkkuus heikkenee, sillä mukaan tulee mukaan helposti vääriä osumia. Esimerkiksi MuseoSuomi-järjestelmässä[7] museokokoelmien kohteiden merkitykset on disambiguoitu ja tieto on indeksoitu merkityksen perusteella. Kun käyttäjä kirjoittaa tekstihakukenttään monimerkityksisen termin kuten ”Nokia”, ei kone voi

[7] http://www.museosuomi.fi

tietää mihin merkityksen käyttäjä sanalla viittaa, mutta koska merkitykset on erotettu niiden nimikkeistä, niin systeemi tunnistaa mahdolliset eri merkitykset ja voi pyytää käyttäjältä apua disambiguointiin. Esimerkiksi kuvassa 3.3 käyttäjä on kirjoittanut ”Nokia” käsitehaku-kenttään, jonka alle kone on tulostanut kyselyn mahdolliset tulkinnat ja valintoja vastaavien tulosjoukkojen koot suluissa. Viimeinen vaihtoehto tarkoittaa tulkintaa, jossa ”Nokia” viittaa *Nokian Kutomo ja Värjäys* -nimiseen osakeyhtiöön esineen valmistajana. Linkkiä painamalla tulokseksi tulee tarkasti vain kaksi tekstiiliä eikä esimerkiksi mobiililaitteita tai kumisaappaita, joita löytyy muiden vaihtoehtojen kautta.[8]

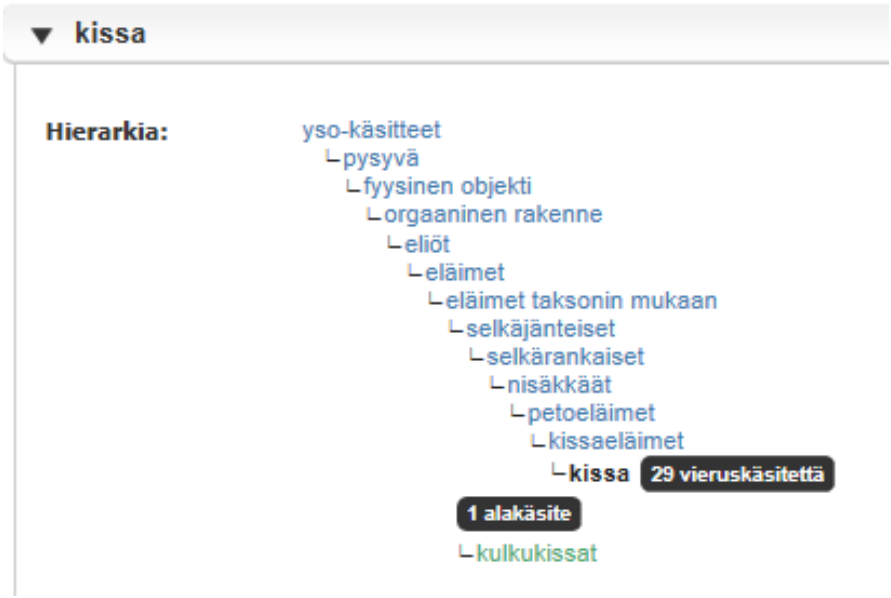
Kuva 3.4: Hakutulos valinnalla Nokia → Nokian Kutomo ja Värjäys Oy

MuseoSuomen tapauksessa kohteiden kuvailussa käytetyt merkitykset disambiguoitiin puoliautomaattisesti aineiston tuotantoprosessin aikana niin, että kone tunnisti monimerkityksiset (tai ontologiaan kuulumattomat) ilmaukset ja antoi ihmisluettelijan tehdä korjaukset niiltä osin, kun disambiguointia ei voitu tehdä automaattisesti. Yleisessä tapauksessa vapaan tekstin käsittelyssä termien semanttinen disambiguointi on haasteellista ja aktiivisen tutkimuksen kohteena.

[8]MuseoSuomen ideaa ja ominaisuuksia on selostettu tarkemmin viitteessä [25]. Sovellus sai v. 2004 Japanissa kansainvälisen Semantic Web Challenge -teknologiapalkinnon.

luokka
yksilö

Käsitteiden määrittely ja niiden välisten suhteiden esittäminen tehdään semanttisessa webissä ns. ontologioiden avulla[9]. Silloin käsitteet esitetään tyypillisesti *luokkina*, jotka edustavat *yksilöidensä* joukkoja. Esimerkiksi *lautanen*-luokka edustaa kaikkien yksittäisten lautasten joukkoa, ja *Urho Kekkonen* on *ihminen*-luokan yksilö. Käsitteisiin liitetään nimiketiedon – esimerkiksi merkkijono ”lautanen”suomeksi tai ”talrik”ruotsiksi – ohella semanttisia suhteita toisiin käsitteisiin, jolloin tuloksena syntyy verkkomaisia, yleensä hierarkkisia rakenteita. Keskeinen hierakkinen suhde on alaluokka-yläluokkasuhde, jolla voidaan esimerkiksi kertoa, että kupit ja lautaset (alaluokkia) ovat astioita (yläluokka), jotka taas ovat esineitä (*astia*-luokan yläluokka). Kuvassa 2.5 on esimerkkinä esitetty *kissa*-käsitteen yläluokat ja alaluokka KOKO-ontologiassa ONKI-ontologiapalvelimen näyttämänä. Käsitteellä on lisäksi 29 vieruskäsitettä kuten *leijona*.



Kuva 3.5: Kissa-käsitteen yläkäsitteitä ja alakäsite KOKO-ontologiassa.

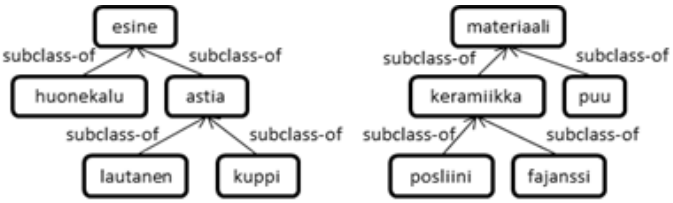
Käsitteiden väliset hierarkkiset suhteet mahdollistavat semanttisessa

[9]Käsitteitä määrittelevät ontologiat ovat semanttisen webin keskeinen elementti, joita “ymmärtävät” niin ihmiset kuin koneetkin. Ontologioita esitellään tarkemmin myöhemmissä luvuissa.

haussa *kyselyn laajentamisen* (query expansion) uusiin suuntiin. Haku voidaan suoritettaa paitsi annetulla hakutermillä ja synonyymeillä, kuten edellä, myös termiin hierarkkisesti liittyvillä muilla termeillä. Esimerkiksi kissa-käsitteen luokkahierarkia avulla voidaan petoeläin-termi laajentaa sen alaluokkiiin, kuten kissaeläimiin, kissoihin ja kulkukissoihin, jolloin haun saanti paranee. Mahdollista kuitenkin on tässäkin tapauksessa, että tarkkuus huononee saannin kustannuksella, jos esimerkiksi tiedon tarpeena onkin vain petoeläimiin yleisesti liittyvät dokumentit, eikä niinkään erilaisiin petoeläimiin liittyvä tieto.

kyselyn laajentaminen

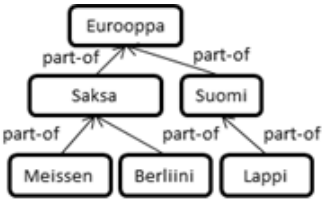
Tarkastellaan esimerkkinä kyselyn laajentamisesta kuvan 3.1 tapausta, joissa tekstihaku ei löytänyt esinettä museon kokoelmatiedoista. Jos käytettävissä on esineet ja materiaalit hierarkkisesti kuvaavat ontologiat ja kyselyt laajennetaan alaluokkiin päin (ks. kuva 3.6), kupit ja lautaset voidaan nyt löytää esineitä haettaessa ja tunnistaa posliini eräänlaisena keramiikkana (esimerkit 1 ja 2).



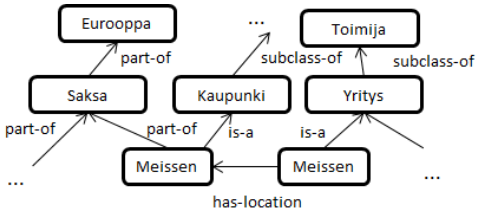
Kuva 3.6: Esine- ja materiaalityyppien luokkahierarkiat

Toinen sovelluksissa paljon käytetty semanttinen suhde on osa-kokonaisuus-suhde (part-of), jolla voidaan esimerkiksi kertoa, että Meissenin kaupunki on osa Saksaa, joka on osa Eurooppaa (kuva 3.7). Näin löytyy Saksa Euroopan osana, ja hakuesimerkin 3 haaste löytää Saksa Eurooppa-hakusanan kautta ratkeaa.

Ontologioissa on usein tärkeää erottaa yksilökäsitteen käsiteluokkaan kuulumista ilmaiseva suhde (is-a) yläluokkasuhteesta (subclass-of). Esimerkiksi toimijaontologiassa olevat yksittäiset yritykset tai henkilöt eivät ole *luokkia* (class, type) vaan yritys- ja henkilöluokan *yksilöitä* (individual, instance). Kuvan 3.8 ontologiassa solmu Meissen oikealla on Yritys-luokan yksilökäsite ja vasemmalla Kaupunki-luokan; ontologia voi näin erottaa samannimisen yrityksen kaupungista.



Kuva 3.7: Paikkaontologia, jossa kuvataan alueellista kuuluvuutta.



Kuva 3.8: Ontologia, jossa on erotettu Meissen-sanan merkitykset kaupunkina ja yrityksenä. Assosiatiivinen suhde has-location kertoo yrityksen sijainnin kaupungissa.

Tämän suhteen avulla on tarvittaessa mahdollista päätellä (sopiva päättelysääntö muotoilemalla), että Meissen-yrityksen valmistamat esineet on valmistettu Meissenin kaupungissa ja löytää yleisemmin eri paikoissa valmistetut esineet valmistajatiedon kautta.

yksilöhaku

Rakenteinen tieto mahdollistaa myös haun kohdistamisen dokumenttien ohella niiden sisältämän datan osiin. *Yksilöhaussa* (entity search) esimerkiksi haetaan yksilökohteista eli entiteeteistä kertovien dokumenttien sijasta itse yksilöiden kuvauksia, kuten paikkoja, henkilöitä, yhtyeitä, teoksia jne. Esimerkiksi haku termillä *Meissen* voisi antaa tuloksena ontologioissa olevat kuvaukset Meissenin kaupungista ja yrityksestä. Idea on jo käytössä hakukoneissa. Jos esimerkiksi kirjoittaa Googlen hakukenttään ”Who is Kekkonen” saa tulokseksi entiteetin Urho Kekkonen tietolaatikossa eikä vain listausta dokumenteista, joissa esiintyvät sanat ”who”, ”is” ja ”Kekkonen”. Hakukysymykseen ”Who is Kekkonen’s spouse” tulee vastaukseksi entiteetti ”Sylvi Salome Uino”.

Haku voidaan kohdistaa objektien ohella myös niiden välisten suhteiden

hakuun. Kuvassa 3.9 on esimerkki Kulttuurisampo-järjestelmän[10] *yhteyshausta*, jossa haun kohteena ovat kohteiden, tässä historiallisten henkilöiden, väliset yhteydet. Käyttäjä on antanut kaksi hakutermiä "Akseli Gallen-Kallela" ja "Napoleon I" ja järjestelmä on hakenut herrojen välisen yhteysketjun järjestelmän toimijaontologian kautta, jossa kuvataan mm. henkilöiden välisiä sosiaalisia suhteita. Tässä tapauksessa on datana hyödynnettiin yhdysvaltalaisen Getty-säätiön ULAN-rekisteristä (Union List of Artist Names) muodostettua ontologiaa.[11]

yhteyshaku

Kuva 3.9: Kulttuurisammon yhteyshaku hakee henkilöiden välisiä yhteyksiä.

Tiedon haussa on viimekädessä kysymys haun taustalla olevan tiedollisen tarpeen ja ongelman ratkaisemisesta. Hakukoneen pitäisi siksi yrittää arvata kyselyn perusteella mihin ongelmaan hakija on etsimässä ratkaisua ja suorittaa haku sen mukaisesti. Äärimmilleen vietynä tämä tarkoittaa sitä, että hakukone olisi älykäs *kysymys-vastaus-järjestelmä* (question answering system), jolle esitetään kysymys ja tuloksena saadaan vastaus. Esimerkiksi: "Kuka voitti keihäänheiton Helsingin olympialai-

kysymys-vastaus-järjestelmä

[10] Vuonna 2007 julkistettu Kulttuurisampo (http://www.kulttuurisampo.fi) yhdistää erilaisia heterogeenisiä suomalaisia kulttuurisisältöjä semanttisessa webissä [28, 37] ja oli FinnONTO-ohjelman laaja-alaisin sovellus.

[11] ULAN-rekisteri sisältää yksityiskohtaista tietoa n. 120 000 historiallisesta henkilöstä. Aineisto on nykyisin saatavilla linkitettynä avoimena datana ja on käytettävissä semanttisen webin standardien mukaisen SPARQL-datapalvelun kautta: http://www.getty.edu/research/tools/vocabularies/ulan/.

sissa?". Askel tähän suuntaan on esimerkiksi WolframAlpha-hakukone, jolta voi kysyä vaikka "Who was president of the US in 1856?" tai "How much does 1kg weight on the Moon?". Tällaisten järjestelmien kehittämisessä ei ole enää kysymys dokumenttien hausta vaan niiden sisältävän rakenteisen semanttisen informaation käsittelystä.

Rakenteista semanttista dataa hyödynnetään myös IBM:n Jepardy-peliä pelaavassa, parhaat ihmisasiantuntijat voittaneessa tekoälyjärjestelmässä Watson, jossa tehtävänä on löytää kysymyksiä annettuihin vastauksiin. Sen yhtenä tietolähteenä on Wikipedioista louhittu eri alojen tietoa sisältävä semanttinen verkko DBpedia.

Faktojen esittämisen ohella semanttista dataa tarvitaan myös järjestelmien personoinnissa käyttäjän mukaan ja käyttökontekstin esittämisessä, jolloin hakujärjestelmä saadaan toimimaan personoidusta käyttötilanteen, esimerkiksi paikan ja ajan mukaan.

Osa II

Linkitetty data

Luku 4

Linkitetyn datan esittäminen

Tässä luvussa esitellään ensin WWW:n perustana olevat ydinstandardit: HTML-kieli, webin keskeiset osoite- ja tunnistejärjestelmät sekä HTTP-protokolla. Linkitetyn datan ideana on kuvata webin eri osoitteita ja tunnisteita vastaavien tietojen välisiä suhteita semanttisena verkkona, joka sisältää metadataa webin tiedosta tietokoneen "ymmärtämällä" tavalla. Kun kone osaa tulkita webin sisältöjä, helpottuu tiedon yhdistäminen laajemmiksi kokonaisuuksiksi ja aiempaa älykkäämpien verkkopalveluiden kehittäminen.

Webin tiedon esittämisessä käytetään tässä luvussa esiteltävää semanttisen webin RDF-tietomallia. RDF-muotoista dataa voidaan havainnollistaa graafisesti verkkoina, tulkita tietokoneella kolmikkoina ja loogisina väittäminä sekä muuttaa data merkkijonoksi, jotta sen voi tallentaa tiedostoon.

4.1 Tiedon esitys semanttisena verkkona

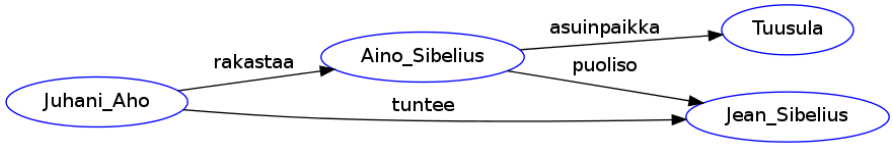
tietämyksen esittäminen
linkitetty data
semanttinen verkko

Semanttisen webin keskeisenä ideana on verkossa olevan *tietämyksen esittämin*en (knowledge representation) *linkitettynä datana* (linked data) sellaisessa muodossa, että tietokoneet voivat sitä tulkita. Tiedon esittämisen perustana on yksinkertainen tietomalli, *semanttinen verkko* (semantic net). Verkolla on eri käyttötarkoituksia varten useita esitys- ja tulkintatapoja, joita tarkastellaan seuraavassa.

4.1.1 Verkko kaaviona

solmu
kaari

Verkkoa kannattaa yleensä ajatella ja esittää havainnollisesti kuviona, jossa on nimettyjä *solmuja* (node) ja näitä yhdistäviä, suunnattuja nimettyjä *kaaria* (arc). Esimerkiksi kuvassa 4.1 on esitetty yksikertainen verkko, joka kuvaa Tuusulan järven alueelle eläneiden suomalaistaiteilijoiden välisiä suhteita.



Kuva 4.1: Yksinkertainen verkko

4.1.2 Verkon esittäminen kolmikkoina

Graafinen esitysmuoto sopii ihmisen tulkittavaksi, mutta tietokone ei pysty kuvioita käsittelemään. Konetta varten verkko voidaan kuitenkin esittää helposti muotoa

<alkusolmu, kaari, loppusolmu>

kolmikko

olevien *kolmikoiden* (triple) joukkona. Esimerkiksi kuvan 4.1 verkko voidaan määritellä seuraavalla joukolla:

```
{<Juhani_Aho, rakastaa, Aino_Sibelius>,
 <Juhani_Aho, tuntee, Jean_Sibelius>,
 <Aino_Sibelius, puoliso, Jean_Sibelius>,
 <Aino_Sibelius, asuinpaikka, Tuusula>}
```

subjekti
predikaatti
objekti

Semanttisessa webissä kolmikoiden jäseniä kutsutaan usein *subjektiksi* (subject), *predikaatiksi* (predicate) ja *objektiksi* (object) kieliteknologiasta lainatun terminologian mukaisesti:

<subjekti, predikaatti, objekti>

Tällaisten kolmikoiden voidaan myös tulkita liittävän solmuihin nimettyjen ominaisuuksien arvoja:

<solmu, ominaisuus, ominaisuuden arvo>

Luettelemalla verkon kaikkien solmujen ominaisuudet tällä tavalla voidaan mielivaltainen verkko esittää yksinkertaisesti joukkona kolmikoita. Kolmikkojoukkojen käsittely on erittäin yksinkertaista ohjelmallisesti, ja kolmikkomuotoisen tiedon käsittelyyn tarkoitetut ohjelmakirjastot ja työkalut kuten esimerkiksi Jena[1] perustuvat tähän. Kolmikoiden käsittelyyn perustuvat myös verkkojen tekstuaaliset esityskielet, joiden avulla verkot voidaan kuvata ja niitä voidaan tallettaa tiedostoon tekstinä.

Jos verkon eri julkaisijat käyttävät tiedon esittämisessä samoja *yhteentoimivia* (interoperable) rakenteita ja periaatteita, voidaan eri tahoilla julkaistua verkkotietoa yhdistellä ja rikastaa niitä toistensa avulla suuremmiksi kokonaisuuksiksi. Kahden verkon V_1 ja V_2 yhdistäminen voidaan tehdä erittäin yksinkertaisesti muodostamalla niiden kolmikoiden joukkojen unioni $V_1 \cup V_2$. Graafisesti tämä tarkoittaa sitä, että verkot vain asetetaan päällekkäin ja samoilla tunnisteilla merkityt solmut ja kaaret yhdistetään. Esimerkiksi tietokannoissa ja taulukkolaskennassa käytetyn taulukkomuotoisen tiedon yhdistäminen on huomattavasti hankalampaa.

4.1.3 Verkon looginen tulkinta

Kolmikot voidaan myös tulkita logiikan avulla *väittämiksi* (statement), jossa kaari kertoo alku- ja päätesolmun välisen yhteyden pitävän paikkansa. Esimerkiksi väittämä *Juhani Aho rakastaa Aino Sibeliusta*, voidaan kirjoittaa logiikan predikaattina muodossa:

väittämä

```
rakastaa(Juhani_Aho, Aino_Sibelius)
```

Näin voidaan semanttisille verkoille luoda täsmällisesti määritelty tulkinta. Logiikka mahdollistaa mm. uuden tiedon muodostamisen päättelyn avulla. Esimerkiksi siitä, että Aino Sibeliuksen puoliso on Jean Sibelius, voidaan sopivalla säännöllä päätellä, että Jean Sibeliuksen puoliso on vastaavasti Aino Sibelius.

4.1.4 Verkko tekstinä

Jotta tietokone voisi lukea muistiinsa verkon tiedostosta ja vastaavasti tallentaa verkon tiedostoon myöhempää käyttöä varten, pitää verkko

[1] Jena on Java-perustainen, paljon käytetty semanttisen webin ohjelmointiympäristö: https://jena.apache.org/

sarjallistaa voida esittää tekstimuodossa merkkijonona eli *sarjallistaa* (serialize).

Argumentti1	**Predikaatti**	**Argmentti2**
Juhani_Aho	rakastaa	Aino_Sibelius
Juhani_Aho	tuntee	Jean_Sibelius
Aino_Sibelius	puoliso	Jean_Sibelius
Aino_Sibelius	asuinpaikka	Tuusula

Taulukko 4.1: Kuvan 4.1 verkko taulukkomuodossa

Verkkojen sarjallistaminen tapahtuu semanttisessa webissä tätä varten kehitettyjen kielten avulla, joita ovat mm. XML-perustainen RDF/XML, N-Triples ja erityisesti Turtle, joihin palaamme tarkemmin seuraavassa luvussa. Esimerkiksi N-Triples-muodossa esimerkkimme verkko voitaisiin esittää yksinkertaisesti kirjoittamalla verkon solmujen tunnisteet kulmasulkujen sisään ja luettelemalla kolmikot pisteellä eroteltuina:

```
<Juhani_Aho> <rakastaa> <Aino_Sibelius> .
<Juhani_Aho> <tuntee> <Jean_Sibelius> .
<Aino_Sibelius> <puoliso> <Jean_Sibelius> .
<Aino_Sibelius> <asuinpaikka> <Tuusula> .
```

Verkko voidaan esittää luontevasti myös taulukkona, jonka rivit kuvaavat kolmikoita (taulukko 4.1). Taulukkolaskimessa tällainen tieto sarjal-
CSV-muoto listetaan *CSV-muodossa* (Comma Separated Values). Siinä jokainen laskimen taulukon rivi on esitetty niin, että sarakkeiden arvot on eroteltu toisistaan pilkuilla (comma)[2]. Esimerkiksi kuvan 4.1 ja taulukon 4.1 data voidaan sarjallistaa neljällä eri rivillä näin:

```
Juhani_Aho, rakastaa, Aino_Sibelius,
Juhani_Aho, tuntee, Jean_Sibelius,
Aino_Sibelius, puoliso, Jean_Sibelius,
Aino_Sibelius, asuinpaikka, Tuusula,
```

Semanttisen webin perustana olevan verkon rakenne on määritelty tarkasti W3C:n suosituksessa (standardissa) nimeltä *Resource Description*
RDF *Framework*[3], lyhyesti *RDF*. Olennaista on huomata, että semanttisessa webissä on kyse tiedon esittämisestä verkkoperustaisella RDF-

[2]Mikäli sarakkeiden arvoissa on pilkkuja, nämä voidaan huomioida omalla lainausmekanismilla. Sarakkeiden *erottimena* (delimiter) voidaan myös sopia käytettävän jotain muuta merkkiä kuin pilkkua.

[3]RDF-suositukset on julkaistu vapaasti verkossa osoitteessa https://www.w3.

tietomallilla, joka voidaan kuvata käyttämällä erilaisia *syntaktisia kieliä*. RDF-standardissa verkot kuvattiin alun perin tätä varten luodun XML-kielen avulla (RDF/XML). RDF:n käsitteellinen ero on kuitenkin iso XML-standardiin nähden: XML:ssä on kyse mekanismista erilaisten merkkauskielten määrittelemiseksi ja käsittelemiseksi, mutta RDF:ssä tietomallista, joka voidaan esittää erilaisilla syntaktisilla kielillä.

RDF-standardin ytimessä on sen nimessäkin esiintyvä *resurssin* (resource) käsite. Resurssilla tarkoitetaan mitä tahansa asiaa, josta voidaan esittää tietoa. Jotta resurssista voitaisiin kertoa jotain, on sillä oltava nimi eli *tunniste* (identifier), identiteetti. Käsitteellisten sekaannusten välttämiseksi tunnisteen on oltava yksikäsitteinen. Ihmisilläkään pelkkä nimi, esimerkiksi *Matti Virtanen*, ei riitä henkilön tunnistamiseen, vaan tarvitaan saman nimisetkin henkilöt yksilöivä henkilötunnus. Semanttisessa webissä tunnisteina käytetään tunnisteina verkko-osoitteita, joiden osoitetiedon kautta resursseita voidaan saada lisätietoa dataa linkitettäessä.

resurssi

tunniste

4.2 Semanttisen webin tekninen perusta

WWW perustuu suureen määrään erilaisia teknologioita ja standardeja. Sen ytimessä on kuitenkin erityisesti kolme standardia:

1. **HTML-kieli** web-sivujen esittämiseen,
2. **URI-osoitteet** sivujen löytämiseksi internetistä ja verkon resurssien tunnistamiseksi sekä
3. **HTTP-protokolla** sivujen ja tiedon välittämiseksi osoitteistaan.

4.2.1 HTML-kieli

HTML-kieli (Hypertext Markup Language) tarjoaa välineet web-sivujen ja niiden välisten linkkien esittämiseen, jotka sitten näytetään käyttäjälle

HTML

`org/RDF/`, jota kautta löytyy runsaasti muutakin aiheeseen liittyvää tietoa ja työkaluja.

selain
merkkauskieli
merkkaus

selaimen (browser) avulla. HTML on ns. *merkkauskieli* (mark-up language), jossa tietosisällön sekaan on upotettu tiedon rakennetta ja ilmiasua kuvaavia sisäkkäisiä *merkkauksia* (tag). Merkkaus alkaa kulmasulkuihin kirjoitetulla alkumerkkauksella <*merkkaus*> ja päättyy loppumerkkaukseen </*merkkaus*>. Esimerkiksi ykköstason otsikko ja sitä seuraava tekstikappale merkitään `h1`- ja `p`-merkkauksella näin:

```
<h1> Johdanto </h1>
<p> Jo muinaiset kreikkalaiset tutkivat logiikkaa. </p>
```

Tämän perusteella selain osaa näyttää otsikon isoilla kirjasimilla (font) kirjoitettuna ja sitten kappaletekstin pienemmällä. Olennaista on, että ilmiasun muodosta päätetään vasta päätelaitteessa: esimerkiksi kännykässä otsikkojen pitää olla pienempiä kuin isommalla näytöllä varustetussa kannettavassa tietokoneessa.

SGML

HTML on alkujaan sovellus *SGML*-metakielestä (Standard Generalized Markup Language), jonka avulla voidaan määritellä merkkauskieliä. SGML standardoitiin jo ennen webin tuloa vuonna 1986 kansainvälisen ISO-järjestön toimesta[4]. Nykyisin HTML-kielestä on käytössä alkuperäistä monipuolisempi versio HTML5[5]. Se perustuu W3C-järjestön vuonna 1998 standardoimaan XML-metakieleen ja standardiperheeseen. XML on yksinkertaistettu versio SGML-metakielestä.

HTML5

4.2.2 HTTP-protokolla

Web-sivu tai muu tieto luetaan selaimeen sen URL-osoitteesta käyttämällä yksinkertaista HTTP-protokollaa. Sen keskeinen innovaatio on maailmanlaajuinen nimipalvelinjärjestelmä, joka takaa sen, että tiedon julkaisijoiden sivujen osoitteet ovat yksikäsitteisiä eivätkä voi mennä sekaisin. Tämä tapahtuu rekisteröimällä *aluenimiä* (domain name). Esimerkiksi Aalto-yliopisto on rekisteröinyt käyttöönsä aluenimen `aalto.fi`. Siksi kaikki HTTP-URL osoitteet, jotka alkavat tuolla nimellä kuten

`http://aalto.fi/fi/research`

[4] ISO 8879:1986
[5] https://www.w3.org/TR/html5/

ohjautuvat yliopiston omalle palvelimelle, joka päättää osoitteen loppuosan avulla, mikä sivu kysyjälle palautetaan. Aluenimen omistaja voi myös ottaa vapaasti käyttöönsä *alialueita* (subdomain), jotka kirjoitetaan nimen eteen pisteellä erotettuna. Esimerkiksi `www.aalto.fi` viittaa yliopiston web-sivustoon ja `data.aalto.fi` yliopiston avoimen linkitetyn datan palveluun.

alialue

Nimen haltijalla on mahdollista ottaa käyttöön *edustapalvelin* (reverse proxy), joka tarvittaessa ohjaa alueelle tulevan HTTP-kutsun uudelleen (redirect) toiseen osoitteeseen. Ohjauksessa voidaan ottaa huomioon mm. käyttäjän selain ja käyttötilanne, kuten aika ja paikka. Esimerkiksi osoite `http://aalto.fi` ohjautuu tätä kirjoitettaessa omassa selaimessani osoitteeseen `http://www.aalto.fi/fi/`, eli palvelun tarjoaja olettaa `http://aalto.fi`-osoitteen käyttäjän haluavan lukea tässä tapauksessa yliopiston suomenkielisen kotisivun, joka löytyy uudelleenohjatusta osoitteesta. Uudelleenohjausta voidaan tarvittaessa muuttaa myöhemmin edustapalvelinta konfiguroimalla. Toinen paljon käytetty palvelintyyppi on yleensä asiakaspäässä oleva *välipalvelin* (proxy server), jonka avulla voidaan mm. tallettaa usein kysyttyjä sivuja paikallisesti ja näin nopeuttaa tiedonhakua. *Turvapalvelin* (security sever) taas on verkon tietoturvaa hoitava erillinen palvelin.

edustapalvelin
uudelleenohjaus
välipalvelin
turvapalvelin

4.2.3 Yhtenäiset osoitteet ja tunnisteet: URI

Webiin sisältyy systemaattinen tapa esittää HTML-sivujen sijainti internetissä *URL-osoitteina* (Uniform Resource Locator). Kun osoite kirjoitetaan esimerkiksi selaimeen tai painetaan linkkiä (joka vastaa URL-osoitetta), järjestelmä käy lukemassa osoitteessa olevan HTML-sivun ja *esittää* (render) sen linkkeineen käyttäjälle.

URL

URL on erikoistapaus yleisemmästä *URI-tunnisteesta* (Uniform Resource Identifier). URI-tunnisteet, kuten URL-osoitteet, määritellään ns. *URI-skeemojen*[6] (URI schema) avulla. Näitä on otettu käyttöön kymmeniä erilaisia, kuten URN-tunnisteet, OID-tunnisteet, GIT-tunnisteet, BITCOIN-tunnisteet jne. Käytetyin tunnistemuoto on kuitenkin web-osoitteina toimivat HTTP-tunnisteet (URL-tunnisteet) (ja tietoturvallisemmat HTTPS-tunnisteet). Semanttisessa webissä pyritään käyttämään erityisesti niitä tunnisteina. Tähän on kaksi painavaa syytä:

URI-tunniste
URI-skeema

[6] `https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml`

1. HTTP-osoitteiden domain-nimiin perustuva kansainvälinen webin nimipalvelujärjestelmä takaa automaattisesti sen, ettei kukaan muu kuin alueen, esimerkiksi `dbpedia.org`, rekisteröinyt käyttäjä voi ottaa käyttöön toista samanlaista tunnistetta, ts. tunnisteet ovat automaattisesti yksilöiviä kaikkialla maailmassa. Tunnisteiden *luominen* (minting) voidaan näin hajauttaa helposti niitä tarvitseviin organisaatioihin. Pelkän nimen kuten *Väinö Linna* tai *Pyhäjärvi* käyttäminen tunnisteena ei riitä, koska olemassa voi olla useita samannimisiä henkilöitä, paikkoja ja muita asioita.

tunnisteiden luominen

2. HTTP-tunnisteeseen sisältyy paitsi itse tunniste myös verkko-osoite, johon on mahdollista tallentaa lisätietoa tunnisteesta. Tiedon saa siten näkyville ja käyttöön pelkän tunnisteen avulla. Koska tietoalkioon voidaan viitata sen URI-tunnisteella, voivat eri toimijat verkossa liittää samaan alkioon (esimerkiksi yritykseen) ominaisuuksia itsenäisesti toisistaan riippumatta ilman sekaannuksia. Jos esimerkiksi eri kirjastojen tietokannoissa viitataan Väinö Linnaan samalla URI-tunnisteella, eivät eri Väinö Linnat ja kirjat mene sekaisin aineistoja yhdistettäessä.

Alla on esitetty yhteenvetona URI-skeemojen yleinen rakenne kaavana:

skeema:*hierarkiaosa*[?*kysely*][#*fragmentti*]

Kaavassa ':'-merkki erottaa skeeman nimen hierarkiaosasta ja kulmasulut '[' ja ']' rajaavat valinnaisia osia *kysely* ja *fragmentti*, jotka tunnistaa alkumerkeistä '?' ja '#'. Skeemoja ovat mm. `http`, `urn`, `mailto` jne. Hierarkiaosa koostuu tyypillisesti aluenimestä (domain), esimerkiksi `aalto.fi` tai `www.helsinki.fi`, ja tätä seuraavasta kauttaviivalla '/' alkavasta hakemistopolkusta palvelimella, kuten esimerkiksi `http://aalto.fi/fi/research/`. Aluenimen jälkeen voidaan kaksoispisteellä erottaen vielä määritellä numeerinen portti[7], johon HTTP-kutsu lähetetään. Esimerkiksi:

```
http://aalto.fi:80/fi/research/
```

HTTP-URI-tunnisteiden yksi merkittävä etu on, että niiden valinnaisessa kyselyosassa voidaan välittää palvelimelle tietoa osana HTTP-kutsua *nimi*=*arvo*-pareina, jotka on erotettu toistaan '&'-merkillä. Tyypillinen

[7] Oletusporttina HTTP-protokollassa on 80 ja HTTPS-protokollassa 443.

käyttötapaus on hakukyselyiden parametrien välittäminen selaimelta palvelimelle. Jos esimerkiksi Google-hakukoneessa kirjoittaa hakukenttään "Helsinki tapahtuma", lähtee palvelimelle alla oleva HTTP-kutsu:

https://www.google.fi/search?sourceid=navclient&hl=fi&ie=UTF-8&rlz=1T4GUEA_enFI654FI654&q=helsinki+tapahtumat

Tunnisteisiin ja niiden käsittelyyn voi sisältyä haun ohella älykästä päättelyä. Esimerkiksi Suomen lainsäädäntöä ja oikeustapauksia julkaisevassa Semanttinen Finlex -palvelussa[8] käytetään EU:n piirissä kehitettyjä ELI-[9] ja ECLI-tunnisteita[10]. Esimerkiksi ELI-tunniste

http://data.finlex.fi/eli/sd/2008/521/ajantasa/20160101

palauttaa vuoden 2008 säädöksen numero 521 (Vakuutusyhtiölaki) ajantasaisen version ajanhetkellä 1.1.2016. Käyttäjän ei tarvitse tietää säädösten tarkkoja voimaantuloaikoja voidakseen viitata niihin, ja ELI-mekanismi mahdollistaa samasta säädöksestä eri aikoina voimassa oleviin versioihin viittaamisen vastaavalla tavalla. Myös lain sisällä voidaan viitata mihin tahansa lainkohtaan ja sen tietyllä hetkellä voimassa olleeseen versioon. Siten esimerkiksi ELI-osoite

http://data.finlex.fi/eli/sd/2008/521/luku/1/pykala/1/ajantasa/20160101

palauttaa Vakuutusyhtiölain 1 luvun 1 pykälän ajanhetkellä 1.1.2016.

URI-kaavan lopussa on vielä valinnainen *fragmenttiosa* (fragment), jolla voidaan viitata HTML-sivun nimettyyn *ankkuriin* (anchor)[11], jolloin selain osaa avata sivun juuri oikeasta kohdasta ankkurista alkaen. Kyselyosasta poiketen fragmenttiosaa ei lähetetä palvelimelle HTTP(S)-kutsun mukana, vaan se on tarkoitettu selaimen sisäiseen käyttöön.

fragmentti
ankkuri

URI-tunnisteiden parametreina välitetään usein tekstidataa palvelimelle, esimerkiksi hakutermejä hakukoneelle tai toinen URI-tunniste uudel-

[8] Palvelu on käytettävissä osoitteessa http://data.finlex.fi ja sen ominaisuuksista ja toteutuksesta löytyy lisätietoa sivulta http://seco.cs.aalto.fi/projects/lawlod.

[9] European Legislation Identifier, http://en.wikipedia.org/wiki/European_Legislation_Identifier

[10] European Case Law Identifier, http://en.wikipedia.org/wiki/European_Case_Law_Identifier

[11] Ankkuri on HTML-kielen merkkaustapa, jonka avulla HTTP-kutsuissa voidaan viitata WWW-sivun sisällä oleviin eri kohtiin.

leenohjausta varten.Tällaisessa datassa ei välttämättä voi käyttää URI-tunnisteissa muuhun käyttöön varattuja tai muita erikoismerkkejä, kuten välilyönti, '&', kaksoispiste, kauttaviiva yms. Muuten URI-tunnisteen yksikäsitteinen tulkinta palvelimella ei välttämättä ole mahdollista. Jotta erikoismerkkien käyttö olisi mahdollista, voidaan parametrien erikoismerkit koodata *URL-koodauksen* (URL encode) avulla, jossa ne muunnetaan vain sallittuja merkkejä sisältäviksi %-alkuisiksi koodeiksi. Esimerkiksi osoite

URL-koodaus

```
http://www.koe.fi/erikoisia merkkejä (osoite)
```

koodautuu muotoon

```
http%3A%2F%2Fwww.koe.fi%2Ferikoisia%20merkkej%C3%A4%20(osoite)
```

muunnoksilla ':' = %3A, '/' = %2F, ' ' = %20 ja 'ä' = %C3%A4.

koodin avaus

Koodausta ja *koodin avausta* (decode) molempiin suuntiin voi tehdä näppärästi ohjelmointikieliin sisältyvillä funktioilla ja interaktiivisesti verkossa olevien muunnospalveluiden[12] sivuilla.

Kukin URI-skeema määrittelee omanlaisensa tunnisteen kirjoitusasun (rakenteen). URL-muotoisen tunnisteen olennainen piirre on, että tunniste sisältää osoitetiedon, jonka perusteella voidaan lukea tunnisteeseen liittyvää tietoa verkosta. Tämä ei ole mahdollista kaikissa URI-skeemoissa kuten *URN-skeemassa* (Universal Resource Name), jonka mukaiset tunnisteet sisältävät vain yksilöivää tunnistetietoa samaan tapaa kuin henkilötunnukset. Esimerkiksi julkaisujen ISBN-numeroille on Suomessa käytössä mm. seuraava URN-tunniste

URN-skeema

```
urn:isbn:978-952-10-4171-6
```

ja alla on esimerkki URN-mallin käytöstä EU-direktiivien tunnisteena:

```
urn:lex:eu:council:directive:2010-03-09;2010-19-UE
```

URN-tunnisteen etuna on sen riippumattomuus aluenimistä ja palvelimista (domain name), jotka voivat vaihtua ajan kuluessa organisaatioiden muuttuessa. URN-tunnistetta ei kuitenkaan voi URL-tunnisteen tapaan kirjoittaa sellaisenaan selaimeen lisätiedon toivossa, koska se ei sisällä osoitetietoa. URN-tunnisteen *tulkitsemiseksi* (resolve) tarvittaisiin erillinen maailmanlaajuinen URN-tulkintajärjestelmä, joka olisi paljolti

[12] Esimerkiksi http://meyerweb.com/eric/tools/dencoder/

päällekkäinen jo olemassa olevan HTTP-osoitteiden tulkintajärjestelmän kanssa, eikä sellaista ole siksi luotu.

Yksi kompromissi on käyttää URN-tunnisteita osana URL-osoitteita. Esimerkiksi Kansalliskirjaston urn.fi-palvelun kautta voi löytää lisätietoa tunnistetta `urn:isbn:978-952-10-4171-6` vastaavasta julkaisusta näin:

`http://urn.fi/urn:isbn:978-952-10-4171-6`

URN-tunnisteiden toinen haaste on, että niiden käyttö edellyttää myös tunnisteiden keskitettyä jakamispalvelua, jotta kaksi eri tahoa eivät vahingossa ottaisi täyttöön samaa tunnistetta eri asioille, kuten samaa ISBN-numeroa kahdelle eri julkaisulle. Kirjastomaailmassa on jo aiemmin luotu globaali ISBN-tunnisteiden jakamisjärjestelmä eri valtioiden kesken, mutta useimmilla muilla sovellusalueilla tällaista globaalia infrastruktuuria ei ole olemassa. HTTP-URL-tunnisteita käytettäessä tällaista tunnisteiden globaalia jakamis- ja hallintaongelmaa ei ole, vaan eri organisaatiot voivat turvallisin mielin luoda hajautetusti ja itsenäisesti tunnisteita hallinnassaan olevilla nimialueilla (esimerkiksi `helsinki.fi` tai `nokia.com`) ilman pelkoa tunnisteiden sekoittumisesta.

Tällöin kuitenkin samalle asialle, vaikkapa Sibeliuksen viidennelle sinfonialle tai Espoon Otaniemelle, voidaan luoda useita eri tunnuksia eri tahojen toimesta paikallisesti, mikä aiheuttaa omia haasteita tunnisteiden samaistamisessa, jos eri toimijoiden tietoja myöhemmin yhdistetään. Esimerkiksi samalle kirjailijalle on yleensä käytössä erilaisia tunnisteita eri valtioiden kirjastojen, arkistojen ja museoiden rekistereissä (ns. auktoriteettitietokannoissa) tai verkkoaineistoissa kuten DBpediassa.

URI-tunnisteitakin laajempi tunnistejoukko ovat *IRI-tunnisteet* (Internationalized Resource Identifier). Tässä *Internationalized* viittaa siihen, että tunnisteissa voidaan käyttää laajaa unicode-merkistöä, kuten ääkkösiä, arabian kielen merkkejä tai japanin kata- ja hiragana-tavukirjoitusta[13]. Perinteisissä URI-tunnisteissa tyydytään käyttämään vain yksinkertaista ASCII-merkistöä, josta puuttuvat mm. ääkköset[14] Haasteena erikoismerkkejä sisältävien IRI-tunnisteiden käytölle on eri maissa käytettävät erilaiset näppäimistöt. Esimerkiksi amerikkalaisissa näppäimistöissä ei ääkkösiä näy, jolloin niitä sisältävän HTTP-osoitteen

IRI-tunniste

[13]Unicode-merkistöön kuuluu nykyisin yli 128 000 merkkiä 135 eri kirjoitusjärjestelmästä.

[14]ASCII-merkistöön kuuluu 128 (tai laajennettuna 256) merkkiä.

syöttäminen vaikkapa selaimeen vaatii erityistoimenpiteitä. Samoin arabian kielisen WWW-osoitteen kirjoittaminen selaimeen suomalaisella näppäimistöllä on haasteellista.

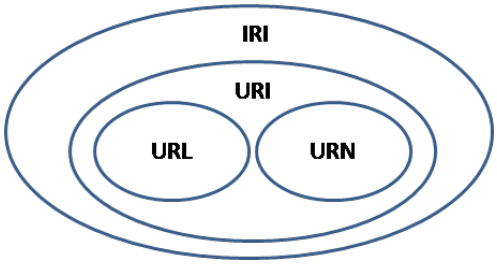
Erikoismerkkejä sisältävät IRI-tunnisteet voidaan koodata yksikäsitteisesti URI-tunnisteissa käytettävissä oleviksi ASCII-merkeiksi ja päinvastoin, joten tässä mielessä on sama käytetäänkö IRI- vai URI-tunnisteita. Käytännössä erikoismerkistön välttäminen URI-tunnisteissa kuitenkin helpottaa paitsi tunnisteiden kirjoittamista myös ohjelmointia. Tunnisteet on tarkoitettu koneille eivätkä ne yleensä näy sovellusten ihmiskäyttäjille. Laajaa eri kielet kattavaa Unicode-merkistöä käytetään loppukäyttäjille näkyvissä tunnisteiden nimikkeissä (label).

Käytännössä on yleensä yksinkertaisinta pitäytyä käyttämään URI-tunnisteita ilman ASCII-järjestelmää ilman erikoismerkkejä. Tämän vuoksi puhumme tässä teoksessa URI-tunnisteista. IRI-tunnisteisiin viitataan vain silloin, jos on tarvetta korostaa erikoismerkkien käyttöä.

pysyvä tunniste

PID

Tärkeimpien webin tunnistetyyppien keskinäistä kattavuutta on havainnollistettu kuvassa 4.2 joukko-opillisella Venn-diagrammilla. Tunnisteet pyritään muodostamaan niin, että ne olisivat ajan saatossa *pysyvä tunnisteita* (persistent indentifier, PID). Eri aloilla on käytössä on monenlaisia tunnuksia (esim. henkilötunnukset), joita voidaan käyttää hyväksi WWW-perustaisten tunnisteiden luomisessa.



Kuva 4.2: Webin eri tunnistetyyppejä joukkoina ja niiden sisältyvyys toisiinsa

4.2.4 Resurssien ja literaalien esittäminen

Resurssi (resource) on semanttisessa RDF-verkossa oleva solmu, joka voidaan esittää graafisesti soikioina. Resurssin ominaisuudet ilmaistaan solmusta lähtevien kaarien avulla. Tarkastellaan esimerkkinä Wikipediasta louhittua RDF-verkkoa DBpedia, joka on julkaistu ja saatavilla verkossa avoimena linkitettynä datana[15]. Esimerkiksi elokuva *Tuntematon sotilas* (alkuperäinen vuoden 1955 versio) on esitetty DBpediassa resurssilla, jonka tunniste on:

```
http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)
```

Tuntemattoman sotilaan resurssista, ts. verkon solmusta, lähtee runsaasti kaaria eli resurssin *ominaisuuksia* (property), jotka kuvaavat metatietoa elokuvasta, kuten että elokuvan nimi on *The Unknown Soldier (1955 film)*, sen pituus on 10 140 sekuntia ja että elokuvan ohjasi *Edvin Laine* ja tuotti *Toivo Särkkä*. Ominaisuuksien arvot ovat verkon solmuja ja ne voivat olla joko

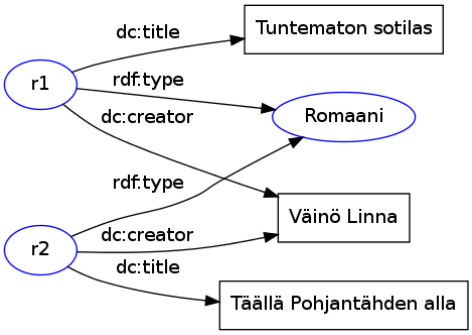
ominaisuus

1. *literaalista* (literal) dataa, kuten resurssin ihmisluettava nimi tai siihen liittyvä numeerinen tieto, tai *literaali*
2. toisia resursseja, joilla voi olla omia ominaisuuksiaan.

Esimerkiksi *Tuntematon sotilas* -elokuvan nimi samoin kuin juonen lyhyt kuvaus esitetään merkkijonoliteraalina. *Edvin Laine* ja *Toivo Särkkä* taas ovat toisia resursseja, jotka on kuvattu omien ominaisuuksiensa avulla DBpedian semanttisessa verkossa.

Literaalisolmut esitetään usein graafisesti suorakaiteen muotoisina solmuina, jotta ne erottuisivat selkeästi soikeista resurssisolmuista. Esimerkiksi kuvassa 4.3 on esitetty tietoa romaaneista *Tuntematon sotilas* ja *Täällä Pohjantähden alla*. Mukana on pyöreiden resurssisolmujen ohella myös literaaleja, jotka on esitetty suorakaiteina. Kuvan solmu *r1* edustaa Tuntematonta sotilas -romaania ja sen ominaisuudet kertovat, että kyse on romaanista (`rdf:type`), että teoksen kirjoitti (`dc:creator`) Väinö Linna ja että teoksen nimi (`dc:title`) on “Tuntematon sotilas”. Tässä esimerkiksi romaani-käsitteeseen *Romaani* ei liity muuta tietoa kuin

[15] http://dbpedia.org



Kuva 4.3: Semanttinen verkko.

sen tunniste. Jos sen sijaan olisi käytetty viittausta KOKO-ontologiassa olevaan vastaavaan käsitteeseen *romaanit*

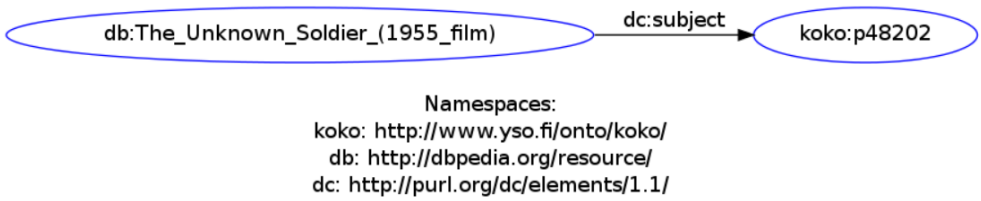
http://www.yso.fi/onto/koko/p35819

olisi kuvaus semanttisesti huomattavasti rikkaampi.

Literaalilla tarkoitetaan merkkijonoa, numeroa, päiväystä tai muuta perustietoa, jota käytetään ominaisuuden arvona. Esimerkiksi:

```
"Helsinki"
3.14
2014-12-13
```

Literaali on itsessään vain arvo, eikä sillä voi olla omia ominaisuuksia. Graafisesti tämä tarkoittaa sitä, että literaali voi olla RDF-verkossa vain kaaren osoittamassa päässä ja että siitä itsestään ei voi lähteä kaaria. Esimerkin tapauksessa *Väinö Linna* on esitetty literaalina eli merkkijonona, jolloin sillä ei voi olla ominaisuuksia, kuten DBpedian tapauksessa. Tällä mallinnusratkaisulla ei ole esimerkiksi mahdollista kertoa, että Väinö Linna syntyi vuonna 1920 tai muuta tietoa hänestä. Se tieto pitäisi liittää Väinö Linnan identifioivaan resurssisolmuun. Literaalidatan kautta data ei linkity resursseina toisiinsa, vaikka esimerkiksi haussa voidaan toki hakea kaikki resurssit, joiden nimi on sama merkkijono “Väinö Linna”. Tiedon haun kannalta graafi kertoo, että molempien romaanien kirjoittaja on *Väinö Linna*, mutta tämä ei sulje pois esimerkiksi sitä mahdollisuutta, että olisi olemassa kaksi Väinö Linnaa, joista toinen kirjoitti teoksen *Tuntemattoman sotilas* ja toinen *Täällä Pohjantähden alla*.



Kuva 4.4: Resurssi, jolla on ominaisuuden arvona toinen resurssi toisessa tietovarannossa.

4.2.5 Datan linkitys tietojoukkojen välillä

Resursseihin voidaan liittää eli linkittää ominaisuuksien avulla tietoa paitsi datajoukon sisällä myös viittauksilla resursseihin, jotka on kuvattu toisissa tietovarannoissa. Esimerkiksi kuvassa 4.4 on kerrottu, että englannin kielisen Wikipedian Tuntematon sotilas -elokuvan (yhtenä) aiheena on *sota*, sillä ominaisuusresurssi (kaari)

`http://purl.org/dc/elements/1.1/subject`

viittaa Dublin Core -metatietostandardissa määriteltyyn ominaisuuteen *subject*, jonka arvolla kuvataan verkossa dokumentin sisältöä. Ominaisuuden arvona oleva resurssi

`http://www.yso.fi/onto/koko/p48202`

taas on kotimaisessa KOKO-ontologiassa määritelty käsite *sota (toiminta)*. Sulmujen tunnisteiden alku-osat `db:`, `dc:` ja `koko:` viittavat kuvan alaosassa ilmaistuihin *nimiavaruuksiin* (namespace) eli web-osoitteisiin, joissa resurssit on tarkemmin määritelty. Palaamme nimiavaruuksiin tarkemmin tuonnempana.

Sota-käsitteen kuvaus RDF-muodossa löytyy tunnisteen mukaisesta osoitteessa ja kertoo mm. käsitteen nimikkeet suomeksi (*sota (toiminta)*), ruotsiksi (*krigföring*) ja englanniksi (*war*), ja että sota on käsitteen *keskinäinen toiminta* alakäsite ja että sillä on suppeampia käsitteitä, kuten *asemasota*, *ilmasota* jne. Näin Tuntematon sotilas -elokuva on mahdollista yhdistää vaikkapa asema- tai ilmasodasta kertovaan kirjallisuuteen.

Kuva 4.4 on esimerkki siitä, miten RDF mahdollistaa resurssiin liittyvän tiedon rikastamisen toisten RDF-verkkojen uusilla ominaisuuksilla,

kunhan vain resurssien tunnisteet ovat tiedossa ja yhdistetään toisiinsa. Tällaisilla datajoukkojen välisillä linkityksillä voidaan esimerkin mukaan yhdistää englanninkielisen DBpedian aineistoja kotimaiseen ontologiainfrastruktuuriin, jolloin datoja voidaan rikastaa semanttisesti toistensa avulla.

nimike

Olennaista on huomata tunnisteiden ja niiden ihmisluettavien literaalisten *nimikkeiden* (label) välinen ero RDF-verkoissa: tunnisteet kuten `http://www.yso.fi/onto/koko/p48202` viittaavat kielistä riippumattomiin käsitteisiin eivätkä kielellisiin ilmauksiin. Näin käy mahdolliseksi esittää tietoa tietokoneelle kielestä riippumattomalla tavalla, eikä eri kieliä varten tarvitse kehittää omia verkkoja tai indeksejä tiedonhakua varten. Nimikkeiden avulla puolestaan voidaan rakentaa silta koneluettavien käsitteiden ja ihmisluettavien ilmausten välillä eri kielille. RDF-tietomalli soveltuu tästä erottelusta johtuen erittäin hyvin monikielisten järjestelmien kehittämiseen.

Myös ominaisuudet (verkon kaaret) ovat RDF-tietomallissa aina resursseja, joten myös niistä voidaan esittää lisää tietoa tunnistetta vastaavassa paikassa verkossa. Esimerkiksi kuvassa 4.4 ominaisuuden `dc:subject` koneluettava määrittely löytyy Dublin Core -standardin sivuilta. Tällä on merkitystä mm. päättelyssä, jossa erilaisten kaarien avulla voidaan tehdä erilaisia johtopäätöksiä. Palaamme päättelyyn tarkemmin ontologioiden esittelyn yhteydessä.

Luku 5

RDF-verkon esittämiskielet

Semanttisen RDF-verkon esittämistä ja ohjelmointia varten verkko pitää voida esittää tekstuaalisesti jonkin jokin kielioppisäännöstön eli *syntaksin* (syntax) eli *notaation* (notaation) avulla merkkijonona. Vastaavasti tietokoneen keskusmuistissa oleva verkko pitää voida muuttaa tekstuaaliseen asuun eli *sarjallistaa* (serialize), jotta verkko voidaan tallettaa massamuistiin tiedostoon tiedonsiirtoa ja pidempiaikaista säilytystä varten. Tässä luvussa esittelemme RDF-verkkojen tärkeimmät kuvauskielet.

syntaksi

notaatio

sarjallistaa

5.1 Data kolmikkoina: N-Triples

Suoraviivaisimman tavan RDF-verkon kuvaamiseen tarjoaa *N-Triples -notaatio*[1], jossa verkon kolmikot esitetään yksinkertaisesti peräkkäin pisteillä toisistaan erotettuina kolmikkoina. Kolmikoiden esitysjärjestyksellä ei ole merkitystä.

N-Triples-notaatio

RDF-verkossa merkkijonotyyppisellä literaalilla ei voi olla ominaisuuksia, mutta sillä voi kuitenkin olla kielen ilmaiseva valinnainen määre muodossa:

literaali`@`*kielitunnus*

Tässä *kielitunnus*[2] on IETF Trust ja ISO-järjestöjen standardien mukainen lyhenne, kuten `fi` (suomi), `sv` (ruotsi) tai `en` (englanti).

kielitunnus

[1] https://www.w3.org/TR/n-triples/
[2] http://www.w3.org/International/articles/language-tags/

Alla on esimerkkinä kaksi kolmikkoa englannin kielisen DBpedian RDF-verkosta, jotka kertovat *Tuntematon sotilas* -elokuvan ohjaajan ja nimen, sekä kolmikko, jolla kuvausta on rikastettu kertomalla filmin aihe suomalaisen KOKO-ontologian mukaan. Tässä notaatiossa resurssit on merkitty URI-osoitteilla kulmasulkujen sisään; merkkijonomuotoinen data taas on merkitty lainausmerkkien väliin. Lainauksen lopussa on merkkijonon kielen ilmaiseva tunnus, tässä englantia tarkoittava `@en`. Risuaidalla '#' alkavat tekstit saman rivin loppuun saakka ovat vain ihmislukijalle tarkoitettuja *kommentteja* (comment), jotka kone ohittaa merkityksemättöminä.

kommentti

```
# Tuntemattoman sotilaan ohjasi Edvin Laine
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
<http://dbpedia.org/ontology/director> # Ohjaaja-ominaisuus
<http://dbpedia.org/resource/Edvin_Laine> . # Edvin Laineen resurssi

# Filmin nimi englanniksi
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
<http://www.w3.org/2000/01/rdf-schema#label> # Resurssin nimike
"The Unknown Soldier (1955 film)"@en . # Literaali arvo

# Tuntemattoman sotilaan aiheena on sota
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
<http://purl.org/dc/elements/1.1/subject> #Aiheen kertova ominaisuus
<http://www.yso.fi/onto/koko/p48202> . # Käsite "sota" KOKO:ssa
```

N-Triples on yksinkertaisin tapa RDF-verkon esittämiseksi ja sen lukeminen tiedostosta on koneelle helppoa ja nopeaa yksinkertaisesta syntaksista johtuen. Kolmikoiden kirjoittaminen käsin tai niiden muodostaman kokonaisuuden hahmottaminen ei kuitenkaan ole välttämättä helppoa ihmislukijalle. Esimerkiksi yllä pitkät URI-tunnisteet haittaavat luettavuutta ja subjekti-URI joudutaan vielä toistamaan jokaisen kolmikon yhteydessä.

Pitkien URI-tunnisteiden lyhentämistä varten URI voidaan jakaa kahteen osaan:

1. Osoitteen loppuosaa viimeisen '/'- tai '#'-merkin jälkeen kutsutaan *palkallisnimeksi* (local name).

paikallisnimi

2. Osoitteen alkuosan hierarkkista hakemistopolkua paikallisnimen saakka kutsutaan *nimiavaruudeksi* (namespace) .

nimiavaruus

Esimerkiksi tunnisteessa

http://www.yso.fi/onto/koko/p48202s

paikallisnimi on p48202 ja nimiavaruus http://www.yso.fi/onto/koko/. Liittämällä nämä yhteen syntyy URI.

Nimiavaruudelle voidaan antaa käyttäjän vapaasti valittavissa oleva lyhyt nimi RDF-graafin yhteydessä *etuliitteenä* (prefix), joka voidaan määritellä muodolla: *etuliite*

`@prefix` *etuliite*: <*nimiavaruus*> .

Tämän jälkeen URIt voidaan ilmaista lyhennysmerkinnällä:

etuliite:*paikallisnimi*

Jos esimerkiksi edellä olevan *Tuntemattoman sotilaan* datan nimiavaruudet on määritelty tiedoston alussa näin

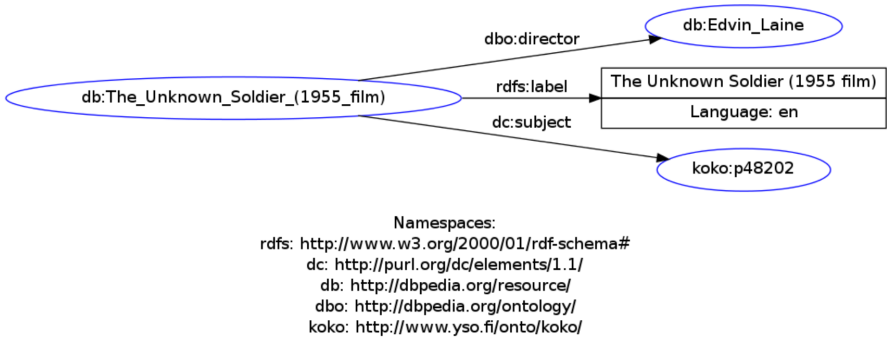
```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix db: <http://dbpedia.org/resource/> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix koko: <http://www.yso.fi/onto/koko/> .
```

voidaan kolmikot ilmasta paljon selkeämmällä tavalla:

```
# Tuntemattoman sotilaan ohjasi Edvin Laine
<db:The_Unknown_Soldier_(1955_film)>
<dbo:director> # Ohjaaja-ominaisuus DBpedian ontologiassa
<db:Edvin_Laine> . # Edvin Laineen resurssi

# Filmin nimi englanniksi
<db:The_Unknown_Soldier_(1955_film)>
<rdfs:label> # Ominaisuus label kertoo nimikkeen
"The Unknown Soldier (1955 film)"@en . # Literaali arvo

# Tuntemattoman sotilaan aiheena on sota
<db:The_Unknown_Soldier_(1955_film)>
<dc:subject> #Aiheen kertova ominaisuus
<koko:p48202> . # Käsite "sota" KOKO-ontologiassa
```

Nämä kolmikot nimiavaruuksien avulla on esitetty graafina kuvassa 5.1, joka on tuotettu Linked Data Finland -palveluun asennetulla RDF Grapher -*visualisaattorilla*[3] (visualizer). Se tarjoaa lomakkeen, johon voi liittää RDF-kuvauksen ja nappia painamalla kone palauttaa vastaavan kuvan RDF-verkosta. Lomakkeella voi valita N-Triples-notaation ohella käytettäväksi muitakin RDF:n esityskieliä (Turtle, RDF/XML, *visualisaattori*

[3] http://www.ldf.fi/service/rdf-grapher/



Kuva 5.1: Kolmikkoja nimiavaruuksien avulla esitettynä

RDF/JSON), joihin palaamme tuonnempana. Myös tuotettavan kuvan formaatin voi valita (PNG, JPG, SVG, PDF, EPS jne.). Visualisaattorin avulla voi samalla *validoida* (validate) RDF-muotoista dataa, ts. tarkistaa, että se on syntaktisesti oikein muodostettua. Myös mm. W3C tarjoaa verkossa RDF-datan visualisointi- ja validointipalvelun[4].

validointi

Nimiavaruuksien avulla webin eri julkaisijat voivat ottaa vapaasti käyttöön (paikallis)nimiä resurssien tunnisteita varten (esimerkiksi `subject`) ilman pelkoa tunnisteiden sekoittumisesta, ts. että joku toinen olisi jo ottanut tunnisteen käyttöön eri merkityksessä. Esimerkiksi ranskalaisen paikkatieto-organisaation luoma tunnus `france:Paris` voisi viitata Ranskan pääkaupunkiin, `texas:Paris` Teksasissa olevaan saman nimiseen kaupunkiin yhdysvaltalaisessa paikkatietorekisterissä ja `julkkikset:Paris` perijätär Paris Hiltoniin.

Nimiavaruuden nimi voi olla myös tyhjä, jolloin edes nimiavaruutta ei tarvitse merkitä. Esimerkiksi määrittelyllä

```
@prefix : <http://dbpedia.org/resource/> .
```

Tuntemattomaan sotilaaseen voi viitata muodolla:

```
:The_Unknown_Soldier_(1955_film)
```

RDF-määrittelyn (ts. tiedoston) oman eli *kantanimiavaruuden* (base) ilmaisemiseen käytetään muotoa:

kantanimiavaruus

`@base` *nimiavaruus* .

[4] https://www.w3.org/RDF/Validator/

Tämän jälkeen RDF-verkossa kulmasulkujen sisään kirjoitettujen paikallisnimien, jotka on kirjoitettu muodossa <*paikallisnimi*> ilman nimiavaruutta, oletetaan kuuluvan kantanimiavaruuteen. Esimerkiksi määrittelyn

```
@base <http://ldf.fi/kirjallisuus/> .
```

jälkeen nimiavaruudeton lyhennetty tunniste

```
<The_Unknown_Soldier_(1955_film)>
```

tarkoittaa samaa kuin:

```
<http://ldf.fi/kirjallisuus/The_Unknown_Soldier_(1955_film)>
```

5.2 Yksinkertaisempaa koodia: Turtle

Täydellisten kolmikoiden luetteleminen tekee N-Triples -koodista toisteista, työläästi kirjoitettavaa ja vaikeasti luettavaa ihmiselle. Tämän johdosta käyttöön on otettu monia käteviä lyhennystapoja, jotka on käyttävissä RDF:n *Turtle-notaatiossa*[5]. Turtle on nykyisin käytetyin ja helppolukuisin RDF-notaatio. Esittelemme seuraavassa käyttökelpoisia muotoja, joilla Turtle laajentaa ja yksinkertaistaa N-Triples -kielen ilmaisumahdollisuuksia.

5.2.1 Monta samaa ominaisuutta

Jos resurssilla on useita samannimisiä ominaisuuksia, voidaan ne luetella pilkuilla erotettuna. Esimerkiksi *Tuntemattoman sotilaan* erikieliset nimikkeet voidaan luetella muodossa

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix db: <http://dbpedia.org/resource/> .

<db:The_Unknown_Soldier_(1955_film)> rdfs:label
  "Tuntematon sotilas (1955 filmi)"@fi,
  "The Unknown Soldier (1955 film)"@en .
```

tarvitsematta kirjoittaa auki erillisiä kolmikoita.

[5] http://www.w3.org/TR/turtle/

5.2.2 Eri ominaisuudet samalla kertaa

Resurssin eri ominaisuuksia voidaan luetella näppärästi yhdellä kertaa puolipisteellä erotettuna, jolloin ei tarvitse toistaa subjektia moneen kertaa. *Tuntematon sotilas* -esimerkin kolme kolmikkoa voidaan näin esittää lyhyemmin muodossa:

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix db: <http://dbpedia.org/resource/> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix koko: <http://www.yso.fi/onto/koko/> .

<db:The_Unknown_Soldier_(1955_film)>
  <dbo:director> <db:Edvin_Laine> ;
  <rdfs:label> "The Unknown Soldier (1955 film)"@en ;
  <dc:subject> <koko:p48202> .
```

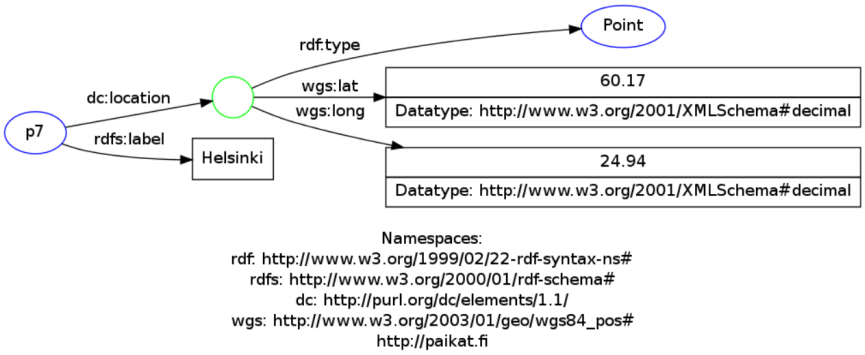
5.2.3 Ominaisuuden arvo omilla ominaisuuksilla

Ominaisuuden arvona oleva toinen resurssi ominaisuuksineen voidaan esittää hakasulkujen '[' ja ']' avulla erotettuna. Esimerkiksi alla on kuvattu *Helsingin* paikkatieto tällaisen resurssin avulla:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix wgs: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix : <http://paikat.fi> .

:p7 rdfs:label "Helsinki" ;
    dc:location [
      rdf:type :Point;
      wgs:lat 60.17;
      wgs:long 24.94
      ] .
```

nimetön solmu

Tällöin hakasulkujen sisältö määrittelee *nimettömän solmun* (blank node) ominaisuuksineen ja saadaan kuvan 5.2 mukainen graafi. Ympyrällä esitetty nimetön solmu edustaa tässä geografista pistettä, jolla on leveysaste (lat) ja pituusaste (long), ja joka kertoo Helsinki-nimisen resurssin sijainnin. Nimettömällä resurssilla on vain järjestelmän sisäisesti ja automaattisesti luotu tunniste, mikä yksinkertaistaa verkon tekstuaalista kuvausta. Tunniste on ainutkertainen URI-tunniste, johon ei ole mahdollista viitata muualta. Tästä ei ole haittaa, koska nimettömän solmun



Kuva 5.2: Graafi, jossa on nimetön solmu.

tunniste ei ole kiinnostava, ja solmua käytetään ainoastaan keräämään yhteen pisteeseen liittyvää koordinaattitietoa kertaluontoisesti.

Solmulle voitaisiin myös antaa tunniste, esimerkiksi `:point-254`:

```
...

:p7 rdfs:label "Helsinki" ;
  dc:location :point-254 .
:point-254 rdf:type :Point ;
  wgs:lat 25.1222 ;
  wgs:long 68.2344 .
```

Tällöin määrittely muuttuu kuitenkin monimutkaisemmaksi ja käyttäjän täytyy itse murehtia siitä, että solmun `:point-254` tunnistetta ei ole käytetty missään muualla, mistä järjestelmä huolehtii automaattisesti nimettömiä solmuja käytettäessä. Tunnisteen käytön etuna kuitenkin on, että siihen voidaan myöhemmin viitata, ja esimerkiksi antaa paikkaa kuvaavalle solmulle muualla uusia lisäominaisuuksia kuten paikan korkeus merenpinnasta.

Huomattavaa on, että käyttäjän nimeämän solmun tunniste on aina sama, kun taas nimettömän solmun tunniste luodaan aina uudelleen järjestelmän toimesta ja se voi olla erilainen eri järjestelmissä ja eri lukukerroilla, millä saattaa olla merkitystä ohjelmoinnissa. Jos esimerkiksi sama graafi talletetaan kahteen kertaan eri tavalla ja luetaan takaisin keskusmuistiin, voi nimettömille tunnisteille muodostua eri tunnisteet eikä niitä pystytä tunnisteen perusteella samaistamaan.

5.2.4 Tietotyypit

tietotyyppi

Kielimäären ohella literaaleihin voidaan liittää *tietotyyppi* (data type). Se ilmaistaan muodossa:

literaali^^*tietotyyppi*

Tietotyyppeinä voidaan käyttää laajaa joukkoa XML Schema -spesifikaatiossa

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

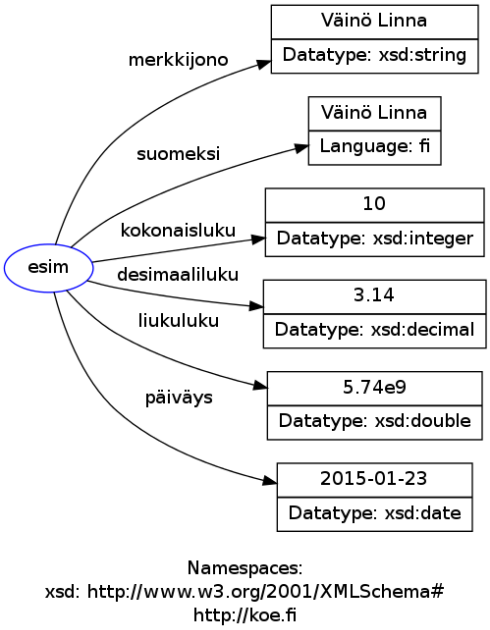
määriteltyjä perustietotyyppejä, kuten *merkkijono* (string), *kokonaisluku* (integer), *desimaaliluku* (decimal), *liukuluku* (double) ja *päiväys* (date). Alla on liitetty resurssiin `:esim` erityyppisiä literaajeja, ja syntyvä graafi on visualisoitu kuvassa 5.3. Siinä literaalin määreet on esitetty literaalia kuvaavan suorakaiteen sisällä.

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix : <http://koe.fi> .

:esim
  :merkkijono "Väinö Linna"^^xsd:string ;
  :suomeksi "Väinö Linna"@fi ;
  :kokonaisluku "10"^^xsd:integer ;
  :desimaaliluku "3.14"^^xsd:decimal ;
  :liukuluku "5.74e9"^^xsd:double ;
  :päiväys "2015-01-23"^^xsd:date .
```

Oletusarvoisesti lainausmerkeissä kirjoitettu literaali tulkitaan merkkijonoksi (`xsd:string`). Jos arvona on kokonaisluku tai pelkkä lukuarvo ilman lainausmerkkejä, osaavat järjestelmät asettaa tiedon tyypiksi jonkun numeerisen tyypin, vaikkei tietotyyppiä olisi erikseen merkitty. Aina ei tätä kuitenkaan voida tehdä yksikäsitteisesti: esimerkiksi 3.14 voi olla desimaali- tai liukuluku.

5.3 Verkon esittäminen RDF/XML-kielellä

Alkuperäinen RDF-kielen syntaksi määriteltiin XML-kielen avulla. Tämä notaatio on edelleen käytössä monissa yhteyksissä, kuten RDF-spesifikaatiossa, vaikka Turtlesta onkin tullut käytetyin tapa RDF-verkkojen esittämisessä. Seuraavassa esitetään siksi vain esimerkkinä



Kuva 5.3: Resurssin literaaliarvoja verkkona.

RDF/XML-kielestä aiempi kuvaus Tuntemattomasta sotilaasta – tarkemmin RDF/XML-kieleen voi tutustua verkossa[6].

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:db="http://dbpedia.org/resource/"
  xmlns:dbo="http://dbpedia.org/ontology/"
  xmlns:koko="http://www.yso.fi/onto/koko/">

<rdf:Description rdf:about="db:The_Unknown_Soldier_(1955_film)">
  <dbo:director>Edwin Laine</dbo:director>
  <rdfs:label>The Unknown Soldier (1955 film)</rdfs:label>
  <dc:subject rdf:resource="koko:p36271" />
</rdf:Description>

</rdf:RDF>
```

Tässä RDF/XML-kuvaus on annettu tiedostossa, jonka alussa kerrotaan tiedoston sisältävän XML-kieltä, ja varsinainen datan kuvaus on

[6] https://www.w3.org/TR/rdf-syntax-grammar/

tämän jälkeen `<RDF>`-merkkauksen sisällä. `RDF`-alkumerkkauksen sisällä määritellään nimiavaruudet tagin `xmlns:`*lyhenne* attribuutteina. RDF-merkkauksen sisällä voidaan kuvailla resurssit `<rdf:Description>`-merkkauksilla, jonka `rdf:about`-attribuutti identifioi kuvattavan resurssin ja alimerkkaukset kuvaavat resurssin ominaisuudet. Ominaisuudet voidaan antaa vaihtoehtoisesti myös attribuutteina.

5.4 Esimerkki tietojen yhdistämisestä

Tarkastellaan esimerkkinä tietojen yhdistämisestä kuvitteellista tilannetta, jossa data kerätään seuraavista organisaatioista:

- *Art* on taidemuseo.
- *Bio* on biografiakeskus, joka ylläpitää henkilörekisteriä.
- *Geo* on paikkatietoa tuottava maanmittauslaitos.
- *Onto* on asiasanastoja ja ontologioita ylläpitävä kirjastopalvelu.

Näillä organisaatiolla on käytössään seuraavat nimiavaruudet:

```
@prefix a: <http://art.org/> .
@prefix b: <http://bio.org/> .
@prefix g: <http://geo.org/> .
@prefix o: <http://onto.org/> .
```

Lisäksi käytämme esimerkissä seuraavia yleisiä nimiavaruuksia.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> . # SKOS-sanasto
@prefix dc: <http://purl.org/dc/elements/1.1/> . # Dublin Core
```

Organisaatioilla on tietokannoissaan taulukkomuotoista tietoa, esimerkiksi relaatiotietokantoja, jota ne haluavat julkaista yhdessä linkitettynä data toinen toistensa dataa rikastaen. Taidemuseossa Art on esimerkiksi taulukon 5.1 mukaista tietoa kokoelmissa olevista maalauksista.

Tunniste	Maalaus	Tekijä	Aika	Aihe
m-985	Aino triptyykki	A. Gallen-Kallela	1891	Kalevala
m-123	Mannerheimin muotokuva	A. Gallen-Kallela	1929	C. G. Mannerheim
...	...	...	...	...

Taulukko 5.1: Taidemuseossa olevia maalauksia.

Taulukkomuotoinen tieto voidaan muuttaa RDF-verkoksi suoraviivaisesti siten, että jokainen rivi vastaa yhtä uutta resurssia. Resurssin ominaisuuksiksi valitaan sarakkeiden nimet ja ominaisuuksien arvoksi sarakkeiden arvot kyseisellä rivillä. Näin saataisiin esimerkiksi taulukon ensimmäiselle riville seuraava RDF-muoto:

```
<r1> <Tunniste> "m-985" ;
  <Maalaus> "Aino-triptyykki" ;
  <Tekijä> "A. Gallen-Kallela" ;
  <Aika> 1891 ;
  <Aihe> "Kalevala" .
```

Kuva 5.4: Aino-triptyykki (Ateneumin taidemuseon kuva), jonka maalasi Akseli Gallen-Kallela Pariisissa 1891.

Tällainen muunnos ei kuitenkaan tuota hyvin linkitettyä dataa seuraavista syitä:

1. *Ominaisuudet eivät linkity.* Ominaisuusresurssit, kuten `<Tekijä>`, ovat vain Art-museon sisäisessä käytössä eivätkä linkity muissa museoissa tai kansainvälisesti käytettyihin resursseihin. Tekijyyteen liittyvät kyselyt yli organisaatiorajojen eivät silloin onnistu.

2. *Ominaisuuksien arvot eivät linkity.* Ominaisuuksien arvot ovat literaalidataa, joka ei linkity vastaaviin resursseihin. Esimerkiksi *C. G.*

Mannerheim voi viitata paitsi sotamarsalkkaan myös hänen saman nimiseen isoisäänsä, joka oli mm. kuuluisa hyönteistutkija.

3. *Uusiin resursseihin on voitava viitata ulkopuolelt*a. Muille muistiorganisaatioille pitäisi tarjota tapa viitata Art-museon omiin teoksiin resursseina. Toisessa museossa voi esimerkiksi olla Aino-triptyykin jäljennös.

Nämä ongelmat voidaan ratkaista organisaatioiden välisen yhteistyön kautta seuraavalla tavalla:

1. *Ominaisuuksien linkittäminen.* Käytetään ominaisuuksille standardeja, esimerkiksi Dublin Core -standardin mukaista resurssia `dc:creator <Tekijä>`:n sijaan, tai sillataan itse käytetyt ominaisuudet tällaisten aliominaisuuksiksi (palaamme aliominaisuuksien käyttöön tarkemmin tuonnempana). Näin standardit yhdistävät ja tekevät ymmärrettäväksi koneelle eri organisaatioiden omien ominaisuusresurssien tulkinnan.

2. *Ominaisuuksien arvojen linkitys.* Käytetään ominaisuuksien arvoina literaalidatan sijasta resursseja, jotka on mahdollisesti määritelty tarkemmin toisten tahojen toimesta. Esimerkissämme tulemme näkemään että *Bio* julkaiseen omassa nimiavaruudessaan biografista tietoa henkilöistä RDF-muodossa, mistä löytyy mm. Akseli Gallen-Kallelan URI-tunniste `b:A_Gallen-Kallela` ja sotamarsalkkamme URI-tunniste `b:C_G_Mannerheim-1`.

3. *Uusien resurssien nimeäminen. Art* ottaa käyttöön pysyvien URI-tunnisteiden muodostamiskäytännön, jotta muut organisaatiot voivat viitata sen kokoelmien resursseihin. Usein sopiva tunniste on tietokannassa jo jonkin sarakkeen arvona valmiiksi olemassa, kuten tässäkin tapauksessa. Käytetään HTTP-tunnisteen osana siis esimerkiksi luetteloinnissa käytettyä tunnusta `a:m-985`, eikä esimerkiksi automaattisesti rivinumeron mukaan luotua numeerista avainta, joka mahdollisesti myöhemmin muuttuu taulukon koon muuttuessa.

Nämä seikat ja yhteisön muiden jäsenten tunnisteet huomioon ottaen voidaan Art-museon kokoelma muuntaa seuraavaan hyvin linkittyvään muotoon.

```
a:m-985 rdfs:label "Aino-triptyykki" ;
  dc:creator b:A_Gallen-Kallela ;
  dc:date 1891 ;
  dc:subject o:Kalevala .
a:m-123 rdfs:label "Mannerheimin muotokuva" ;
  dc:creator b:A_Gallen-Kallela ;
  dc:date 1929 ;
  dc:subject b:C_G_Mannerheim-1 .
```

Biografiakeskuksen osalta oletamme saatavilla olevan taulukkodataa, joka liittää henkilöresurssit kuntiin, joissa henkilön suku on vaikuttanut:

Henkilö	**Syntymäpaikka**	...
A. Gallen-Kallela	Lemu	...
C. G. Mannerheim	Askainen	...
...	...	...

Taulukko 5.2: Henkilörekisteri Biografiakeskuksessa

Henkilöiden syntymäpaikat voidaan esittää RDF-muodossa vaikkapa näin:

```
b:A_Gallen-Kallela g:from g:Lemu .
b:C_G_Mannerheim-1 g:from g:Askainen .
```

Paikkatieton osalta taas käytössä on Geo-maanmittauslaitoksen taulukko, jossa kerrotaan kuntien sijoittumisesta Suomen eri alueille, josta saadaan tuotettua seuraavanlaista RDF-verkkoa:

```
g:Lemu
  rdf:type o:Kunta ;
  g:partOf g:Varsinais-Suomi .
g:Askainen
  rdf:type o:Kunta ;
  g:partOf g:Varsinais-Suomi .
```

Kirjaston ylläpitämässä ontologiapalvelussa määritellään (myöhemmin tarkemmin esiteltävää) SKOS-standardia käyttäen mm. asiasanastoja, joista selviää mm. mitä *kunta*, *lääni* ja *Kalevala* tarkoittavat:

```
o:Kunta a skos:Concept .
o:Lääni a skos:Concept .
o:Kalevala a skos:Concept .
```

Jos yhteisö käyttää kokoelmatietojensa esittämisessä toistensa tunnisteita ja yhdessä sovittuja asiasanastoja, voidaan eri tahojen tieto yhdistää muodostamalla eri RDF-verkkojen joukko-opillinen unioni yhdellä rivillä ohjelmakoodia. Lopputulos on esitetty kuvassa 5.5.

Laajemmasta graafista voidaan esimerkiksi havaita Mannerheimin muotokuvaa tutkittaessa, että taiteilija ja malli ovat kotoisin Varsinais-Suomesta ja päätellä, että kenties herrat tunsivat hengenheimolaisuutta keskenään tämän yhteyden kautta?

Mikäli eri organisaatioiden käyttämät tunnisteet eivät ole samoja, syntyisi kuvan 5.5 verkkoon esimerkiksi useita erillisiä gallen-kalleloita, mannerheimeja, paikkoja ja muita käsitteitä eli semanttinen sekamelska ilman linkitystä. Semanttinen yhteentoimivuus edellyttää sopimista yhteisten aineistojen kuvailussa käytettävien ontologioiden käyttämisestä. Tämä oli kansallisen FinnONTO-projektin keskeisenä tavoitteena[7]. Vaihtoehtona on vaivalloinen eri organisaatioiden datojen yhdistäminen jälkikäteen. Hankkeen mottona olikin siksi Albert Einsteinin lausuma viisaus: “Intellektuellit ratkovat ongelmia, mutta nerot estävät niiden syntymisen”.

5.5 Muita RDF:n ominaisuuksia

RDF:n ydin on edellä kuvattu yksinkertainen verkkoperustainen tietomalli. Standardiin kuuluu lisäksi eräitä verkkojen avulla kuvattuja kehittyneempiä perustietorakenteita ja ilmaisumuotoja, joista tärkeimpiä luettellaan alla laajemman yleiskuvan muodostamiseksi RDF-suosituksesta.

5.5.1 Säiliöt

säiliö

Säiliöt (container) ovat tietorakenteita, joiden avulla on mahdollista kuvata toisiinsa liittyvää joukkoa resursseja standardilla tavalla. RDF-spesifikaatiossa

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
```

[7] http://seco.cs.aalto.fi/projects/finnonto/

on määritelty taulukon 5.3 luokkia resurssi-yksilöiden säiliöiden esittämistä varten. Säiliössä olevat yksilöt esitetään ominaisuuskaarilla säiliöstä siihen kuuluviin jäseniin. Määrittelyjen tavoitteena on luoda yhtenäinen käytäntö yksilöiden erilaisten joukkojen esittämistä varten verkossa.

Luokka	Merkitys
`rdf:Bag`	*Monijoukko* eli joukko, jossa voi olla useita samoja alkioita. Alkioihin viitataan ominaisuudella `rdf:_`n, missä n on kokonaisluku, mutta monijoukon alkioiden järjestyksellä ei ole merkitystä. Monijoukon tyypiksi merkitään luokka `rdf:Bag` ominaisuudella `rdf:type`.
`rdf:Seq`	*Sekvenssi* (sequence) on monijoukko `rdf:Bag`, jonka alkioiden oletetaan olevan `rdf:_`n-ominaisuuksien mukaisessa järjestyksessä. Sekvenssin tyypiksi merkitään luokka `rdf:Seq` ominaisuudella `rdf:type`.
`rdf:Alt`	*Vaihtoehdot* (alternatives) on kuten `rdf:Bag`, mutta ensimmäinen alkio (`rdf:_1`) oletetaan valituksi vaihtoehdoksi kokoelmasta. Vaihtoehto-säiliön tyypiksi merkitään luokka `rdf:Alt` ominaisuudella `rdf:type`.

monijoukko

sekvenssi

vaihtoehdot

Taulukko 5.3: RDF-spesifikaatioon kuuluvat *säiliöluokat* (container), joiden avulla voidaan esittää yksilöiden kokoelmia. Kokoelmaluokkien yläluokka on `rdfs:Container`.

kokoelmaluokka

Säiliön jäsenet liitetään säiliöön ominaisuuksilla, joiden muoto on `rdf:_`n, missä $n > 0$ on positiivien kokonaisluku. Jäsenet voidaan RDF/XML-kielessä ilmaista numeroimatta ominaisuudella `rdf:li`. Alla on esimerkki, jossa säiliöön kuuluu joukko henkilöitä ja määrittelystä syntyy kuvassa 5.6 oleva RDF-verkko.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://esim.fi/henkilö/sanasto#">
  <rdf:Description rdf:about="http://esim.fi/yhteisö/Tuusula">
    <s:henkilö>
       <rdf:Bag>
          <rdf:li rdf:resource="s:Jean"/>
          <rdf:li rdf:resource="s:Aino"/>
          <rdf:li rdf:resource="s:Juhani"/>
       </rdf:Bag>
    </s:henkilö>
  </rdf:Description>
</rdf:RDF>
```

5.5.2 Kokoelmat

kokoelma
lista

Säiliö on avoin tietorakenne, johon voidaan lisätä uusia jäseniä. RDF:ssä on myös määriteltynä tietorakenne *kokoelma* (collection), jonka avulla voidaan esittää suljettuja järjestettyjä joukkoja *listana* (list). Sen jäsenet kiinnitetään kokoelman luomisen yhteydessä. Listoja esitetään tietotekniikassa sisäkkäisinä sulkurakenteina. Esimerkiksi listassa

```
(:a :b (:c :a) :d)
```

on neljä alkiota, joista kolmas on kaksi alkiota sisältävä toinen lista.

pää
häntä

Listan alkiot ovat RDF-suosituksen mukaan tyyppiä `rdf:List` (ts. `rdf:List`-luokan yksilöitä), joita ketjutetaan toisiinsa `rdf:first` (listan ensimmäinen alkio eli *pää* (head)) ja `rdf:rest` ominaisuuksilla (listan loput alkiot eli *häntä* (tail)). Erityinen tyhjä lista `rdf:nil` häntänä päättää listan. Alla on esimerkki kolmen henkilön muodostamasta listasta RDF/XML-kielellä. Tätä vastaava RDF-verkko on esitetty kuvassa 5.7.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://esim.fi/henkilö/sanasto#"
         xmlns:h="http://esim.fi/henkilö/henkilö#">
  <rdf:Description rdf:about="http://esim.fi/yhteisö/Tuusula">
    <s:henkilö rdf:parseType="Collection">
       <rdf:Description rdf:about="h:Jean"/>
       <rdf:Description rdf:about="h:Aino"/>
       <rdf:Description rdf:about="h:Juhani"/>
    </s:henkilö>s
  </rdf:Description>
</rdf:RDF>
```

Sama lista ja verkko voidaan esittää yksinkertaisemmin Turtle-kielellä näin:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix s: <http://esim.fi/henkilö/sanasto#> .
@prefix h: <http://esim.fi/henkilö/henkilö#> .

<http://esim.fi/yhteisö/Tuusula> s:henkilö (h:Jean h:Aino h:Juhani) .
```

Turtle-notaatiolla listojen esittäminen on helppoa kirjoittamalla lista yksinkertaisesti sulkuihin perinteiseen tapaan. Turtle-jäsennin jäsentää tämän automaattisesti verkoksi, jossa listat on esitetty RDF-verkkona `rdf:first` ja `rdf:rest` ominaisuuksien avulla.

5.5.3 Reifikaatio

RDF:n *reifikaatio* (reification) on mekanismi, jonka avulla tietokolmikkoon voidaan liittää sitä kokonaisuutena kuvaavaa metatietoa. Näin voidaan esimerkiksi kertoa kolmikkokohtaisesti, mistä tietolähteestä kukin tieto on peräisin. Tällainen metatieto voi olla tarpeen mm. esitettäessä tietoon liittyviä väitteitä, esimerkiksi miten varmaa tieto on tai *provenienssitietoa* (provenience data) siitä, mistä kukin tieto on peräisin. Se voi olla hyödyllistä tiedon hallinnan kannalta tai tiedon luotettavuutta arvioitaessa.

reifikaatio

provenienssi

Tarkastellaan esimerkkinä kolmikkoa:

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

:Maa :muoto :litteä .
```

Tavoitteena on kertoa, että tämä tieto pitää tai ei pidä paikkansa tietolähteestä riippuen. Koska kolmikolla ei ole identiteettiä (URI-tunnistetta), ei siihen voida viitata. Reifikaatiossa kolmikko esitetään siksi uutena tyyppiä `rdf:Statement` olevana resurssina, jonka subjekti, predikaatti ja objekti annetaan vastaavilla ominaisuuksilla `rdf:subject`, `rdf:predicate` ja `rdf:object`. Kun kolmikko on nyt resurssi, siihen voidaan viitata ja esimerkiksi kertoa, että tieto on Litteä Maa -järjestön väite ja että HELDIG-keskus kiistää sen (kuva 5.8):

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://esim.fi> .

:väite-23 rdf:type rdf:Statement ;
  rdf:subject :Maa ;
  rdf:predicate :muoto ;
  rdf:object :litteä .

<https://theflatearthsociety.org> :väittää :väite-23 .
<https://www.heldig.fi> :kiistää :väite-23 .
```

Näin semanttisessa webissä voidaan esittää ristiriitaistakin tietoa logiikan keinoin ja tehdä ero tosiasioiden ja erilaisten käsitysten välillä.

Toinen RDF:n tapa esittää tietoon liittyvää korkeammanasteista tietoa on RDF Dataset Language TriG[8]. TriG on Turtle-notaation laajennus, jolla kolmikkojoukot voidaan määritellä useampana nimettynä graafina,

[8] https://www.w3.org/TR/2014/REC-trig-20140225/

joilla on omat URI-tunnisteensa. Liittämällä graafin tunnisteeseen metatietoa, voidaan antaa korkeammanasteisia kuvauksia kokonaisista graafeista, ei vain kolmikoista kuten reifikaatiossa.

5.6 JSON-LD

Modernit web-sovellukset perustuvat yhä enemmän asiakaspuolen, ts. selaimen ohjelmointiin. Ylivoimaisesti käytetyin ohjelmointikieli tähän on JavaScript. JavaScriptin keskeinen rakenteisen tiedon esittämismuoto on *JavaScript Object System* eli *JSON*, josta on muodostunut tärkeä standardi sekä tiedon esittämisessä että välityksessä. RDF:n tapaan JSON perustuu ominaisuuksien esittämiseen, mikä mahdollistaa RDF-verkkojen kuvaamisen JSON:lla suoraviivaisesti. Linkitetyn datan JSON-muodosta käytetään nimitystä *JSON-LD*[9]. JSON-LD-muodon etuna on, että JavaScript tukee suoraan JSON-objektien käsittelyä.

JSON

JSON-LD

JSON-objekti koostuu joukosta *ominaisuus-arvo-pareja* (key-value pair), jotka ovat muotoa:

ominaisuus-arvo-pari

"*ominaisuus*": "*arvo*"

Ominaisuuden arvo voi olla literaalidataa, hierarkkisesti toinen JSON-objekti tai lista arvoja. Esimerkiksi:

```
{
  "nimi": "Jean Sibelius",
  "syntyi": "1865-12-08",
  "puoliso": "http://dbpedia.org/resource/Aino_Sibelius"
}
```

JavaScriptissä tällaisesta JSON-objektista voidaan lukea tehokkaasti ominaisuuksien arvoja ja muokata niitä. Notaatio on lisäksi tiivis ja helppolukuinen.

JSON-LD määrittelee joukon ominaisuuksia ja sopimuksia, joiden avulla RDF-rakenteita voidaan esittää JSON-muodossa. Nämä ominaisuudet ovat @-alkuisia. Keskeinen ominaisuus on `@context`, jonka avulla määritellään JSON-LD-rakenteen tulkintakonteksti, esimerkiksi käytettävissä olevia lyhennysmerkintöjä vähän samaan tapaan kuin nimiavaruuksia

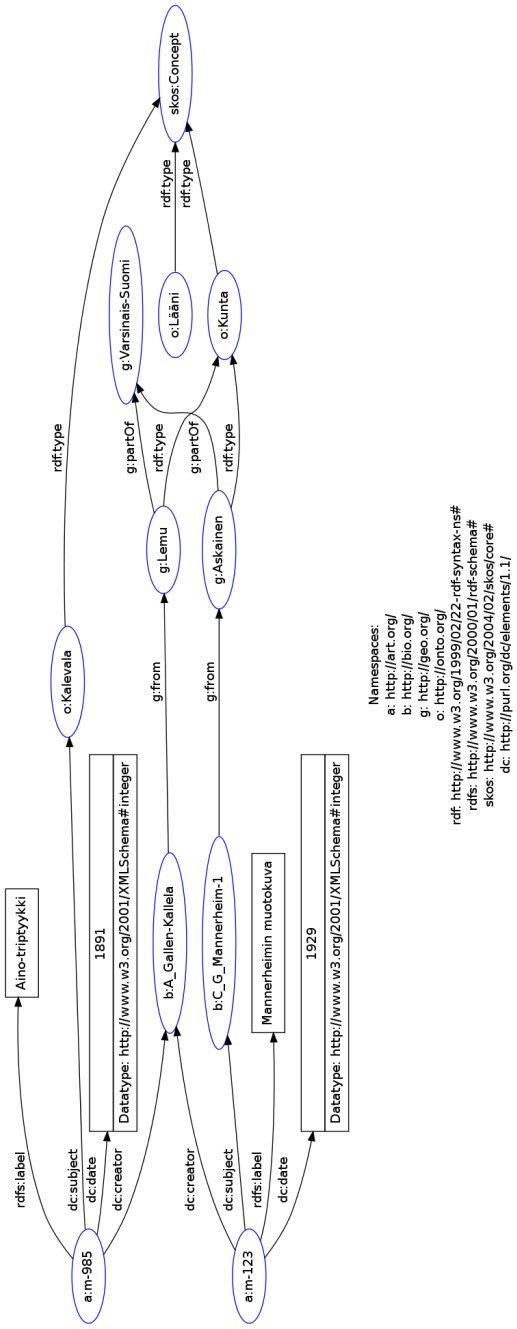
[9] http://json-ld.org/spec/latest/json-ld/

käytettäessä. Eri JSON-LD-objektit voivat viitata samaan kontekstiin URL-viittauksen avulla. `@type`-ominaisuudella taas voidaan kertoa solmun tyyppi. Alla on esimerkkinä määritelty Helsingissä pidetty tilaisuus HELDIG Kick-off Symposium. Siinä `@context` ominaisuuden arvona oleva objekti määrittelee nimiavaruudet `helka` ja `xsd` ja kertoo, että tässä kontekstissa `helka:aika` ominaisuuden tyyppi on XML-skeemassa määritelty päiväys `dateTime`. Varsinainen data on esitelty konkteksin jälkeen:

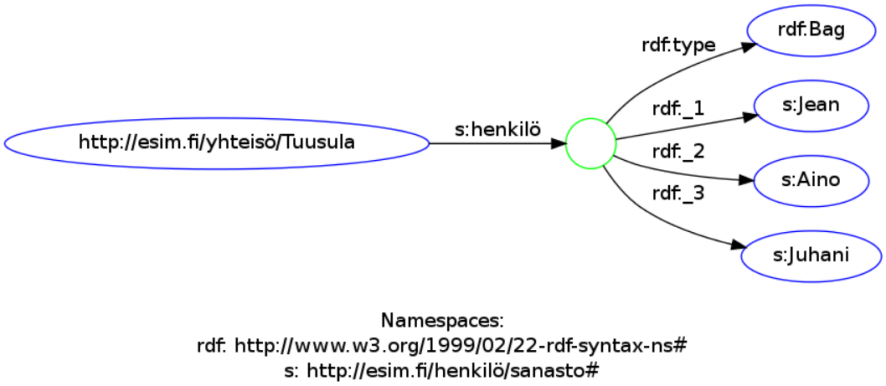
```
{
  "@context": {
    "helka": "http://heldig.fi/kalenteri#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "helka:aika": {
      "@type": "xsd:dateTime"
    }
  },
  "helka:otsikko": "HELDIG Kick-off Symposium",
  "helka:paikka": "Helsingin yliopisto, Pieni juhlasali",
  "helka:aika": "2016-10-05"
}
```

Tässä tapauksessa rakenteessa on kolme ominaisuutta, ja se voidaan muuntaa alla olevaksi kolmen kolmikon RDF-verkoksi, jossa `_:b0` on järjestelmän luoma nimettömän solmun tunniste:

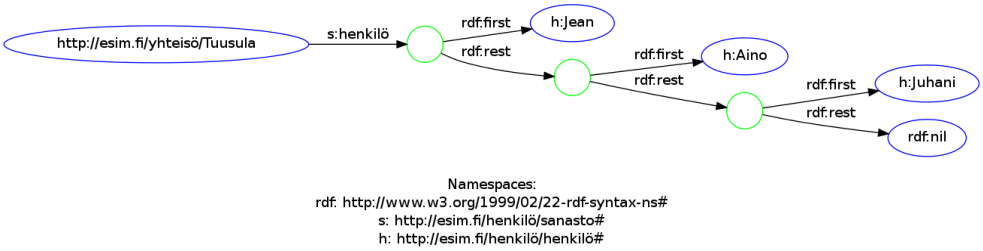
```
_:b0 <http://heldig.fi/kalenteri#aika>
    "2016-10-05"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
_:b0 <http://heldig.fi/kalenteri#otsikko>
    "HELDIG Kick-off Symposium" .
_:b0 <http://heldig.fi/kalenteri#paikka>
    "Helsingin yliopisto, Pieni juhlasali" .
```



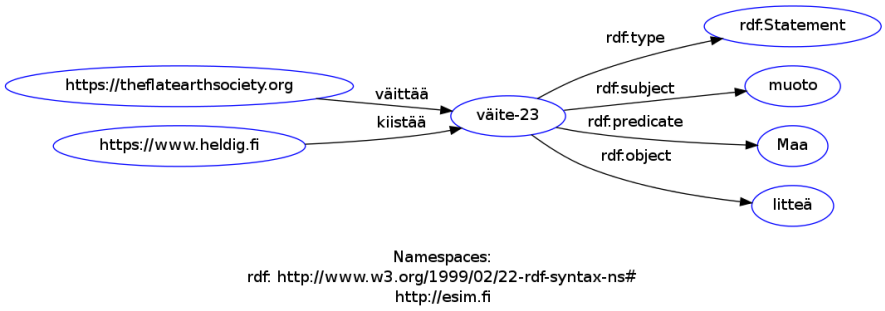
Kuva 5.5: Eri organisaatioiden data yhdistyy automaattisesti laajemmaksi semanttiseksi verkoksi.



Kuva 5.6: Esimerkki henkilöistä muodostuvasta kokoelmasta, joka on tyyppiä monijoukko (bag).



Kuva 5.7: Henkilöiden listan (h:Jean h:Aino h:Juhani) esitysmuoto RDF-verkkona



Kuva 5.8: Reifikaatio

Luku 6

Tiedon haku ja ylläpito: SPARQL

Semanttisen webin sovellukset kehitetään usein siten, että käyttöliittymä ja sovelluksen toiminnallisuus toteutetaan selaimessa JavaScriptillä ja erilaisilla siihen perustuilla kirjastoilla ja kehitysympäristöillä. Selaimen sovellus kyselee dataa SPARQL-kielen avulla verkosta. SPARQL (SPARQL Protocol and RDF Query Language) on semanttisen webin kysely- ja tiedonhallinnan kieli, jonka avulla voidaan kysellä RDF-muotoista tietoa SPARQL-palvelupisteistä ja ylläpitää niiden tietosisältöjä. Kieli muistuttaa relaatiotietokannoissa käytettyä SQL-kieltä, mutta soveltuu verkkomuotoisen RDF-standardin mukaisen datan käsittelyyn. Linkitetty data muodostaa webiin ikään kuin jättimäisen hajautetun tietokannan, jonka sisällöt on talletettu tiedon julkaisijoiden ylläpitämiin SPARQL-palvelupisteisiin heidän hallinnoimissa HTTP-osoitteissa.

SPARQL on yksi keskeisimmistä semanttisen webin standardeista ja ohjelmoinnin työkaluista. Sen ensimmäinen versio SPARQL 1.0[1] julkaistiin vuoden 2008 tammikuussa. Standardia laajennettiin merkittävästi vuoden 2013 maaliskuussa julkaisussa SPARQL 1.1 -versiossa, jonka dokumentaatio koostuu yhdestätoista eri osadokumentista[2]. Kyselyjen muodostaminen ja kielen käyttäminen on kuitenkin varsin suoraviivaista.

[1] http://www.w3.org/TR/rdf-sparql-query/

[2] Järjestelmän dokumentaatio ja ohjeistusta löytyy W3C:n verkkosivuilta: http://www.w3.org/TR/sparql11-query/

Tässä luvussa esitellään ensin SPARQL-datapalvelupisteen käsite ja sen käyttöä. Tämän jälkeen kerrotaan, miten SPARQL-kielen avulla voidaan kysellä dataa ja muokata sen sisältöä.

6.1 SPARQL-datapalvelun perustaminen

SPARQL-kyselyiden tekemistä varten tarvitaan SPARQL-datapalvelu, joka voi olla verkossa tai jollaisen voi itse asentaa helposti omalle koneelle. Verkossa on käytettävissä lukuisia julkisia SPARQL-datapalvelupisteitä, joiden dataa voi kysellä ja hyödyntää ohjelmallisesti HTTP-kyselyiden avulla. Esimerkiksi Wikipedian dataa voi kysellä DBpedian ja Wikidatan palvelupisteiden

```
http://dbpedia.org/sparql
http://query.wikidata.org
```

kautta ja lukuisia suomalaisia datajoukkoja voi käyttää Linked Data Finland -palvelun eri SPARQL-rajapintojen avulla:

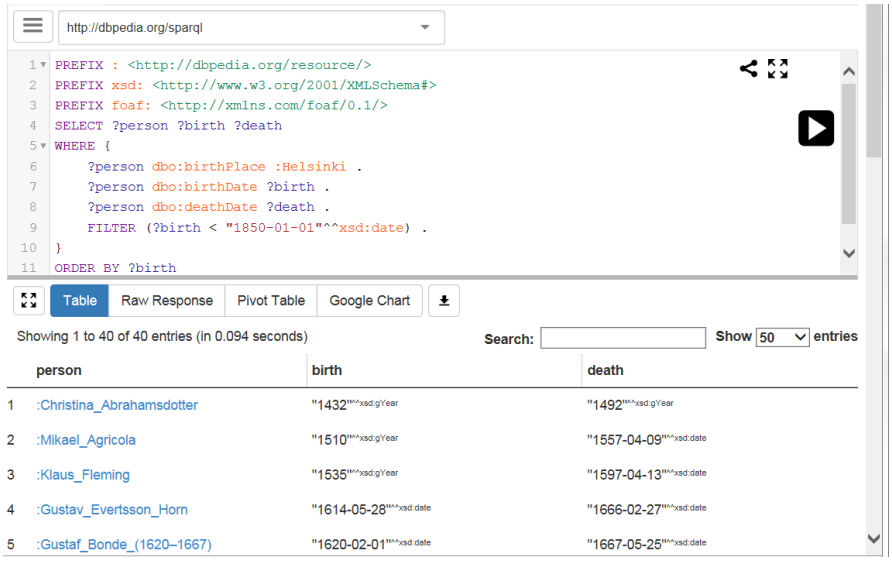
```
http://ldf.fi/
```

Kyselyiden testaamista varten on käytettävissä helppokäyttöisiä web-käyttöliittymiä, joissa voi määritellä käytettävän datapalvelun, kirjoittaa kyselyn ja suorittaa haun. SPARQL-käyttöliittymän avulla voi paitsi hakea tietoa, myös tutustua sen rakenteisiin ja muokata palvelupisteen dataa, mikäli tämä on käyttäjälle sallittu.

Kuvassa 6.1 on esimerkkinä paljon käytetty open source -kyselykäyttöliittymä YASGUI[3]. Sen yläreunassa käyttäjä on valinnut käytettäväksi DBpedian SPARQL-datapalvelun `http://dbpedia.org/sparql`. Tämän alle on kirjoitettu kysely SPARQL-kielellä. Oikeassa reunassa olevaa nappia painamalla YASGUI on hakenut DBpediasta kyselyn vastaukset ja näyttää ne taulukkomuodossa käyttöliittymän alaosassa. Kyselyiden vastauksille voi valita taulukkomuodon (Table) ohella myös muita esitysmuotoja. Vasemmassa ylänurkassa olevan työkaluvalikon kautta kyselyjärjestelmän toimintaa voi säätää eri tavoilla.

RDF-tietokannan ja SPARQL-palvelupisteen voi asentaa myös omalle tietokoneelle paikalliseen käyttöön; web-palvelimelle asennettuna sama

[3]Käyttöliittymää voi käyttää osoitteessa `http://yasgui.org/`.



Kuva 6.1: YASGUI-käyttöliittymä SPARQL-kyselyille. Haun kohteena ovat ennen v. 1850 Helsingissä syntyneet henkilöt DBpediassa.

palvelu on käytettävissä verkon kautta. Esimerkiksi Linked Data Finland -palvelussa käytetty Apache Jena Fuseki RDF-tietokanta ja -palvelu voidaan ottaa käyttöön seuraavalla tavalla[4]:

1. Lataa Apache Jena Fuseki -ohjelmistopaketti verkosta ja pura se johonkin hakemistoon. Koneessa on oltava asennettuna Java-ympäristö; asenna sellainen tarvittaessa.

2. Käynnistä ohjelmistoon sisältyvä palvelin hakemistosta (esimerkiksi Windowissa voit kaksoisklikata tiedostoa `fuseki-server.bat`). Fuseki on Java-pohjainen ja toimii samalla tavalla eri käyttöjärjestelmissä.

3. Avaa selaimessasi alla oleva paikallisosoite, jota äsken käynnistämäsi palvelin kuuntelee:

   `http://localhost:3030`

4. Avautuvan käyttöliittymän linkkien ja painikkeiden avulla voit ladata järjestelmään RDF-tiedostoja ja luoda graafeja.

[4] https://jena.apache.org/documentation/fuseki2/

5. Tämän jälkeen voit tehdä dataan kyselyjä ja muokata dataa samalla tavalla kuin verkkoversiolla, mutta kaikilla datan ylläpito-oikeuksilla. Myös SPARQL-palvelupisteen ohjelmallinen rajapinta on käytettävissä sovellusten kehittämistä varten.

nimetty graafi
oletusgraafi

SPARQL-palvelupisteen data koostuu joukosta *nimettyjä graafeja*. Näitä voidaan käsitellä joko erikseen tai yhtenä kokonaisuutena, josta käytetään nimitystä *oletusgraafi* (default graph). SPARQL-kieleen kuuluu välineet sekä

- graafimuotoisen datan kyselyitä varten (graph query) että
- komennot palvelupisteen graafien luomiseen, poistamiseen ja muokkaamiseen (graph management, graph updata).

Seuraavassa esitetään ensin yhteenveto SPARQL-kielen neljästä eri kyselytyypeistä ja tämän jälkeen keskeiset komennot graafien luomista ja ylläpitoa varten.

6.2 Peruskyselyt

`SPARQL`-kielen ydin koostuu neljästä eri kyselytyyppistä:

1. `SELECT`-kyselyllä voidaan hakea vastauksia kyselyihin. Vastauksena palautetaan arvosijotuksia kyselyssä esiintyville muuttujille.
2. `ASK`-kyselyllä voidaan selvittää, toteuttaako RDF-tietokanta kyselyssä ilmaistun ehdon vai ei. Vastauksena palautetaan totuusarvo.
3. `DESCRIBE`-kyselyllä voidaan hakea annettuun URI-tunnisteeseen liittyvä, sitä kuvaileva data. Vastauksena palautetaan RDF-muotoista dataa.
4. `CONSTRUCT`-kyselyllä voidaan muodostaa RDF-datan perusteella uutta RDF-muotoista dataa, joka palautetaan kyselyn arvona.

Kielen ensimmäinen spesifikaatio SPARQL 1.0 koostui olennaisesti näiden kyselytyyppien määrittelystä. Uusi laajempi standardi SPARQL 1.1 toi kyselyihin useita merkittäviä laajennuksia sekä kokonaan uuden mahdollisuuden graafien muokkaamiseen. Tarkastelemme seuraavaksi ensin kyselyiden perustyyppejä ja tämän jälkeen niihin tehtyjä laajennuksia.

6.2.1 Tiedon haku: SELECT

SELECT-kysely muistuttaa relaatiotietokantojen SQL-kielen vastaavaa kyselytyyppiä. SPARQL-kielessä kyselyn ideana on tiedontarpeen muotoileminen *graafihahmona* (graph pattern). Hahmo on Turtle-notaatiolla kirjoitettu kolmikkojoukko sillä lisämahdollisuudella, että kolmikoissa voidaan käyttää resurssien ja literaalien ohella niihin viittaavia *muuttujia* (variable). Muuttujat esitetään kirjoittamalla niiden eteen '`?`' tai '`$`' eli muodossa `?`*muuttuja* tai `$`*muuttuja*. Kysely suoritetaan sovittamalla graafihahmoa graafissa olevaan dataan: vastauksena saadaan kaikki mahdolliset yhdistelmät hahmon muuttujien arvoja, jolla hahmo on sovitettavissa dataan.

graafihahmo

muuttuja

Tarkastellaan esimerkkinä seuraavan listauksen dataa:

```
@prefix :      <http://koe.fi/> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix db:    <http://dbpedia.org/resource/> .
@prefix dc:    <http://purl.org/dc/elements/1.1/> .

:teos4  dc:creator  db:Hugo_Simberg ;
        dc:time     "1896"^^xsd:dateTime ;
        dc:title    "Kuoleman puutarha" .

:teos2  dc:creator  db:Hugo_Simberg ;
        dc:title    "Haavoittunut enkeli" .

:teos1  dc:creator  db:Akseli_Gallen-Kallela ;
        dc:time     "1896"^^xsd:dateTime ;
        dc:title    "Sammon puolustus" .

:teos3  dc:creator  db:Akseli_Gallen-Kallela ;
        dc:time     "1897"^^xsd:dateTime ;
        dc:title    "Lemminkäisen äiti" .
```

Tästä voidaan hakea esimerkiksi Akseli Gallen-Kallelan teokset kyselyllä

```
prefix dc: <http://purl.org/dc/elements/1.1/>  # Dublin Core -metadata
prefix db: <http://dbpedia.org/resource/>

SELECT ?maalaus ?nimi WHERE {
  ?maalaus dc:creator db:Akseli_Gallen-Kallela .
  ?maalaus dc:title ?nimi .
}
```

Huomaa, että etuliitteet (`prefix`) kirjoitetaan SPARQL-kielessä hieman eri tavalla kuin Turtlessa: '@'-merkki puuttuu samoin kuin lopun piste.

Kyselyn graafihahmon

```
?maalaus dc:creator db:Akseli_Gallen-Kallela .
?maalaus dc:title ?nimi .
```

ensimmäinen kolmikko sopii kaikkiin maalauksiin, joiden tekijä on Akseli Gallen-Kallela, ja toisen kolmikon sovittaminen dataan merkitsee sitä, että edellä löydetyille maalauksille haetaan vielä nimet. Vastauksena saatavat muuttujien arvokombinaatiot (maalaus-nimi) voidaan esittää luontevasti ihmisluettavassa muodossa taulukkona, jonka sarakkeina ovat kyselyn muuttujat ja jokainen rivi edustaa yhtä vastausta kyselyyn. Kuva 6.2 esittää Fusekin antamaa vastasta. Ohjelmointia ja konetta varten vastaustaulukko luetaan yleensä JSON-muodossa, jota on helppo käsitellä selaimessa JavaScript-kielellä.

	maalaus	nimi
1	<http://koe.fi/teos1>	"Sammon puolustus"
2	<http://koe.fi/teos3>	"Lemminkäisen äiti"

Kuva 6.2: SPARQL-kyselyn vastauksia taulukkona.

`SELECT`-kyselyn yleinen muoto on:

```
# Kyselyn oletus kanta-URI (valinnainen tieto)
BASE uri
# Kyselyn nimiavaruuksien määrittelyt (valinnainen tieto)
PREFIX nimiavaruus: uri
...
# Vastaukseen mukaan haluttujen muuttujien esittely
SELECT muuttujat
# Graafi, josta tietoa haetaan
FROM graafin osoite, tiedostopolku tai verkkosoite
# Kyselyn graafihahmon muotoilu
WHERE {
kolmikot, vaihtoehdot (FILTER, OPTIONAL ja UNION)-rajoitteet
}
# Vastauksen muotoilukomennot
tulosten järjestäminen (ORDER BY), määrä (LIMIT) ja valinta (OFFSET)
```

Kannan ja nimiavaruuksien esittelyn avulla sekä kyselyt että vastaukset voidaan esittää kompaktimmassa muodossa kuin täysiä URI-tunnisteita käytettäessä. SELECT-kyselyn muuttujien esittelyllä voi rajata vastaukseen mukaan vain kiinnostavat muuttujat ja annetussa järjestyksessä. Jos haluaa vastukseen kaikki muuttujat, voi käyttää merkintää:

```
SELECT *
WHERE {...
}
```

SELECT-kyselyssä käytettävissä olevan FROM-muodon avulla voi kertoa, mistä osoitteesta käsiteltävä RDF-data löytyy. Usein kysely kohdistetaan suoraan johonkin SPARQL-palvelupisteeseen, eikä FROM-määrittelyä tarvita.

Esimerkiksi seuraava kysely hakee DBpediassa (ks. kuva 6.1) olevat Helsingissä syntyneet henkilöt, listaa heidät syntymävuoden mukaisessa järjestyksessä (`ORDER BY`), mutta jättää näyttämättä kaksi ensimmäistä tulosta (`OFFSET`) ja rajoittaa haun 7 tulokseen (`LIMIT`).

```
PREFIX :    <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/> # DBpedian ontologia
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?henkilö ?syntyi ?kuoli
WHERE {
  ?henkilö dbo:birthPlace :Helsinki .
  ?henkilö dbo:birthDate ?syntyi .
  ?henkilö dbo:deathDate ?kuoli .
}
ORDER BY ?syntyi
OFFSET 2
LIMIT 7
```

Graafihahmon määrittelyssä voidaan hahmon määrittelyä monipuolistaa `FILTER`-, `OPTIONAL`-, `UNION`- muodoilla:

Vastausten suodattaminen: FILTER

Kyselyn graafihahmossa voi olla muuttujia sisältävien kolmikoiden ohella myös *suodattimia* (filter), jotka esitetään muodossa: *suodatin*

`FILTER (`*`ehto`*`)`

Suodattimen avulla poistetaan ratkaisujen joukosta pois ei-toivottuja ratkaisuja; vain sellaiset muuttujasijoitukset jäävät jäljelle, joilla suodattimen ehtolauseke palauttaa arvon tosi. Esimerkiksi seuraavassa haetaan

Funktiotyyppi	**Funktio**	**Merkitys**
Looginen	! && ||	negaatio konjunktio disjunktio
Matemaattinen	+, -, *, /	yhteen-, vähennys-, kerto- ja jakolasku
Vertailu	=, !=, >, <	Samuuden ja suuruuden vertailu
Tyypin testaus	isURL(x) isBlank(x) isLiteral(x) bound(x)	tosi, jos x on resurssi tosi, jos x on tyhjä solmu tosi, jos x on literaali tosi, jos x:llä on arvo
Lukufuntiot	str(x) lang(x) datatype(x)	Arvona palautuu x merkkijonona Arvona palautuu x:n kielitunnus Arvona palautuu x:n tietotyyppi
Muut funktiot	sameTerm(x,y) langMatches(x,y) regex(t,s,p)	Tosi, jos x ja y ovat sama termi Tosi, jos kielitunnus x vastaa y:tä Tosi, jos säännöllinen lauseke s sopii merkkijonoon t. Valinnaisella parametrilla p voi tarkentaa sovitusta.

Taulukko 6.1: SPARQL-muodoissa käytettävissä olevia funktiot

DBpediasta vain sellaisia syntyperäisiä helsinkiläisiä, jotka ovat syntyneet 1800-luvulla tai aiemmin.

```
PREFIX :    <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/> # DBpedian ontologia
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?henkilö ?syntyi ?kuoli
WHERE {
  ?henkilö dbo:birthPlace :Helsinki .
  ?henkilö dbo:birthDate ?syntyi .
  FILTER (?syntyi < "1900-01-01"^^xsd:dateTime )
}
ORDER BY ASC(?syntyi)
```

Suodatusehtoja voidaan muodostaa käyttämällä hyväksi taulukossa 6.1 lueteltuja loogisia, matemaattisia, vertailu ja muita funktioita. Lukufunktioiden avulla voidaan tehdä datamuunnoksia. Jos esimerkiksi literaali `"Matti Pellonpää"@en` on sidottu muuttujan `?nimi` arvoksi, palautuu kutsun `str(?nimi)` arvona `"Matti Pellonpää"` ilman kielitunnusta. Tätä voidaan sitten käyttää esimerkiksi etsittäessä 'Matti'-alkuisia syntyperäisiä Helsinkiläisiä säännöllisellä lausekkeella alla olevassa kyselyssä. Säännöllisen lausekkeen `^Matti` merkki '^' tarkoittaa merkkijonon alkua eli tässä haetaan vain 'Matti'-alkuisia nimiä.[5]

```
PREFIX :    <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?nimi ?henkilö
WHERE {
  ?henkilö dbo:birthPlace :Helsinki .
  ?henkilö rdfs:label ?nimi .
  FILTER (lang(?nimi)='en')
  FILTER (regex(str(?nimi),'^Matti','i'))
}
```

Valinnaisuuden ilmaiseminen: OPTIONAL

`OPTIONAL` *hahmo* -muodolla voidaan ilmaista ei-pakollinen hahmon osa. Jos esimerkiksi edellisessä henkilödatassa ei jokaisen henkilön kohdalla ole tiedossa kuolinvuotta (esimerkiksi elossa olevat henkilöt), voidaan henkilöt kuitenkin löytää näin, esimerkiksi kaikki ennen Pariisin rauhansopimusta Helsingissä syntyneet:

[5] Säännölliset lausekkeet ovat erittäin käyttökelpoisia SPARQL kyselyitä muotoiltaessa. Niihin voi tutustua verkossa esimerkiksi osoitteessa http://www.w3schools.com/jsref/jsref_obj_regexp.asp.

```
PREFIX :    <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/> # DBpedian ontologia
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT *
WHERE {
  ?henkilö dbo:birthPlace :Helsinki .
  ?henkilö dbo:birthYear ?syntyi .
  OPTIONAL {?henkilö dbo:deathYear ?kuoli }
  FILTER (?syntyi < "1947-02-10"^^xsd:date ) # Pariisin rauhansopimus
}
ORDER BY DESC (?syntyi) # Järjestä myöhäisin ensin
```

Tällöin vastauksessa ei kaikkien henkilöiden osalta ole tietoa kuolinajasta.

Vaihtoehtojen ilmaiseminen: UNION

Muodolla

hahmo-1 UNION *hahmo-2*

voidaan puolestaan ilmaista vaihtoehtoisia ratkaisuja samassa kyselyssä, esimerkiksi että henkilö on syntynyt joko Helsingissä tai Turussa:

```
PREFIX :    <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/> # DBpedian ontologia
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT *
WHERE {
  {?henkilö dbo:birthPlace :Helsinki}
    UNION
  {?henkilö dbo:birthPlace :Turku}
  ?henkilö dbo:birthDate ?syntyi .
  ?henkilö dbo:deathDate ?kuoli .
  FILTER (?syntyi < "1917-12-06"^^xsd:dateTime)
}
ORDER BY ASC (?syntyi) # Järjestä varhaisin ensin
```

6.2.2 Tiedon testaaminen: ASK

Joissain datan käyttötilanteissa riittää selvittää, toteuttaako data annetun graafihahmon ilman, että on tarvetta selvittää, millä muuttujien arvoilla hahmo toteutuu. Tällaisia tilanteita varten on käytettävissä SELECT-kyselyä yksinkertaisempi ASK-kysely, jossa ei tarvitse esitellä muuttujia ja joka palauttaa arvonaan totuusarvon “true” tai “false”.

Esimerkiksi se, onko datassa ennen Suomen itsenäistymistä syntynyt Nordsiöld-niminen henkilö, selviää alla olevalla kyselyllä:

```
PREFIX :    <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/> # DBpedian ontologia
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
ASK {
  ?henkilö dbo:birthDate ?syntyi ;
          rdfs:label ?nimi .
  FILTER (?syntyi < "1917-12-06"^^xsd:dateTime) .
  FILTER (regex(str(?nimi),'Nordenskiöld'))
}
```

6.2.3 Resurssin kuvaus: DESCRIBE

Yksi usein vastaantuleva tiedon tarve on selvittää, mitä ominaisuuksia joillain tietyllä resurssilla on. Tähän on käytettävissä DESCRIBE-kysely, jolla asia selviää ysinkertaisemmin kuin SELECT-kyselyllä. Yksinkertaisimmillaan DESCRIBE-kyselyn argumenttina annetaan yksi URI. Esimerkiksi kysely

```
PREFIX k: <http://www.yso.fi/onto/kaunokki#>
DESCRIBE k:ateos_26600
}
```

Kirjasampo-datapalveluun[6] palauttaa Mika Waltarin *Sinuhe egyptiläiseen* liittyvän RDF-kuvauksen eli kolmikot, joissa kuvattava resurssi on joko subjektina tai objektina.

Kaikkien Mika Waltarin kirjoittamien teosten kuvausten muodostama verkko voidaan muodostaa vastaavasti muuttujaa ja graafihahmoa DESCRIBE-kyselyssä käyttämällä:

```
PREFIX k: <http://www.yso.fi/onto/kaunokki#>
DESCRIBE ?teos
WHERE {?teos k:tekija k:person_123175957531148 }
```

[6]Yleisten kirjastojen ylläpitämä Kirjasammmon SPARQL-palvelupiste on osoitteessa http://ldf.fi/kirjasampo/sparql ja on käytössä Kirjasampo.fi-palvelussa (2011). Sen sisältämä RDF-verkko hakee vertaistaan maailmassa semanttisessa rikkaudessaan [36]. Järjestelmä kehitettin alunperin osana FinnONTO-tutkimushanketta, ja sisältää nykyään alkuperäisen kaunokirjallisuuden ohella mm. tietokirjallisuutta.

6.2.4 Uuden RDF-verkon luominen: CONSTRUCT

SPARQL:n yleinen verkonluomiskomento on CONSTRUCT, jonka muoto on:

> CONTSRUCT *graafihahmo-1*
> [FROM *osoite*]
> WHERE *graafihahmo-2*

Siinä sovitetaan ensin *graafihahmo-2* FROM-osoitteessa mainittuun dataan tai jos osoitetta ei ole mainittu, suoraan SPARQL-palvelun dataan. Tuloksena olevat muuttujasijoitukset sijoitetaan sitten *graafihahmo-1*:een yksi toisensa jälkeen, ja kyselyn vastauksena palautetaan näin muodostuneiden graafien unioni. Esimerkiksi Mika Waltarin teosten joukko suomen ja ruotsinkielisine nimikkeineen voidaan muodostaa Kirjasammon datasta näin:

```
PREFIX k: <http://www.yso.fi/onto/kaunokki#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
CONSTRUCT { ?teos rdfs:label ?nimi }
WHERE {
  ?teos k:tekija k:person_123175957531148 .
  ?teos skos:prefLabel ?nimi
}
```

6.3 Laajennuksia peruskyselyihin

SPARQL 1.1 toi käytettäväksi lukuisia uusia hyödyllisiä ominaisuuksia kyselyiden muotoilua varten. Seuraavassa esitellään näistä yhteenvetona keskeisimpiä; kattavat ohjeet kyselyiden muodostamista varten löytyvät kyselykielen dokumentaatiosta W3C-järjestön sivuilta[7].

6.3.1 Ominaisuuspolut

ominaisuuspolku

Ominaisuuspolulla (property path) tarkoitetaan kahden solmun välistä ominaisuuksien ketjua graafissa. Yksinkertaisimmillaan ketjun muo-

[7] https://www.w3.org/TR/sparql11-query/

dostaa yksi ominaisuus, jolloin polun pituus on 1. Useammasta ominaisuudesta muodostuvia ominaisuuspolkuja voidaan kuvata SPARQL-kyselyissä joustavasti *polkulausekkeilla*. Niitä voidaan muodostaa taulukon 6.2 operaattoreiden avulla. *polkulauseke*

Operaattori	**Merkitys**
x/y	ketjut x ja y peräkkäin
$x\|y$	vaihtoehtoiset ketjut x ja y
$\hat{}x$	x:n käänteinen kulkusuunta
$?x$	x on valinnainen ominaisuus
$+x$	x-ominaisuuksien ketju 1...n kpl
$*x$	x-ominaisuuksien ketju 0...n kpl

Taulukko 6.2: Polkulausekkeiden määrittelymuodot

Esimerkiksi seuraavalla kyselyllä voidaan hakea *Helsinki*-yksilön kaikki erilaiset (DISTINCT) luokat (`rdf:type`-kaarta pitkin) ja sitten niiden kaikki yläluokat `?luokka` (`rdfs:subClassOf*` ketjujen kautta) DBpediassa:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://dbpedia.org/resource/>
SELECT DISTINCT *
WHERE {
  :Helsinki rdf:type/rdfs:subClassOf* ?luokka .
}
```

6.3.2 Arvosijoitus

Arvosijoitusten avulla voidaan kyselyssä antaa muuttujille arvoja joko etukäteen tai laskemalla arvoja kyselyn suorituksen aikana. *arvosijoitus*

Muuttujille voidaan kyselyssä asettaa alkuarvoja muodolla

VALUES (*muuttuja-1* ... *muuttuja-N*) *arvosijoitukset*

joka sijoitetaan SELECT-kyselyn sisään ja jossa *arvosijoitukset* annetaan muodossa

{(a_{11} a_{12} ...a_{1n}) (a_{21} a_{22} ...a_{2n}) ...}

Esimerkiksi alla oleva kysely selvittää, mitä valuuttoja Pohjoismaissa käytetään

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT * WHERE {
  ?valtio dbo:currency ?valuutta
}
VALUES (?valtio) {(:Finland) (:Sweden) (:Norway) (:Denmark) (:Iceland)}
```

Arvosijoitus kyselyn aikana voidaan tehdä muodolla

```
BIND (lauseke AS muuttuja)
```

jollaisia voi sijoittaa kyselyhahmossa kolmikoiden paikalle. Esimerkiksi seuraava kysely hakee DBpediasta Helsingissä syntyneet jo kuolleet henkilöt[8] ja luettelee heistä 30 ensimmäistä ikävuosien mukaisessa järjestyksessä vanhimmasta nuorempiin (`DESC`). Päivämäärien vähennyslaskun tuloksena saadaan elinaika sekunteina, joka pitää vielä muuntaa jakolaskulla vuosiksi (oletuksena on, että vuodessa on keskimäärin 365 vuorokautta). `FILTER`-lauseilla varmistetaan, että vain päivämäärämuotoinen data käsitellään, koska DBpediassa henkilöiden syntymä-ja kuolinaikaominaisuuden arvona saattaa olla paitsi päivämäärä (`xsd:date`) myös lisäksi merkkijono. Huomaa myös funtiokutsun `datatype` käyttö tietotyyppien muuntamiseen päivämäärillä tapahtuvaa vähennyslaskua varten.

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?henkilö ?ikä
WHERE {
  ?henkilö dbo:birthPlace :Helsinki .
  ?henkilö dbo:birthDate ?syntyi .
  FILTER ( datatype (?syntyi) = xsd:date )
  ?henkilö dbo:deathDate ?kuoli .
  FILTER ( datatype (?kuoli) = xsd:date )
  BIND ((xsd:date(?kuoli) - xsd:date(?syntyi))/(3600*24*365)
       AS ?ikä)
}
ORDER BY DESC(?ikä)
LIMIT 30
```

BIND-muodolla voidaan luoda uusia muuttujia, antaa niille arvoja, ja myös käyttää laskettuja arvoja kyselyssä, jos tarpeen.

[8] Tätä kirjoitettaessa 758 eri henkilöä

6.3.3 Aggregaattikyselyt

Monissa eri sovelluksissa on tarve käsitellä hakutulosten joukkoa kokonaisuuksina ja palauttaa vastauksena niistä laskettuja arvoja. Esimerkiksi tietynlaisia tuotteita haettaessa saatetaan etsiä tuoteryhmittäin halvinta tai kalleinta tuotetta tai kiinnostuksen kohteena voi olla selvittää keskimääräinen hinta. Samanlaisia tarpeita löytyy vaikkapa säätilaston minimi- ja maksimilämpötiloja kyseltäessä tai maa-alueiden kokoja tutkittaessa. Tällaisia kyselyitä, joissa kyselyn lisäksi analysoidaan vastausjoukkoja ja annetaan vasta analyysin tulokset vastauksena, kutsutaan *aggregaattikyselyiksi* (aggregate query). *aggregaattikysely*

SPARQL-kyselyiden tulosten analysointi voidaan tehdä myös ohjelmallisesti sovelluksessa tulosjoukkoa tutkien, esimerkiksi etsiä minimiarvoa, mutta koska saman tyyppiset kyselytilanteet toistuvat eri sovelluksissa, on aggregaattikyselyitä ryhdytty standardoimaan ja tukemaan niitä jo osana kyselykieltä. Aggregaattikyselyt ovat olleet tärkeä osa relaatiotietokantojen SQL-kyselykielessä ja ne on omaksuttu sieltä myös SPARQL-kieleen.

Aggregaattikyselyssä rajataan ensin ne muuttujat, joiden arvojen muodostaman kombinaatiojoukon (aggregaatin) suhteen analyysi tehdään. Aggregoitavat muuttujat määritellään lisäämällä SELECT-kyselyn loppuun muoto:

```
GROUP BY muuttuja-1 muuttuja-3 ...muuttuja-n
```

Jos muuttujia on vain yksi, generoituu SELECT-kyselyn vastauksesta joukko vastauksia siten, että niistä kussakin on muuttujalla sama arvo. Jos muuttujia on useita, generoidaan yksi vastausjoukko jokaisella muuttujien arvokombinaatiolla. Mikäli GROUP BY -määrittelyä ei ole, käytetään aggregoinnissa oletusarvoisesti vastausjoukkoa sellaisenaan.

Tämän jälkeen SELECT-kyselyn tulosmuuttujina käytettyjen aggregaattimuuttujien arvoja voidaan käsitellä erityisten *aggregaattifunktioiden* avulla ja asettaa niiden arvo uuden tulosmuuttujan arvoksi muodolla *aggregaattifunktio*

```
aggregaattifuntion-kutsu AS uusi-muuttuja
```

Esimerkiksi seuraava kysely palauttaa arvonaan Wikipediasta (DBpedia) varhaisimman helsinkiläisen syntymäajan (tätä kirjoitettaessa päivämäärän 1732-1-11):

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT (MIN(?syntyi) AS ?varhaisin)
WHERE {
  ?henkilö dbo:birthPlace :Helsinki .
  ?henkilö dbo:birthDate ?syntyi
}
```

Koska GROUP BY määrittelyä ei ole, on tulosjoukkojen joukossa tässä tapauksessa vain SELECT-kutsun vastaus. Funktiota `MIN(`x`)` sovelletaan muuttujan `?syntyi` arvoihin vastauksessa, eli tässä tapauksessa yhteen joukkoon. Jos käytettäisiin esimerkiksi `GROUP BY ?syntyi` muotoa, aggregoitaisiin ensin SELECT-kyselyn tulosjoukko (kaikki henkilöt) syntymäajan perusteella eri aikoina syntyneisiin ja sitten jokaisesta joukosta valittaisiin varhaisin aika, joka tietysti olisi kaikille joukon tuloksille sama. Tuloksena olisi silloin eri syntymäajat (mikä ei olisi kovin mielenkiintoista).

Jos halutaan löytää varhaisimman syntymäajan ohella ja myös silloin syntynyt henkilö, tässä ruotsalainen tutkimusmatkailija ja luonnontieteilijä Peter Forsskål, voidaan käyttää esimerkiksi seuraavaa kyselyä, joka perustuu tulosjoukon järjestämiseen:

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?henkilö ?syntyi
WHERE {
  ?henkilö dbo:birthPlace :Helsinki .
  ?henkilö dbo:birthDate ?syntyi
}
ORDER BY ASC(?syntyi)
LIMIT 1
```

Seuraavassa kyselyssä haetaan maittain niiden pääkaupungissa syntyneiden henkilöiden varhaisimmat syntymäajat. Nyt tulosjoukkoina ovat pääkaupungeittain siellä syntyneet henkilöt, joiden syntymäaikojen minimit koostetaan maittain vastaukseen varhaisemmasta myöhäisempään:

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?maa (MIN(?syntyi) AS ?varhaisin)
WHERE {
  ?maa dbo:capital ?pääkaupunki .
  ?henkilö dbo:birthPlace ?pääkaupunki .
  ?henkilö dbo:birthDate ?syntyi .
}
GROUP BY ?maa
ORDER BY ?varhaisin
```

Funktio	**Merkitys**
COUNT(x)	arvojen lukumäärä. COUNT(*) palauttaa vastausten määrän.
SUM(x)	arvojen summa
MIN(x)/MAX(x)	arvojen minimi/maksimi
AVG(x)	arvojen keskiarvo
GROUP_CONCAT(x; separator="s")	arvojen liitos erottimella s eroteltuina
SAMPLE(x)	yksi satunnainen arvo valittuna

Taulukko 6.3: Agregointifunktiot, joita sovelletaan erikseen jokaiseen aggregoituun vastausjoukkoon muuttujan x suhteen.

Taulukoon 6.3 on koottu MIN-funktion ohella käytettävissä olevat aggregaattifunktiot, joita sovelletaan aggregoinnin aikana luotuihin vastausjoukkohin. Vastausjoukkojen generointi kyselyn hahmosta määritellään GROUP BY muodolla; jos muotoa ei ole, oletetaan vastausjoukossa olevan vain yksi joukko, alkuperäinen vastaus. Aggregointifunktioita sovelletaan sitten jokaiseen vastusjoukkoon erikseen, jolloin tuloksista voidaan muodostaa uusi lopullinen vastaus. Esimerkiksi COUNT palauttaa arvonaan vastausjoukon alkioiden lukumäärän ja AVG-funktiolla lasketaan keskiarvo. GROUP_CONCAT-muodolla taas voi näppärästi koostaa muuttujan arvoon vastausjoukon kaikki arvot vaikkapa pilkulla erotettuina. Esimerkiksi seuraava kysely hakee kolme maata, joissa DBpedian mukaan on merkitty puhuttavan eniten eri kieliä, ja luettelee kielet ja niiden lukumäärät maittain englanniksi pilkulla ja välilyönnilä erotettuina listoina kuvan 6.3 mukaisesti. Tämän perusteella ei kuitenkaan voida tietää, missä valtioissa todellisuudessa puhutaan eniten eri kieliä. Esimerkiksi Kiinan osalta kaikkia kieliä ei ehkä ole merkitty Wikipediaan, valinnassa on eri maiden osalta voitu käyttää erilaisia kriteerejä tai

merkinnöissä voi olla virheitä. Myös täysin automaattisessa muunnoksessa Wikipedioista DBpedian RDF-muotoon tapahtuu virheitä. Verkon datalähteiden käyttäminen tutkimuksessa vaatii aina erityisen huolellista lähdekriittistä tarkastelua.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?maa (GROUP_CONCAT(?kielinimi; separator=', ') AS ?kielet)
            (COUNT(?kielinimi) as ?luku)
WHERE {
  ?maa a dbo:Country ;
       dbo:language [ rdfs:label ?kielinimi] .
  FILTER (lang(?kielinimi) = 'en')
}
GROUP BY ?maa
ORDER BY DESC(?luku)
LIMIT 3
```

	maa	kielet	luku
1	http://dbpedia.org/resource/Ukraine	Karaim language, Armenian language, Bulgarian language, German language, Greek language, Hungarian language, Polish language, Romanian language, Russian language, Slovak language, Crimean Tatar language, Krymchak language, Rusyn language, Azerbaijani language, Belarusian language, Romani language, Ukrainian language, Gagauz language, Moldovan language, Yiddish language	"20"^^xsd:integer
2	http://dbpedia.org/resource/East_Timor	Tetum language, Kemak language, Portuguese language, Bekais language, Habun language, Bunak language, Galoli language, Kawaimina languages, Makasae language, Mambai, Uab Meto language, Makalero dialect, Fataluku language, Idalaka language, Makuv'a language, Tokodede, Atauru language	"17"^^xsd:integer
3	http://dbpedia.org/resource/Pakistan	Urdu, Kashmiri language, Burushaski, Balochi language, Gawar-Bati language, Brahui language, Punjabi language, Domaaki language, Balti language, Kalash language, English language, Shina language, Pashto language, Dameli language, Khowar language, Sindhi language	"16"^^xsd:integer

Kuva 6.3: Kolme valtiota, joissa DBpedian mukaan käytetään eniten eri kieliä.

6.3.4 Alikyselyt

alikysely

SPARQL 1.1 mahdollistaa SELECT-kyselyiden käyttämisen osana toista kyselyä. Tällöin graafihahmossa olevan kolmikon paikalle upotetaan *alikysely* kaarisulkujen sisällä. Kyselyt suoritetaan sisältä ulos siten, että ensin suoritetaan sisin kysely. Sen vastauksessa olevat muuttujien ar-

vosijoitukset sijoitetaan tämän jälkeen ulompaan kyselyyn yksi toisensa jälkeen ja vastausten unionista muodostetaan ulomman kyselyn vastaus. Jos tämäkin on alikysely, prosessi jatkuu, kunnes lopulta saadaan laskettua kaikkein uloimman kyselyn vastaus.

6.3.5 Federoidu kysely

Olemme edellä tehneet kyselyjä vain yhteen SPARQL-palvelupisteeseen. Joskus kyselyyn vastaaminen saattaa kuitenkin edellyttää tietojen yhdistämistä useasta eri datalähteestä verkossa. Tietojen yhdisteleminen on mahdollista tekemällä sovelluksesta käsin useita eri kyselyjä eri tietolähteisiin ja yhdistämällä vastaukset ohjelmallisesti. SPARQL-kielessä on kuitenkin myös mahdollista tehdä vastausten yhdistäminen yhden kyselyn sisällä ns. *federoituna kyselynä* (federated query) Federoitu kysely ilmaistaan upottamalla kyselyn graafihahmon kolmikoiden sekaan SERVICE-muoto:

federoitu kysely

```
SERVICE <SPARQL-palvelupiste> kysely
```

Siinä *kysely* voi olla muodon *SPARQL-palvelupisteseen* sovitettava graafihahmo tai kokonainen SELECT-kysely, joka suoritetaan, ja johon vastauksena saadut arvositukset sijoitetaan ulomman kyselyn muuttujille vastaavaan tapaan kuin alikyselyitä tehtäessä. Federoitujen kyselyiden haasteena on usein laskennallinen tehokkuus.

6.4 Graafien hallinta ja päivitys

Edellä on kuvattu SPARQL-kielen kyselytyyppejä. Näitä täydentämään on SPARQL 1.1 standardissa sovittu lisäksi keskeisistä SPARQL-komennoista palvelupisteen RDF-datan muokkaamiseen ja päivittämiseen. Nämä muokkauskomennot voidaan jakaa kahteen ryhmään:

1. *Graafien hallintakomennoilla* voidaan lisätä, poistaa, kopioida ja siirtää kokonaisia graafeja palvelupisteessä. *graafien hallinta*
2. *Graafien päivityskomennoilla* voidaan lisätä ja poistaa kolmikoita yhdessä graafissa. *graafien päivitys*

Keskeisiä graafien hallinnan komentoja ovat:

1. `CREATE`-komennolla voidaan luoda palvelupisteeseen uusia nimettyjä graafeja.
2. `DROP`-komennolla voidaan vastaavasti poistaa kokonaisia graafeja palvelupisteestä.
3. `LOAD`-komennolla voidaan ladata graafiin RDF-muotoista dataa, esimerkiksi tiedosto kerrallaan.
4. `CLEAR`-komennolla voidaan vastaavasti poistaa palvelupisteen graafista sen kolmikot: jäljelle jää vain tyhjä graafi.
5. `COPY`-komennolla luodaan graafista uusi kopio halutulla nimellä.
6. `MOVE`-komento luo graafista COPY-komennon tapaan uuden kopion, mutta poistaa samalla vanhan, eli siirtää graafin uuden nimen alle.

Esimerkiksi komento

```
COPY DEFAULT TO <http://esimerkki.fi/teokset>
```

kopioisi kuvitteellisen SPARQL-palvelupisteen oletusgraafin kolmikot nimettyyn graafiin `teokset`.

Graafien hallinnan komentojen tarkempi kuvailu ja esimerkkejä käytöstä on löytyy W3C:n SPARQL Updata -dokumentaatiosta[9].

6.5 Graafin datan päivittäminen

Olemassa olevan palvelupisteen yksittäisen graafin dataa voidaan muokata kolmikoittain seuraavilla komennoilla:

- `INSERT DATA`-komennolla voidaan lisätä kolmikoita palvelupisteeseen.

[9] https://www.w3.org/TR/sparql11-update/

- `DELETE DATA`-komennolla voidaan vastaavasti poistaa kolmikoita palvelupisteestä.
- `INSERT`-komennolla voidaan lisätä dynaamisesti dataa graafihahmon mukaisesti.
- `DELETE`-komennolla voidaan poistaa dynaamisesti kolmikoita graafihahmon mukaisesti.

Esimerkiksi seuraavalla komennolla voitaisiin lisätä kuvitteellisen SPARQL-palvelupisteen kirjojen metadataa sisältävään nimettyyn graafiin uusi, kolmikkona esitetty hintatieto teokselle `Semanttinen-web`.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://esimerkki.fi/ns#>
INSERT DATA
{ GRAPH <http://esimerkki.fi/teokset>
        { <http://esimerkki.fi/Semanttinen-web> ns:hinta 32 } }
```

INSERT DATA ja DELETE DATA -komennoissa ei ole mahdollista käyttää muuttujia, vain lisätä tai poistaa ekplisiittisesti lueteltuja kolmikoita. INSERT- ja DELETE-komennoilla lisäykset ja muutokset voidaan tehdä muuttujia hyödyntäen gaafihahmon kuvaamille kolmioille, joka on kuvattu WHERE-muodolla vastaavaan tapaan kuin SELECT-kyselyssä. Erillistä kolmikoiden päivityskomentoa ei ole, vaan muutokset tehdään poistamalla ja lisäämällä kolmikoita. Alla olevassa esimerkissä muutetaan RDF-dokumentin kaikkien Masa-nimisten henkilöiden etunimi viralliseen muotoon Matti:

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>

DELETE { ?henkilö foaf:givenName 'Masa' }
INSERT { ?henkilö foaf:givenName 'Matti' }
WHERE { ?henkilö foaf:givenName 'Masa' }
```

Luku 7

Linketyn datan julkaiseminen palveluna

Tässä luvussa esitetään, miten linkitettyä dataa julkaistaan verkossa palveluna.

7.1 Linkitetyn datan neljä periaatetta

Linkitetyn datan julkaisemisessa verkossa pyritään noudattamaan Tim Berners-Leen esittämää neljää *linkitetyn datan periaatetta* (linked data principle).

linkitetyn datan periaatteet

1. Nimeä asiat URI -tunnisteilla.
2. Käytä HTTP-URI-tunnisteita, jotta käyttäjät voivat löytää niihin liittyvän tiedon.
3. Kun joku hakee tietoa HTTP-URI-tunnisteella, tarjoa tietoa standardien avulla (kuten RDF ja SPARQL).
4. Linkitä dataa toisiin URI-tunnisteisiin, niin että käyttäjä voi löytää lisää tietoa.

Keskeinen idea on HTTP-URI-osoitteiden käyttö. Ne toimivat samalla sekä maailmanlaajuisina yksilöivänä tunnisteina että osoitteina, joiden

kautta tunnisteeseen liittyvä tieto on saatavilla. Globaalina visiona on muodostaa Tiedon verkko, Web of Data, jossa tiedot liittyvät maailmanlaajuisesti toisiinsa RDF-ominaisuuksien kautta hieman samaan tapaan kuin perinteisen webin sivut linkittyvät toisiinsa HTTP-linkeillä. Sama idea toimii myös paikallisissa sovellusympäristöissä, joissa yhdistetään ja julkaistaan vaikkapa yhden maan museokokoelmia, kuten MuseoSuomessa.

Jotta verkon WWW-sivujen ja tietojen HTTP-URI-tunnisteet eivät menisi sekaisin, on 1) sovittava käytännöistä erilaisten tunnisteiden esittämisestä ja 2) mekanismista, jolla erilaisiin HTTP-osoitteisiin verkossa vastataan. Jos esimerkiksi kirjoitamme selaimeen URI-osoitteen, miten voidaan tietää, pitääkö vastauksena palauttaa WWW-sivu vai URI-tunnisteeseen liittyvää dataa, ja mitä tuo data silloin olisi?

7.2 HTTP-kutsujen dereferointi

Edellä olemme esitelleet, millaisessa muodossa esimerkiksi Wikipediassa esitetään WWW-sivuja ja DBpediassa niitä vastaavaa dataa. Tähän tarvitaan useita eri osoitteita. Esimerkiksi *Helsinkiin* liittyvät seuraavat osoitteet:

```
http://dbpedia.org/resource/Helsinki # Helsingin käsite
http://dbpedia.org/data/Helsinki     # Helsinkiin liittyvä RDF-data
http://dbpedia.org/page/Helsinki     # Em. datan esittävä WWW-sivu
```

Jos webin *aluetunnus* (domain) ottaa käyttöön WWW-sivuosoitteiden ohella dataosoitteita, kuten yllä, pitää sen palvelimen osata ratkaista, palautetaanko HTTP-kyselyyn jokin verkkosivu vai dataa. Tätä valintaa kutsutaan osoitteiden *dereferoimiseksi* (dereference) eli *ratkomiseksi*.

ratkominen

Osoitteiden analysointi voidaan tehdä edustapalvelimen avulla, joka tutkii saamansa HTTP-kutsut ja reagoi niihin palveluntarjoajan haluamalla tavalla.

Verkko toimii siten, että kun sille lähetetään HTTP-kutsu joltain sovellukselta, esimerkiksi verkoon kytketyn tietokoneen selaimelta, kutsuun sisältyy sekä osoite että *otsikkotieto* (header), jolla voidaan asettaa pyyntöjä kutsun käsittelyä varten. Otsikkotiedolla voidaan mm. kertoa haluttu vastauksen formaatti, kuten HTML, RDF tai JSON. Esimerkiksi

otsikko (HTTP)

seuraavan HTTP-kutsun otsikkotiedot kertovat, että vastauksen pitäisi olla HTML-muotoista tekstiä UTF-8 merkistöllä koodattuna.

```
...
Accept: text/plain
Content-Type: text/html; charset=utf-8
...
```

Osoite otsikkotietoineen lähetetään ensin sille verkon palvelimelle, johon sovelluksen laite on kytketty. Tämän jälkeen ratkaistaan HTTP-kutsun osoitteen ja verkon nimipalvelimien muodostaman verkoston avulla se, mikä verkon palvelin kutsun käsittelee, missä se fyysisesti sijaitsee ja reititetään kutsu tämän palvelimen vastattavaksi. WWW:n nimipalvelun avulla pidetään yllä maailmanlaajuista yksilöivää tietoa siitä, mikä palvelin mistäkin verkkotunnuksesta vastaa. Esimerkiksi Linked Data Finland -palvelun verkkotunnus ldf.fi on rekisteröity suomalaisia fi-tunnuksia ylläpitävän Ficoran nimipalveluun. Tunnuksen haltija, tässä Aalto-yliopisto, voi itsenäisesti ottaa käyttöön alitunnuksia, kuten www.ldf.fi tai data.ldf.fi, ja luoda näiden avulla uusia HTTP-URI-skeeman mukaisia tunnuksia ja päättää siitä, miten vastaaviin HTTP-kutsuihin vastataan.

HTTP-kutsut ohjataan tyypillisesti tutkittavaksi verkkotunnuksen edustapalvelimelle, joka tekee päätökset siitä, miten ja missä erilaiset kutsut käsitellään. Esimerkiksi `http://ldf.fi` -alkuiset HTTP-tunnukset ohjautuvat tätä kirjoitettaessa Aalto-yliopiston SeCo-tutkimusryhmän eräälle edustapalvelimelle, johon asennettu Varnish-ohjelmisto tutkii HTTP-osoitteiden muodon ja niihin liittyvät otsikkotiedot, ja ohjaa kutsut tämän perusteella edelleen toisten sovelluspalvelimien vastattavaksi.

LDF.fi palvelu sisältää kymmeniä erilaisia nimettyjä *datapalveluja* (service). Esimerkiksi Sotasampo.fi-sovellus[1] (WarSampo) perustuu LDF.fi:n palveluun nimeltä *warsa* (WarSampo). Jokainen palvelu voi sisältää joukon erillisiä RDF-graafeja, ja lisäksi käytössä on kaikkien graafien yhdistelmä, oletusgraafi (default graph). Graafien URI:t ovat muotoa:

`http://ldf.fi/`*palvelu*`/`*graafi*

Esimerkiksi *warsa* palveluun sisältyvän n. 160 000 valokuvan metatiedot ovat *photographs*-graafissa

`http://ldf.fi/warsa/photographs`

[1] `http://www.sotasampo.fi`

ja yksittäisten kuvien URI:t on muodostettu tämän aliosoitteina. Alla on esimerkkinä URI

http://ldf.fi/warsa/photographs/sakuva_57717

valokuvalle, jossa sotamarsalkka Mannerheim saapuu seurueineen Mainilaan 17.9.1941.

LDF:fi-palvelun URI:iin liittyvän datan lukeminen tapahtuu muodolla

`http://ldf.fi/`*`palvelu`*`/data?uri=`*`URI`*

esimerkiksi näin:

http://ldf.fi/warsa?uri=http://ldf.fi/warsa/photographs/sakuva_57717

URI:iin liittyvän tiedon lukeminen WWW-sivuna selailua varten onnistuu vastaavasti muodolla

`http://ldf.fi/`*`palvelu`*`/page?uri=`*`URI`*

ja SPARQL-kyselyn kohdistaminen palveluun muodolla

`http://ldf.fi/`*`palvelu`*`/sparql?query=`*`kysely`*

jossa *kysely* on SPARQL-muotoinen kysely. Vastaus palautetaan HTTP-kyselyn otsikkotiedon toiveen mukaisen formaatin mukaan esimerkiksi JSON-muodossa JavaScript-sovellusta varten.

7.3 URI-tunnisteiden sisältöneuvottelu

sisältöneuvottelu

Kun palvelin saa HTTP-kutsun, sen pitää päättää, haluaako asiakas vastauksena HTML-sivun tai dataa. Tämän ratkaisemista kutsutaan *sisältöneuvotteluksi* (content negotiation).

Yleisesti URI-tunnisteiden muodostamisessa on käytössä kaksi eri tapaa[2]:

1. Hash URIt
2. 303 URIt

[2]Ks. tarkemmin "Cool URIs for the Semantic Web": https://www.w3.org/TR/cooluris/

Näiden käsittely sisältöneuvottelussa poikkeaa toisistaan.

Hash URI -nimellä viitataan osoitteisiin, joissa käytetään URI-mallin mukaista, #-merkillä erotettua fragmenttiosaa. Esimerkiksi RDFS-määrittelyn käsitteiden tunnisteet, kuten

```
http://www.w3.org/2000/01/rdf-schema#Class
```

on määritelty hash URI:en avulla. Fragmentin avulla voidaan viitata WWW-sivulla olevaan, HTML-kielen ankkurilla (anchor) merkittyyn kohtaan. Kun asiakasohjelma, esimerkiksi WWW-selain, lähettää HTTP-kutsun, leikkaantuu fragmenttiosa kuitenkin pois, sillä se on tarkoitettu vain asiakasohjelman sisäiseen käyttöön. Jos kysytään esimerkiksi, miten yllä oleva käsite `Class` on määritelty kirjoittamalla em. hash-merkkiä käyttävä osoite selaimeen, saa palvelin ainoastaan tunnisteen alun `http://www.w3.org/2000/01/rdf-schema` tutkittavakseen. Vastauksena voidaan lähettää joko osoitetta vastaava WWW-sivu tai RDF-tiedosto, jossa `Class` ja muut RDFS:n käsitteet on määritelty. W3C.org palvelimen sisältöneuvottelu on ohjelmoitu toimimaan niin, että vastauksena lähetetään RDF-dataa (RDFS-skeeman määrittely) eikä WWW-sivua. Asiakasohjelman murheeksi jää etsiä RDF-datasta juuri `Class`-resurssin määrittely, jos se on tarpeen.

Hash URIen käytön etuna on yksinkertainen ja suoraviivainen toimintamalli, jossa palautetaan kaikki määritelmät sisältävä RDF-tiedosto. Malli on kuitenkin haasteellinen, jos tiedosto on iso. Esimerkiksi DBpediassa hash URIen käyttö ei olisi mielekästä.

303 URI on puolestaan mekanismi, jonka avulla datapalvelussa voidaan palauttaa täsmällisesti vain kysytyn URI:n data. Siinä sisältöneuvottelu tapahtuu kahdessa vaiheessa: Ensin päätellään HTTP-kutsun otsikkotiedon avulla, onko haluaako asiakasjärjestelmä vastaukseksi WWW-sivun vai dataa. Vastaus välitetään asiakasohjelmalla HTTP 303 -vastauksena (status code 303), joka tarkoittaa kyselyn uudelleenohjaamista toiseen osoitteeseen. Vastaus alkuperäiseen kutsuun muuttuu näin joko WWW-sivun tai datan palauttavaksi uudeksi HTTP-kutsuksi. Uudelleenohjaus voidaan tehdä HTTP-kutsun otsikkotiedossa olevan tiedon perusteella, jossa voidaan ilmaista vaatimus vastauksen sisällöstä.

Esimerkiksi LDF.fi-palvelussa aiempaan Mannerhiemin valokuvaan liittyvä URI

http://ldf.fi/warsa?uri=http://ldf.fi/warsa/photographs/sakuva_57717

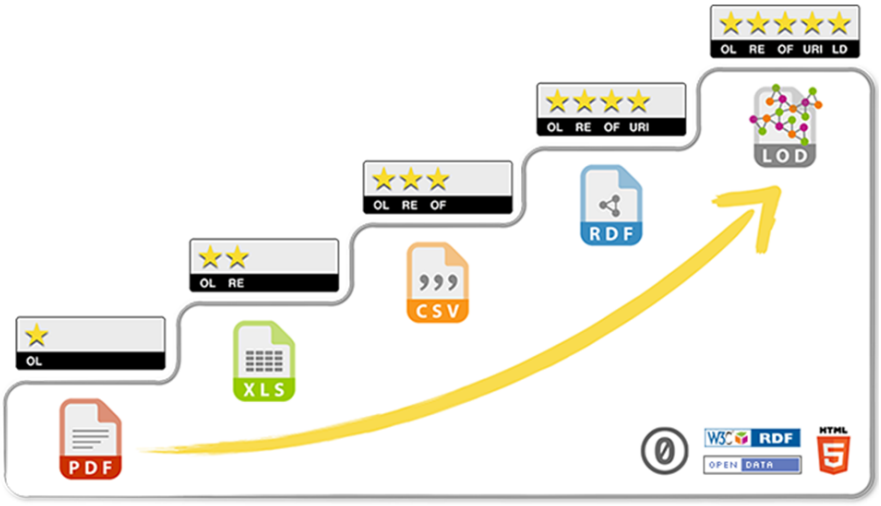
palauttaa RDF-tietoa Turtle-muodossa, jos kutsun otsikossa on tieto

```
Accept: text/turtle
```

mutta URI:n metatiedot HTML-sivuna dataeditorissa esitettynä, jos kutsun osoitetiedoissa lukee `Accept: text/html`. Jälkimmäinen uudelleenohjaus tehdään myös silloin, jos URI kirjoitetaan suoraan selaimeen.

303 URIen käytön etuna on mahdollisuus lukea täsmätietoa yhdestä URIsta kerrallaan, mutta haittana 303 uudelleenohjauksesta johtuva ylimääräinen HTTP-kutsu.

7.4 Datajulkaisujen tähtiluokitus



Kuva 7.1: Tim Berners-Leen lanseeraama avoimen linkitetyn datan viiden tähden luokitus (Lähde: http://5stardata.info/en/)

Avoimen datan palveluiden arviointia varten on Tim Berners-Leen toimesta kehitetty kuvassa 7.1 esitetty tähtiluokitus, jonka tavoitteena on kannustaa julkaisijoita julkaisemaan dataa verkossa mahdollisimman käyttökelpoisessa ja avoimessa muodossa:

⋆
: Ensimmäisen tähden saa, jos ylipäätään avaa dataa verkossa, vaikkapa PDF-domumenttina.

⋆⋆
: Toinen tähti tulee siitä, että data on julkaistu rakenteisessa muodossa, esimerkiksi Excel-taulukkona, niin että datan koneellinen käyttö helpottuu.

⋆⋆⋆
: Kolmannen tähden ideana on kannustaa julkaisemaan dataa yleisiä avoimia standardeja käyttäen, mikä edistää datan käytettävyyttä uudelleen eri järjestelmissä. Esimerkiksi taulukkomuotoinen data on parempi julkaista standardoidussa avoimessa CSV-muodossa kuin Microsoftin Excel-formaatissa.

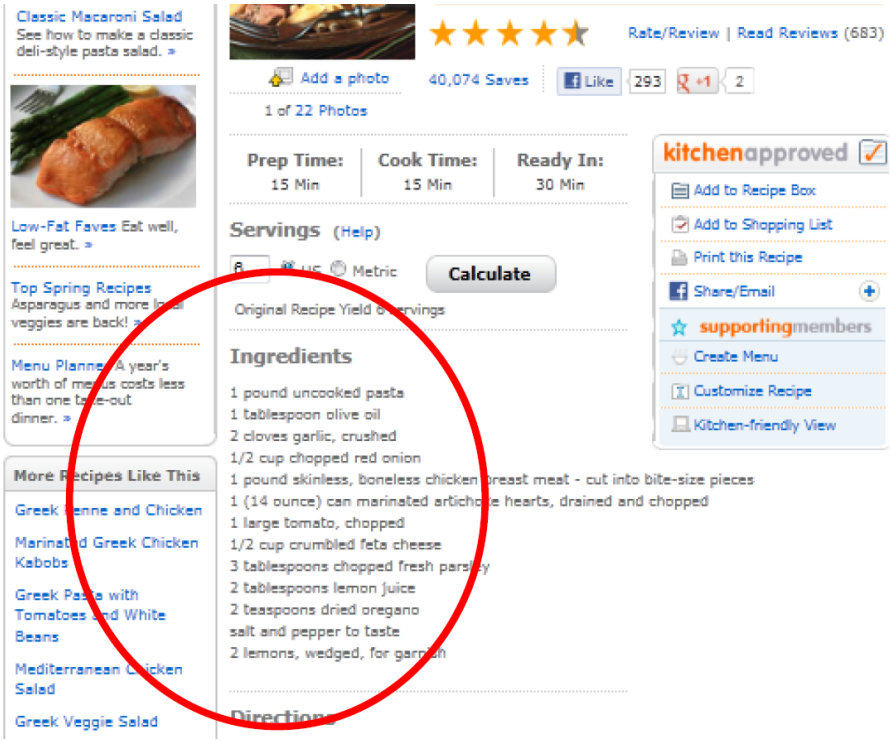
⋆⋆⋆⋆
: Neljännen tähden saaja on jo siirtynyt linkitetyn datan käyttöön ja ottanut käyttöön yksilöivät HTTP-URI-tunnisteet epätäsmällisten nimikkeiden sijaan.

⋆⋆⋆⋆⋆
: Viides tähti myönnetään datajulkaisulle, joka on linkitetty URI-tunnisteilla paitsi sisäsesti, myös muihin ulkopuolisiin datajoukkoihin.

7.5 Datan julkaiseminen WWW-sivuilla

Linkitettyä dataa julkaistaan käytettäväksi toiminnallisten datapalveluiden välityksellä SPARQL-palvelupisteinä. Toinen tapa datan julkaisemiseen ja jakamiseen on upottaa metatietoa verkon WWW-sivuille hakukoneiden, sosiaalisen median ja muiden sovellusten käytettäväksi. Perinteisesti tätä on tehty *mikroformaattinen* (microformat) avulla, jotka ovat *mikroformaatti*

eri sovellusalueille kehitettyjä, yhteisesti sovittuja tapoja esittää tietoa HTML-sivuilla koneiden käytettäväksi.



Kuva 7.2: Reseptisivu verkossa

Yksi esimerkki mikroformaatista on hRecipe, jonka avulla esitetään reseptiihin liittyvää tietoa hakukoneita varten. Kuvassa 7.2 on esimerkki WWW-sivulla olevasta reseptistä, jossa käytetään ympyrällä merkityn listan ainesosia. Listaa vastaavassa HTML-kielisessä kuvauksessa (kuva 7.3) on kerrottu, että listassa on ainesosia (`class="incredients"`) kuten ympyrällä merkittyä sitruunamehua (lemon juice). Mikroformaattien avulla sovellukset kuten hakukoneet ymmärtävät sivujen sisältöjä ja voivat hyödyntää tätä toiminnassaan. Googlen hakukone esimerkiksi tukee hRecipe-formaattia ja tarjoaa mahdollisuuden reseptien täsmähakuun, jossa hakua voi tarkentaa ainesosien avulla. Esimerkiksi kuvassa 7.4 käyttäjä on valinnut hakuun reseptit, joissa käytetään sitruunaa.

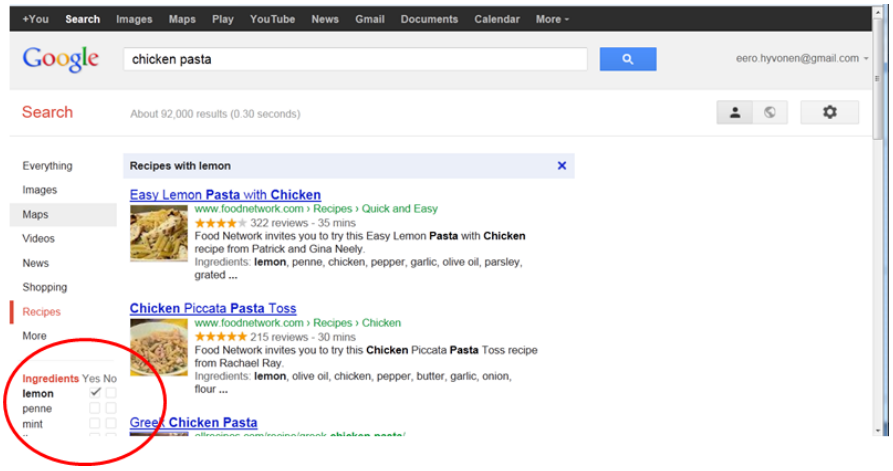
```
<div class="ingredients" style="margin-top: 10px;">
    <h3>
        Ingredients</h3>

            <ul>

            <li class="plaincharacterwrap ingredient">
                1 pound uncooked pasta</li>

            <li class="plaincharacterwrap ingredient">
                1 tablespoon olive oil</li>

            <li class="plaincharacterwrap ingredient">
                2 cloves garlic, crushed</li>

            <li class="plaincharacterwrap ingredient">
                1/2 cup chopped red onion</li>

            <li class="plaincharacterwrap ingredient">
                1 pound skinless, boneless chicken breast meat - cut into bite-size pieces</li>

            <li class="plaincharacterwrap ingredient">
                1 (14 ounce) can marinated artichoke hearts, drained and chopped</li>

            <li class="plaincharacterwrap ingredient">
                1 large tomato, chopped</li>

            <li class="plaincharacterwrap ingredient">
                1/2 cup crumbled feta cheese</li>

            <li class="plaincharacterwrap ingredient">
                3 tablespoons chopped fresh parsley</li>

            <li class="plaincharacterwrap ingredient">
                2 tablespoons lemon juice</li>
```

Kuva 7.3: Merkkaus hRecipe-miroformaatin avulla kuvan 7.2 reseptisivulla

Idea mikroformaateista on otettu käyttöön HTML5-standardissa[3] uudella nimellä ja uusilla kehittyneemmillä esitysmuodoilla *mikrodatana* (microdata)[4]. Semanttisen webin puolella sama ajatus metadatan upottamisesta WWW-sivuille on synnyttänyt RDFa-spesifikaation[5], joka määrittelee tavan upottaa RDF-muotoista dataa verkkosivuille. Verkkosivuille upotettujen merkkausten irroittamiseen HTML/XML-koodista on spesifioitu GRDDL-mekanismi[6]. Yhä suositummaksi metadatan upotusformaatiksi on muodostumassa JSON-LD, jolloin metadatan merkkaus ja HTML-koodi voidaan pitää toisistaan erossa ja metadatan erottaminen ja lukeminen on helppoa JSON-muodossa. Tällöin HTML-tekstin osien ekonominen käyttö samalla kertaa sekä merkkauksessa että sivun ilmia-

mikrodata
RDFa

[3]Suositus julkaistiin vuonna 2014 osoitteessa https://www.w3.org/TR/html5/.
[4]http://www.w3.org/TR/microdata/
[5]http://www.w3.org/TR/xhtml-rdfa-primer/
[6]https://www.w3.org/TR/grddl/



Kuva 7.4: Google tukee reseptien hakua hRecipe-merkkausten avulla

sussa ei kuitenkaan ole mahdollista.

Schema.org

Erityisen merkittäväksi mikroformaatiksi on muodostunut isojen hakukoneyhtiöiden Google, Microsoft, Yahoo ja Yandex vuonna 2011 lanseeraama Schema.org[7]. Sen määrittelyjen avulla voidaan upottaa verkkosivuille, sähköpostiviesteihin ja muihin aineistoihin rakenteisia metadatakuvauksia, jotka perustuvat laajaan ontologiseen luokkahierarkiaan. Sen kymmenet luokat kuten Book, Movie, Event, Person ja Place on määritelty ominaisuuksien avulla vastaavaan tapaan kuin semanttisen webin ontologioissa. Merkkaukset voidaan esittää RDFa, mikrodata tai JSON-LD-muodossa. Hakukoneet hyödyntävät upotettua metatietoa mm. sivujen indesoinnin parantamisessa ja entiteettihaussa, jossa haetaan dokumenttien sijasta kuvauksia henkilöistä, paikoista, elokuvista ja muista entiteeteistä, jotka ovat Schema.org-luokkien yksilöitä. Rakenteisen tiedon avulla voidaan kuvata myös hakutulokset rikkaammassa muodossa ja esittää WWW-sivut ja entiteetit strukturoidusti lisätiedoilla varustettuna.

Metadatamerkkaukset ovat tärkeitä hakukoneiden ohella myös sosiaalisen median sovelluksille kuten Facebook, Twitter ja Instagram, jotka ovat lanseeranneet asiakkaittensa käytettäväksi omia tapojaan sisältöjen merkkaamiseksi, verkkosivujen näyttämiseksi ja näkyvyyden paran-

[7] http://schema.org

tamiseksi omissa sovelluksissaan. Facebookin käyttämä Open Graph -järjestelmä[8] perustuu RDFa-standardiin.

[8] https://developers.facebook.com/docs/sharing/opengraph

Osa III

Tietämyksen esittäminen

Semanttisen webin tietosisällöt ovat konetulkittavia kuvauksia reaalimaailman ilmiöistä, asioista tai tiedoista, ja niitä kutsutaan yleisesti metadataksi. Esimerkiksi kirjaston kokoelman teokset voidaan esittää yksilöllisinä (meta)data-alkiona (resursseina), joita kuvaavia ominaisuuksia ovat teoksen kirjoittaja, julkaisija, julkaisuvuosi, painopaikka jne. Ominaisuuksien arvot voivat olla joko literaalidataa, kuten merkkijonoja, lukuja tai päiväyksiä, tai viittauksia toisiin käsitteisiin.

Tietosisältöjen kuvaukset perustuvat kahdenlaisiin määrittelyihin:

1. *Metadatamalli* (metadata model) kertoo, minkälaisilla ominaisuuksilla erityyppisiä yksilökäsitteitä kuvaillaan. *metadatamalli*
2. *Aiheontologiat* (domain ontology) määrittelevät ominaisuuksien arvoina käytettävät käsitteet, kuten esineet, paikat, henkilöt, tapahtumat ja ajat. *aiheontologia*

Tässä teoksen osassa esitellään tapoja ja kieliä, joiden avulla semanttisessa webissä esitetään metadatamalleja ja niihin liittyviä aiheontologioita.

Luku 8

Metadatan esittäminen

Seuraavassa tarkastellaan ensin metadatan käsitettä ja metadatan erilaisia muotoja. Tämän jälkeen esitellään tarkemmin esimerkkeinä kaksi tärkeää laajemmassa käytössä olevaa yleiskäyttöistä metadatamallia: verkkodokumenttien esittämiseen tarkoitettu Dublin Core ja kulttuurialan sisältöjen harmonisoinnissa käytetty CIDOC CRM -ontologia.

8.1 Metadatan käsite ja muodot

Metadata tarkoittaa kirjaimellisesti "tietoa tiedosta". Termin perustana oleva kreikan sana "meta" viittaa takana olevaan ja "datum" tietoon. American Library Association määritteli metadatan käsitteen vuonna 1999 seuraavasti:

> *Metadata is structured, encoded data that describe characteristics of information-bearing entities to aid in the identification, discovery, assessment, and management of the described entities.*

WWW-alalla metadatalle on kuitenkin annettu huomattavasti laajempi merkityssisältö. W3C-järjestön mukaan metadata on ylipäätään konetutkittavaa tietoa[1]:

[1] http://www.w3.org/Metadata/

Metadata is machine understandable information for the Web.

tietämyksen esittäminen

Metadatassa on tämän mukaan kysymys *tietämyksen esittämisestä* (knowledge representation), joka on yksi keskeinen tekoälytutkimuksen osa-alue[2]. Semanttisessa webissä esitetään tietämystä paitsi kirjastojen kaltaisten kulttuurilaitosten kokoelmista perinteiseen tapaan metadatana myös tietoa taustalla olevasta reaalimaailmasta.

Metadataa on perinteisesti tuotettu erilaisten kokoelmien luetteloinnin yhteydessä ammattilaisten toimesta[3]. Webissä metadataa tuottavat myös harrastajat julkaistessaan vaikkapa ottamiaan valokuvia tai Twitter-viestejä verkossa. Yhä enemmän metadataa tuotetaan nykyisin automaattisesti mm. tiedonlouhinnan keinoin.

Webin sisältökohteisiin liittyvää metadataa voidaan luokitella mm. seuraavalla tavalla[4]:

1. **Hallinnollinen metadata**, jota käytetään kokoelmien hallinnassa: esimerkiksi kohteen hankintaan ja säilytyspaikkaan liittyvä tieto.
2. **Kuvaileva metadata**, jota käytetään kohteen tunnistamiseen ja luonnehtimiseen: esimerkiksi museoesineen tyyppi ja materiaali tai maalauksen aihe ja valmistumisaika.
3. **Säilyttämiseen liittyvä metadata**, joka kuvaa esimerkiksi museoesineen fyysistä kuntoa ja konservointia.
4. **Tekninen metadata**, kuten laitteiston tai ohjelmiston toiminnallinen kuvaus.
5. **Käyttöön liittyvä metadata**, kuten tieto aineistojen käyttöoikeuksista tai verkkopalvelun lokitiedot.

Metadataa voidaan luonnehtia myös sen käytön perusteella:

1. **Indeksoinnin metadata**, jota käytetään hakukoneissa ja suosittelujärjestelmissä sisältöjen hakemiseen ja linkittämiseen.

[2]Tietämyksen esittämiseen voi tutustua tarkemmin esimerkiksi teoksissa [43, 7].
[3]Hyvä johdatus metadatan maailmaan on esimerkiksi [5].
[4]Ks. [15]

2. **Visualisoinnin metadata**, jota käytetään aineistojen esittämiseen ihmiselle havainnollisilla tavoilla.

Metadata esitetään käyttämällä määrämuotoisia rakenteita eli *metadataskeemoja* (metadata schema). Skeemat koostuvat joukosta ominaisuuksia ja näiden arvoja, joiden avulla kuvattava asia esitetään. Esimerkiksi kirjallista teoksista voidaan esittää metatietoina sen kirjoittaja, julkaisija, julkaisuaika, hyllyluokitus ja sisältöä kuvaavia asiasanoja. *metadataskeema*

Metadatamalleja on kehitetty eri tarkoituksia varten:

- **Luettelointiskeemat**, joilla voidaan kuvailla kokoelmissa olevia kohteita museoissa, kirjastoissa tai arkistoissa. Käytössä on lukuisia alakohtaisia malleja, kuten Suomeenkin rantautunut museoalan SPECTRUM[5].

- **Verkkotiedon skeemat**, joilla voidaan kuvata webissä julkaistavia dokumentteja. Näistä tunnetuin on Dublin Core[6].

- **Harmonisointiskeemat**, joilla voi yhtenäistää eri skeemojen avulla luetteloituja tai kuvailtuja tietoja. Tähän kategoriaan voidaan lukea kirjastojen Functional Requirements for Bibliographic Records (FRBR) käsitemalliperhe[7], jota kehittää IFLA (International Federation of Library Associations and Institutions). Museoalalla taas on käytössä ISO standardi CIDOC CRM[8], ontologia, jota kehittää ICOM-järjestön (International Council of Museums) työtyöhmä[9].

- **Harvestointiskeemat**, joita käytetään tiedon esittämiseen ja siirtämiseen, kun tietoa haetaan verkosta eri lähteistä. Paljon käytetty tapa on esimerkiksi Open Archive Initiative Protocol for Metadata Harvesting (OAI-PMH)[10], jossa tietoa kysellään HTTP-protokollalla tietyillä kyselymuodoilla, ja jossa metadata siirretään XML-muodossa.

[5] http://collectionstrust.org.uk/spectrum/
[6] http://dublincore.org/
[7] https://www.ifla.org/node/2016
[8] http://www.cidoc-crm.org/
[9] http://www.cidoc-crm.org/
[10] https://www.openarchives.org/pmh/

Metadataskeemoja voidaan määritellä erilaisilla formaateilla ja malleilla, kuten XML tai UML-kaavio, mutta olennaisesti skeema koostuu joukosta ominaisuuksia ja reunaehtoja näiden arvoille. Tietyn ominaisuuden arvolle määritellään tavallisesti sen tyyppi, esimerkiksi että arvo on aikamäärе, ja syntaktisia vaatimuksia tiedon esittämisellä, esimerkiksi että aikamääreet esitetään ISO 8601 standardin mukaisella tavalla. Lisäksi ominaisuudelle voidaan asettaa *kardinaliteettirajoite* (cardinality constraint) ominaisuuden arvojen lukumäärällä ja ominaisuuden pakollisuudelle tai valinnaisuudelle. Metadataskeeman käyttämisestä voidaan antaa erillisiä *luettelointiohjeita* (cataloging rules)..

kardinaliteettirajoite

luettelointiohje

Tarkastelemme seuraavassa tarkemmin verkkotiedon metadatamalleja esimerkkinä Dublin Core ja tämän jälkeen metatiedon harmonisointia CIDOC CRM -mallin avulla.

8.2 Dublin Core -malli

Dublin[11] Core (DC) on laajasti käytetty metadatastandardi ja -malli, jonka avulla voidaan kuvata monenlaisia sisältökohteita, kuten kirjoja, valokuvia, videoita, web-sivuja ja taide-esineitä.

Standardin ytimessä (core) on 15 standardoitua[12] ominaisuutta eli *elementtiä* (element): DC Metadata Element Set[13] (DCES) (ks. taulukko 8.1).

elementti

Taulukko 8.1: DC Metadata Element Set -määrittelyn 15 ydinominaisuutta

title	creator	subject	description	publisher
contributor	date	type	format	identifier
source	language	relation	coverage	rights

Kaikkien DC-elementtien kardinaliteetti on $0..n$ eli ne ovat valinaisia ja niillä voi olla useita arvoja. Esimerkiksi kirjalla voi olla useita kirjoittajia

[11] Dublin viittaa USA:n Ohiossa olevaan Dublinin kaupunkiin, jossa sijaitsee kirjastoalalla keskeisen vaikuttajaorganisaation OCLC (Online Computer Library Center) pääkonttori.

[12] NISO Standard Z39.85-2001 and ISO Standard 15836-2003

[13] http://dublincore.org/documents/dces/

(*creator*). Elementit esitetään taulukon mukaisessa järjestyksessä *title*, *creator*, *subject* jne.

Keskeinen idea DC-standardissa on *dumb-down-periaate*, jonka avulla Dublin Coren 15 ydinominaisuuden joukkoa voidaan laajentaa yhteentoimivalla tavalla uusilla sovelluskohtaisilla ominaisuuksilla. Nämä ns. *kvalifioidut ominaisuudet* (qualified element) ovat merkitykseltään suppeampia kuin jokin olemassa oleva elementti, mikä voidaan esittää RDF:n aliominaisuutena. Esimerkiksi elementti *manufacturer* olisi merkitykseltää tarkempi kuin ydinominaisuus *creator*. Tämä tarkoittaa, että luojia (*creator*) etsittäessä löytyvät myös valmistajat (*manufacturer*). DC Core -sanastoa voidaan myös laajentaa kokonaan uusilla ominaisuuksilla. Näin määriteltyjä laajennettuja metadatamalleja kutsutaan *DC sovelluksiksi*.

dumb-down-periaate

Tärkeä DC-standardin sovellus on DC-yhteisön itse määrittelemä DCMI (Metadata) Terms. Siitä on muodostunut suosittu RDF-sanasto verkossa olevan metatiedon esittämistä varten. DCMI Terms sisältää taulukossa 8.2 luetellut ominaisuudet, jotka on määritelty nimiavaruudessa:

`http://dublincore.org/documents/dcmi-terms/`

Taulukko 8.2: DCMI Metadata Terms

abstract	accessRights	accrualMethod	accrualPeriodicity	accrualPolicy
alternative	audience	available	bibliographicCitation	conformsTo
contributor	coverage	created	creator	date
dateAccepted	dateCopyrighted	dateSubmitted	description	educationLevel
extent	format	hasFormat	hasPart	hasVersion
instructionalMethod	isFormatOf	isPartOf	isReferencedBy	isReplacedBy
identifier	isRequiredBy	issued	isVersionOf	language
license	mediator	medium	modified	provenance
publisher	references	relation	replaces	requires
rights	rightsHolder	source	spatial	subject
tableOfContents	temporal	title	type	valid

Sovellus sisältää kaikki 15 DC ydinelementtiä ja näiden laajennuksia ja lisäyksiä. Esimerkiksi ominaisuudelle *date* löytyy tarkemmat ominaisuudet *dateAccepted*, *dateCopyrighted* ja *dateSubmitted*. DCMI Terms -spesifikaatioon kuuluu lisäksi *enkoodausmalleja* (encoding scheme), joiden avulla harmonisoidaan ominaisuuksien arvojen esitysmuotoja.

enkoodausmalli

Teknisten spesifikaatioiden ohella DC-yhteisö on julkaissut erilaisia vapaamuotoisempia ohjeita mm. siitä, miten DC-sovelluksia voidaan määritellä.

Tarkastelemme esimerkkinä DC:n soveltamisesta TerveSuomi-järjestelmää, jonka pilottiversio kehitettiin osana kansallista FinnONTO-hanketta.

Järjestelmän taustalla oli havainto siitä, että terveyden edistämiseen liittyviä ohjeita ja aineistoja tuotettiin Suomessa Terveyden ja hyvinvoinnin laitoksen lisäksi mm. yli sadan terveysjärjestön toimesta. TerveSuomen ideana oli luoda maahamme hajautetun tiedon tuotannon malli ja keskitetty semanttinen portaali, johon eri toimijoiden aineistot voitaisiin koota ja rikastaa niitä toistenta avulla sisällöllisesti dataa linkittämällä. Näin voitaisiin samalla poistaa turhaa päällekkäistä sisällöntuotantoa ja parantaa terveysjärjestöjen keskinäistä työnjakoa.[14]

Teknisenä ratkaisuna oli luoda yhteinen metatietomalli, johon eri tiedontuottajien tuottamien verkkosivujen ja -dokumenttien metatiedot voitaisiin muuntaa. Ominaisuuksien arvojen harmonisointia varten luotiin joukko ontologisia sanastoja ja käytänteitä. Metatietomalliksi tuli kuvassa 8.1 esitetty Dublin Core -mallin mukainen sovellus[15]. Nimiavaruus `cd` viittaa Dublin Coreen ja `dct` Dublin Core Terms -määrittelyyn, joita on laajennettu eräillä TerveSuomen nimiavaruudessa `ts` kuvatuilla ominaisuuksilla.

Kuvassa 8.2 on eräs järjestelmään kuulunut Työterveyslaitoksen verkkosivu, jossa opastetaan silmälasien hankinnassa näyttöpäätetyöskentelyä varten. Dokumenttiin liittyvä metadataskeeman mukainen RDF-verkko on esitetty kuvassa 8.3. Siitä nähdään mm. se, että kyseisen dokumentin tyyppi (ominaisuuskaari `rdf:type`) on julkaisu (`ts:Publication`), dokumentin on luonut (`dc:creator`) ja julkaissut (`dc:publisher`) Työterveyslaitos (resurssi `agent:ttl`), ja että dokumentin aiheet (`dc:subject`) "silmälasit", "työolot" ja "päätetyö" on esitetty asiasanaviittauksilla Yleiseen suomalaiseen ontologiaan YSO[16] (nimiavaruus `yso`). Kun järjestelmään kuuluvien esimerkin kaltaisten verkkosivujen ja dokumenttien metatiedot esitettiin samaa metadatamallia hyödyntäen ja ominaisuuksien arvot otettiin yhdessä sovituista ontologioista, voitiin sivujen ja dokumenttien tiedot yhdistää laajemmaksi semanttiseksi verkoksi ja julkaista data SPARQL-palvelupisteessä. Datajulkaisun rajapinnan varaan kehitettiin sitten asiakaskäyttöliittymä, eli semanttinen hakukone ja suosit-

[14] TerveSuomi-järjestelmä (HeathFinland) [46] sai kansainvälisen Semantic Web Challenge teknologiapalkinnon v. 2008 Saksassa, ja siitä tuotteistettiin yksi Terveyden ja hyvinvoinnin laitoksen kansallisista portaaleista.

[15] Malli on kuvattu tarkemmin dokumentissa [47].

[16] YSO-ontologia [42] on kehitetty Kansalliskirjaston Yleisesta suomalaisesta asiasanastosta YSA ja se muodostaa aiemmin esitellyn laajemman yhdistelmä-ontologian KOKO keskeisimmät käsitteet.

telujärjestelmä aineistojen etsimistä ja selailua varten.

8.3 CIDOC CRM: metatietojen harmonisointi

Edellä kuvattu Dublin Core -malli ja siihen sisältyvä dumb-down-periaate tarjoavat menetelmän metatietojen harmonisoimiseksi yhteentoimivaan muotoon ja julkaisemiseksi verkossa. Menetelmä on luonteeltaan dokumentti- ja objektikeskeinen: siinä kuvataan dokumentteja tai objekteja näiden ominaisuuksien avulla. Maailmaan liittyvä tieto ei kuitenkaan ole aina dokumentteja ja maailmassa on muutakin kuin objekteja. Esimerkiksi henkilöiden kuvaamisessa käytetyt elämäkerrat ovat kyllä dokumentteja, mutta niiden taustalla oleva elämä koostuu tapahtumista, joihin liittyy aika, paikka, henkilö itse ja muita toimijoita. Samalla tavalla tietomme historiasta tai kokoelmaesineeseen liittyvä provenienssitieto sille tehdystä konservoinnista, kohteen omistuksista tai näyttelyhistoriasta on luonteeltaan pikemminkin tietoa tapahtumista kuin dokumenteista tai objekteista, jotka toki kuuluvat myös kuvattavaan reaalimaailmaan.

Reaalimaailmaan liittyvän (meta)tiedon esittämistä varten on kehitetty *tapahtumaperustaisia* (event-based) metatietomalleja, joista tunnetuin on museoalan CIDOC CRM -standardi[17]. Sen ideana on tarjota ontologinen tapa sovellusmaailman kuvaamista varten ja muuntaa kaikki metatieto mallin mukaiseen muotoon tapahtumiksi. Esimerkiksi Dublin Core -muotoinen metatieto kirjasta voidaan palauttaa kirjan kirjoittamis- ja julkaisemistapahtumiksi, joiden toimijana on kirjoittaja ja julkaisija tiettyyn aikaan tietyssä paikassa. Jos kirjailijan elämäkerta esitetään syntymä- ja kuolintapahtumien välisten tapahtumien avulla, voidaan tietoa hänestä rikastaa yhteentoimivasti edellä kuvatulla kirjan kirjoittamistapahtumallakin. CIDOC CRM -malli ei ole tarkoitettu dokumenttien luettelointiin tai visualisointiin, vaan se on luonteeltaan datan harmonisointimalli, johon eri aineistojen eri tavoilla esitetty metadata voidaan muuntaa yhteentoimivalla tavalla toisiaan rikastamaan.

tapahtumaperustaisuus

CIDOC CRM -spesifikaatio[18] sisältää 90 luokkaa, jotka on numeroitu *E1*–

[17] Standardin perusteita on esitelty esimerkiksi viitteissä [11, 12] ja hankkeen kotisivulla http://cidoc-crm.org.

[18] Tässä tarkasteltava mallin versio on määritelty viitteessä [10].

E90) sekä 149 ominaisuutta (*P1-P149*), joiden avulla luokkien yksilöitä voidaan yhdistää toisiinsa. Järjestelmän ydinluokkia ja niiden hierarkiaa on esitetty kuvassa 8.4. Myös CIDOC CRM -ominaisuudet muodostavat RDF Schema -malllin mukaisen hierarkian.

Esimerkiksi museossa olevan esineen metatieto voidaan esittää luomalla ensin luokasta *E22 Man-Made Object* yksilö. Yksilön luokka ilmaistaan ominaisuuden *P2 has type of* arvolla. Kohteeseen voidaan sitten liittää tunnistetieto ja nimiä eli *apellaatioita* (appellation) (*E41 Appellation*),

apellaatio

tietoa kohteen hankinnasta ja omistuksesta, paikkatietoa, tietoa kohteeseen liittyvistä historiallisista tapahtumista, kohteen kuvailutietoa ja tietoa kohteen fyysisistä mitoista (esimerkiksi *P43 has dimension*). Ominaisuuksien arvot ovat tyypillisesti viittauksia toisiin resursseihin. Esimerkiksi ominaisuuden *P52 has current owner* arvot ovat yksilöitä luokasta (*E39 Actor*).

Sisältöjä kuvaillaan tyypillisesti tapahtumaluokan (*E5 Event*) ja sen alaluokkien yksilöinä, kuten *E10 Transfer of Custody*, *E11 Modification*, *E65 Creation* ja *E87 Curation Activity*. Näitä luonnehditaan aikamääreillä (esimerkiksi *P10 falls within*), paikkatiedolla (esimerkiksi *P7 took place at*) ja toimijoilla, jotka liittyvät tapahtumiin erilaisilla rooleilla (esimerkiksi *P11 had participant*).

CIDOC CRM -järjestelmää kehittää aktiivinen yhteisö kansainvälisen ICOM CIDOC -organisaation alaisuudessa (International Committee for Documentation[19]). Palaamme myöhemmin mallin soveltamiseen käytännössä kirjan viimeisessä osassa.

[19] http://icom.museum/the-committees/international-committees/international-committee/international-committee-for-documentation/

	Nimi	Tunniste	P/T	Arvotyyppi	Arvojoukko
perustiedot	**Identifiointitunnus**	**dc:identifier**	1	URI	
	URL-osoite	ts:url	0..1	URL	
	Nimeke	**dc:title**	1[a]	Vapaateksti	Ei-tyhjät merkkijonot
	Kuvaus	**dc:description**	1[a]	Vapaateksti	Ei-tyhjät merkkijonot
	Kieli	**dc:language**	1	Rakenteinen merkkijono	RFC4646
aikamääreet	**Julkistamisaika**	**dcterms:issued**	1	Rakenteinen merkkijono	W3CDTF (ISO 8601)
	Hyväksymisaika	dcterms:dateAccepted	0..1	Rakenteinen merkkijono	W3CDTF (ISO 8601)
	Muokkausaika	dcterms:modified	0..1	Rakenteinen merkkijono	W3CDTF (ISO 8601)
tekijät	**Julkaisija**	**dc:publisher**	1..*	Yksilö	foaf:Organization
	Tekijä	dc:creator	0..*	Yksilö	foaf:Organization, foaf:Person tai foaf:Group
	Kuuluu kokoelmaan	dcterms:isPartOf	0..1	Yksilö	ts:PublicationCollection
	Oikeudet	dc:rights	0..*	Vapaateksti ja/tai dokumentti	URI tai sanallinen kuvaus
	Tallennuskielto	ts:noindex	0..1	Boolean	"true", "false", "1" tai "0"
luokittelut	**Julkaisun laji**	**ts:genre**	1	Käsite	Julkaisun laji -käsitteet
	Esitysmuoto	**dc:type**	1[b]	Käsite	DCMI Type -käsitteet
	Formaatti	**dc:format**	1	Rakenteinen merkkijono	IANA-rekisteröidyt MIME-tyypit
sisällönkuvailu	**Aihe**	**dc:subject**	1..*	Käsite	TERO-, YSO-, MeSH-, TESA- ja STAMETA-käsitteet
	Avainsana	ts:keyword	0..*	Vapaateksti	Ei-tyhjät merkkijonot
	Kohdeyleisö	**dcterms:audience**	1	Käsite	TerveSuomi-portaalin osioluokitus

[a] Erikieliset versiot sallittu, kunhan kullakin kielellä on annettu vain yksi arvo.
[b] Pakollinen, jos arvo voi olla muuta kuin Teksti.

Kuva 8.1: TerveSuomi-järjestelmän metadataskeema (kuvattu tarkemmin raportissa [47]). Pakolliset kentät on lihavoitu. Sallittujen arvojen määrä eli pakollisuus ja toisteisuus on ilmaistu P/T-sarakkeessa UML-merkintätavan mukaisesti. Esimerkiksi 0..* tarkoittaa valinnaista kenttää, joka voi saada useita arvoja ja 1 tarkoittaa, että kenttä on pakollinen, mutta saa vain yhden arvon.

TTL - Maijan uudet näyttöpäätelasit http://www.ttl.fi/ergonomia

Maijan uudet näyttöpäätelasit

Sopivien silmälasien hankinta tietotyöhön voi useinkin olla pulmallista.
Seuraavassa näet, miten ikänäköinen Maija päätyi silmälasiratkaisuunsa

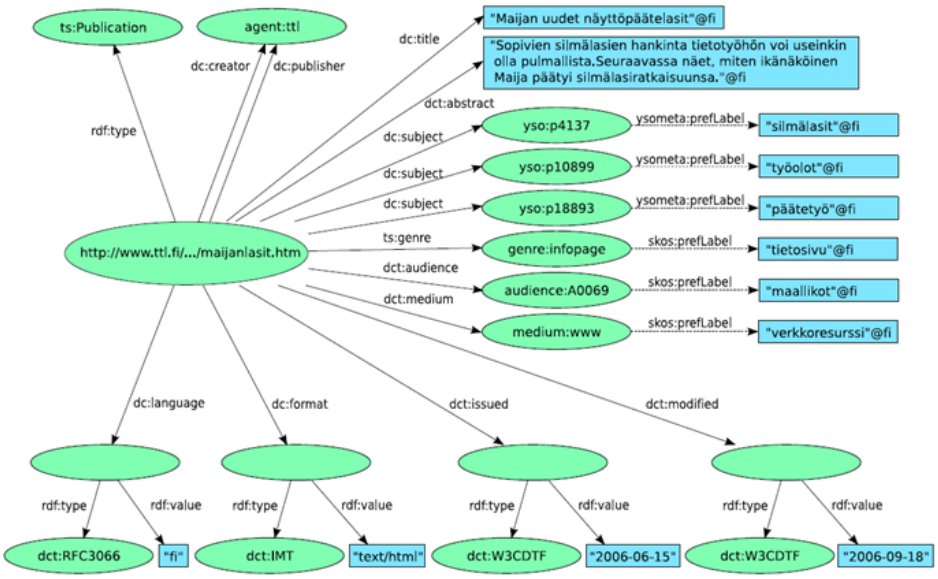
Maija sai lähinäön huonontuessa aluksi lukulasit. Hän hankki kymmenen vuoden aikana kolmet lukulasit: uudet aina hiukan entistä vahvemmat.

Päätteeltä lukeminen alkoi käydä ongelmalliseksi. Maija ei osannut kirjoittaa sokkona joten näppäimistölle näkeminen oli myös välttämätöntä. Niska ja hartiat jomottivat, silmiä kirveli ja vasemmassa silmässä oli elohiiri viikkokausia.

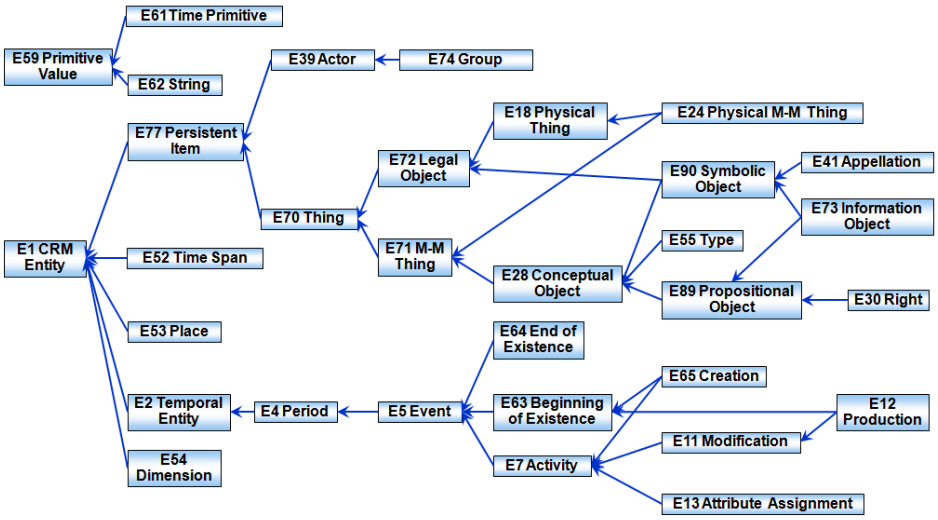
Lukulasit käytössä.
Maija joutuu istumaan epämiellyttävän lähellä päätettä. Kyynärpäiden voimakas koukistaminen sitoo käsiä ja vaikeuttaa näppäilytyötä.

Lukulasit käytössä.
Maija joutuu taivuttamaan niskaa voimakkaasti eteen nähdäkseen näppäimet. Päätteen ja näppäimistön vuorottainen katselu aiheuttaa jatkuvaa niskan liikettä.

Kuva 8.2: Työterveyslaitoksen julkaisema verkkodokumentti TerveSuomi-järjestelmässä.



Kuva 8.3: Kuvan 8.2 metatiedot RDF-verkkona kuvan 8.1 metatietomallin mukaisesti kuvattuna.



Kuva 8.4: CIDOC CRM -mallin ydinluokkien hierarkia. Kaaret ovat RDF-mallin mukaisia *subclass-of* suhteita. (Lähde: http://www.cidoc-crm.org/).

Luku 9

Ontologian käsite

Ontologialla (ontology) tarkoitetaan[1] tietotekniikassa *formaalia, yhteisesti sovittua käsitteellistä kuvausta maailmasta*: *ontologia*

- *Formaali* tarkoittaa, että kuvaus on yksikäsitteisesti tulkittavissa tietokoneella.
- *Yhteisesti sovittu* merkitsee, että samaa kuvausta käytetään jonkin yhteisön piirissä yleisemmin, mikä mahdollistaa yhteisön eri tahojen tuottamisen tietojen yhteentoimivuutta.
- *Käsitteellinen kuvaus* maailmasta tarkoittaa sitä, että ontologian esittämä malli vastaa jossain mielessä kuvaamaansa ilmiötä tai asiaa.

Tämä tekninen määritelmä ontologiasta[2] poikkeaa huomattavasti termin perinteisestä merkitystä tieteessä, jossa ontologia on olevaisen perimmäistä olemusta tutkiva filosofian osa-alue.

Ontologia määrittelee *sovellusalan* (domain) kuvaamisessa tarvittavat käsitteet, kuten *pöytä*, *Helsinki* tai *Jeesus* käsitteiden verkkona toistensa avulla. Ideana on, että ontologiassa määriteltyjen käsitteiden avulla kuvattua tietoa voidaan tulkita tietokoneen avulla ja tehdä tiedosta päätelmiä eli rikastaa tietoa automaattisesti. Jos esimerkiksi tiedetään, että *sovellusala*

[1] Semanttisen webin ontologian käsitettä on määritelty mm. artikkelissa [16].

[2] Ontologioita semanttisen webin kontekstissa on käsitelty laajasti teoksessa [44].

Mikki on eräs hiiri ja että hiiret ovat jyrsijöitä, voidaan päätellä Mikin olevan jyrsijä.

Ontologia-termiä käytetään semanttisessa webissä monessa eri merkityksessä: sillä voidaan tarkoittaa sekä käsitteitä määritteleviä aiheontologioita että myös metadatamalleja. Rajanveto ei ole aina selvää, sillä semanttisessa webissä sekä metadatamallit että käsiteontologiat esitetään yhtenäisesti RDF-perustaisena semanttisena verkkona päinvastoin kuin esimerkiksi tietokantajärjestelmissä, joissa tietokannan tietomalli (skeema) ja data on erotettu toisistaan.

tietämyksen esittämisjärjestelmä

Aiheontologia kuvataan tyypillisesti käyttämällä hyväksi jotain *tietämyksen esittämisjärjestelmää* (knowledge organization system, KOS). Tärkeimmät semanttisen webin KOS-standardit ovat

- RDF Schema (RDFS),
- Simple Knowledge Organization System (SKOS) ja näitä monipuolisempi ja mutkikkaampi
- Web Ontology Language (OWL),

jotka kaikki perustuvat RDF-verkkoihin.

Linkitetyssä datassa pitäydytään usein yksinkertaisissa RDFS- ja SKOS-verkoissa. Seuraavassa esitellään ensin näitä kahta yksinkertaista KOS-standardia ja sitten huomattavasti monipuolisemman OWL-kielen mahdollisuuksia. Kaikki esiteltävät järjestelmät ja kielet ovat semantiikaltaan yleiskäyttöisiä, ts. sovellusalasta rippumattomia. Ajatuksena on, että niiden avulla voidaan kuvata varsinaiset alakohtaiset ontologiat, esimerkiksi biologian taksonomiat, kirjaston asiasanastot, henkilörekisterin tiedot tai paikkatietojärjestelmän nimistö. Yhteisen semanttisen järjestelmän ja kielen käyttö mahdollistaa heterogeenisten tietojen esittämisen yhteentoimivalla tavalla ja tietojen yhdistämisen merkitysten tasolla.

Luku 10

RDF Schema

10.1 Luokat ja yksilöt

RDFS tuo semanttiseen webiin *luokan* (class) käsitteen ja idean *päättelystä* (inference), jonka avulla RDF-verkkoa voidaan rikastaa automaattisesti uusilla kolmikoilla. Luokalla tarkoitetaan yksinkertaisesti joukkoa *yksilöitä* (instance, individual), jotka kuuluvat luokkaan ja joilla on sen ominaisuudet. Esimerkiksi Maalaukset-luokkaan kuuluvat yksittäiset maalaukset, kuten *Taistelevat metsot* tai *Aino-triptyykki*. Yksilön ja sen luokan suhde ilmastaan ominaisuudella `rdf:type`:

luokka

yksilö

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

:Taistelevat_metsot rdf:type :Maalaus .
:Aino-triptyykki rdf:type :Maalaus .
```

Tapana on, että luokkien URIt kirjoitetaan yleensä isolla alkukirjaimella ja yksilöiden pienellä, ellei kyse ole yksilön nimestä, kuten edellä.

Resurssi on luokka, jos se on RDFS-sanastoon kuuluvan luokan `rdfs:Class` yksilö:

```
:Maalaus rdf:type rdfs:Class .
```

Resurssi `:Maalaus` on siis samaan aikaan sekä yksilö (luokasta `rdfs:Class`) että luokka (jolla on omia yksilöitä). Koska `rdfs:Class` on myös luokka, se on myös itsensä yksilö:

```
rdfs:Class rdf:type rdfs:Class .
```

10.2 Luokkien hierarkia

luokkahierarkia

Luokista voidaan rakentaa *luokkahierarkioita* (class hierarchy) `rdfs:subClassOf`-ominaisuuden avulla; RDF Schema on yksinkertainen KOS-järjestelmä hierarkkisten luokkaontologioiden määrittelemiseksi. Sen avulla voidaan esimerkiksi määritellä, että `Metso` on `Lintu`-luokan alaluokka, joka puolestaan on `Eläin`-luokan aliluokka.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

:Metso rdfs:subClassOf :Lintu .
:Lintu rdfs:subClassOf :Eläin .
```

Luokkahierarkioita käytetään paljon maailman ilmiöiden jäsentämiseen ja mallintamiseen. RDFS:n kannalta keskeinen käyttötapaus on, että luokkahierarkia mahdollistaa yksinkertaisen hierarkkiseen yksilöluokkasuhteen päättelyyn: yksilön voidaan aina päätellä olevan kaikkien yläluokkiensa yksilö, vaikkei tätä tietoa RDF-graafin olisi erikseen kolmikoilla esitetty kaikkien yläluokkien osalta. Jos esimerkiksi tiedetään, että maalauksessa on kuvattu metso, voidaan hierarkkisen ontologian avulla päätellä maalauksen esittävän myös lintua ja eläintä. Näin esimerkiksi tiedonhaussa eläinaiheisia maalauksia haettaessa voidaan löytää myös metsoa esittävät ja yleisemmin muutkin lintu- ja eläinaiheiset maalaukset. Luetteloinnin kannalta riittää kertoa, että maalauksen yhtenä aiheena on metso, sillä aiheen kuuluminen myös tämän yläluokkiin voidaan päätellä automaattisesti.

10.3 Ominaisuuksien hierarkia

`rdfs:Class`-luokan ohella toinen keskeinen RDFS:n luokka on ominaisuuksien luokka `rdf:Property`, joka on määritelty RDF:n nimiavaruudessa. Kaikki RDF:n ominaisuudet ovat tämän luokan yksilöitä. Esimerkiksi:

```
dc:creator rdf:type rdf:Property .
```

RDF:n ja RDFS:n erottaminen toisistaan johtuu osin historiallisista syistä. Nimiavaruudet muodostavat yhdessä kokonaisuuden, josta käytetään lyhennettä RDF(S).

RDFS sisältää luokkahierarkioiden ohella myös vastaavanlaisen mahdollisuuden määritellä *ominaisuushierarkioita* (property hierarchy) `rdfs:subPropertyOf`-suhteella. Voidaan esimerkiksi määritellä, että sekä teoksen kirjoittaja että maalauksen maalaaja ovat *aliominaisuuksia* (subproperty) ominaisuudelle `:teoksen_luoja`:

ominaisuushierarkia

aliominaisuus

```
:teoksen_kirjoittaja rdfs:subPropertyOf :teoksen_luoja .
:maalauksen_maalaaja rdfs:subPropertyOf :teoksen_luoja .
```

Jos tiedonhaussa etsitään henkilöitä, jotka ovat luoneet teoksia, ja tiedetään Väinö Linnan kirjoittaneen teoksen ja Ferdinand von Wrightin maalanneen maalauksen, löytyvät molemmat, koska aliominaisuuden olemassaolosta voidaan päätellä sen kaikkien yläominaisuuksien olemassaolo samojen resurssien välillä.

10.4 Alue- ja arvorajoitteet

Luokkien ja ominaisuuksien hierarkioiden ohella kolmas keskeinen RDFS-sanaston tuoma uutuus on mahdollisuus määritellä ominaisuuksien käytölle *rajoitteita* (constraint):

rajoite

- *Aluerajoitteen* (domain constraint) avulla voidaan vaatia, että ominaisuus voidaan antaa vain tietyn luokan yksilöille. *aluerajoite*
- *Arvorajoitteen* (range constraint) avulla voidaan vaatia, että ominaisuuden arvona on tietyn luokan yksilö. *arvorajoite*

Esimerkiksi se, että henkilöt kirjoittavat romaaneja, mutteivat esimerkiksi lintuja, voidaan ilmaista näin:

```
Henkilö rdf:type rdfs:Class .
Romaani rdf:type rdfs:Class .

:kirjoitti rdfs:type rdf:Property ;
  rdfs:domain :Henkilö;
  rdfs:range :Romaani .
```

Rajoitemekanismia käytetään luokka- ja ominaisuushierarkioiden tapaan uuden tiedon päättelyyn. Esimerkiksi tiedosta

```
:h13 kirjoitti :r43 .
```

voidaan päätellä, että :h13 on Henkilö-luokan ja :r43 Romaani-luokan yksilö:

```
:h13 rdf:type :Henkilö .
:r43 rdf:type :Romaani .
```

Ominaisuudelle määritellyt rajoitteet periytyvät ominaisuuden mahdollisille aliominaisuuksille siten, että aliominaisuuden alueen ja arvon luokka voi olla vain sama tai rajoittuneempi kuin yläominaisuuden.

Tarkemmat RDF- ja RDF Schema -määrittelyt ja niihin liittyviä muita spesifikaatioita löytyvät kootusti W3C:n kotisivuilta[1].

[1] https://www.w3.org/standards/techs/rdf

Luku 11

SKOS – Simple Knowledge Organization System

Museoissa, kirjastoissa, arkistoissa ja mediataloissa on jo pitkään käytetty erilaisia luokituksia ja asiasanastoja kokoelmakohteiden sisällönkuvailussa. Esimerkiksi Suomessa monet tieteelliset erikoiskirjastot käyttävät teoksissaan UDK-luokitusta, jonka mukaan teokset asetetaan hyllyihin. Yleisissä kirjastoissa taas on käytössä luokitusjärjestelmänä YKL[1] ja Helsingissä HKLJ[2]. Sisällön tarkempaan kuvailuun on käytetty mm. Kansalliskirjaston Yleistä Suomalaista Asiasanastoa (YSA) ja muita eri alojen asiasanastoja. Perinteisiin luokitusjärjestelmiin ja asiasanastoihin sisältyy käsitteitä ja niiden välisiä suhteita ja niitä voidaan pitää eräänlaisina yksinkertaisina ontologioina. Tällaisten käsiterakenteiden kuvaaminen sellaisenaan semanttisena rakenteena on usein hyödyllistä sovellusten kannalta.

Jotta eri aloilla kehitetyt laajat luokitukset ja sanastot voitaisiin hyödyntää mahdollisimman vähällä vaivalla semanttisessa webissä, on niiden kuvaamista varten kehitetty oma standardinsa, Simple Knowledge Organization System SKOS[3]. Se tarjoaa valmiit RDF-perustaiset ratkaisut, joilla useimmat yksinkertaiset luokitukset ja sanastot voidaan esittää yhteismitallisessa semanttisessa muodossa. SKOS-sanasto on määritelty alla olevassa nimiavaruudessa, josta käytetään jatkossa nimeä `skos`:

[1] http://ykl.kirjastot.fi/
[2] http://hklj.kirjastot.fi/
[3] http://www.w3.org/TR/skos-reference/

http://www.w3.org/2004/02/skos/core#

11.1 Käsitemalli

Ominaisuus	Merkitys
skos:inScheme	Ilmaisee käsiteyksilön käsitemallin.
skos:hasTopConcept	Viittaa käsitemallista sen juurikäsitteisiin.
skos:topConceptOf	Kertoo juurikäsitteen käsitemallin.

Taulukko 11.1: Käsitemallin ja sen käsitteiden väliset viittaukset SKOS-suosituksessa.

käsitemalli
käsite

SKOS esittää tietämystä *käsitemallina* (concept scheme), joka koostuu joukosta *käsitteitä* (concept). Käsitemalli määritellään luokan `skos:ConceptScheme` yksilönä ja yksittäiset käsitteet luodaan luokasta `skos:Concept`. Käsitemalliin kuuluu yksi tai useampia *pääkäsitteitä*

pääkäsite

(top concept), jotka liitetään toisiinsa taulukon 11.1 kolmen eri ominaisuuden avulla. Pääkäsitteille voidaan sitten määritellä merkitykseltään *suppeampia* (narrower) alikäsitteitä, jotka muodostavat asteittain tarkentuvia käsitteiden hierarkioita.

11.2 Käsitteen nimikkeet

Käsitteet yksilöidään URI-tunnisteilla ja niihin voidaan liittää erityyppisiä ja erikielisiä nimikkeitä (label) ihmiskäyttäjiä varten. Erottamalla käsitteen nimikkeet niiden tunnisteista on mahdollista luoda monikielisiä käsitemalleja.

Käsitteellä voi olla kolmenlaisia nimikkeitä (ks. taulukko 11.2):

suositeltu nimike

1. Jokaisen kielen osalta yksi nimike on valittu *suositelluksi nimikkeeksi* (preferred label).

vaihtoehtoinen nimike

2. Suositellun nimikkeen ohella käsitteellä voi olla lisäksi *vaihtoehtoisia nimikkeitä* (alternative label).

3. Nimikkeet voivat olla myös *piilonimikkeitä* (hidden label), jolloin niitä käytetään vain sisäisesti tiedonhaussa synonyymeinä, muttei tietoa kuvailtaessa. *piilonimike*

Ominaisuus	**Merkitys**
skos:prefLabel	Käsitteen suositeltu nimike, joita on yksi jokaisella eri kielellä. Sekaannusten välttämiseksi nimikkeet kannattaa määritellä yksilöivästi kunkin kielen piirissä niin, ettei kahdella eri käsitteellä ole samaa suositeltua nimikettä, vaikka se periaatteessa on mahdollista.
skos:prefLabel	Käsitteen vaihtoehtoinen nimike, joita voi olla useita.
skos:hiddenLabel	Käsitteen vaihtoehtoinen nimike, jota voidaan käyttää sisäisesti esimerkiksi haun saantia parantamaan, mutta jota ei toivota käytettävän tiedon kuvailussa.

Taulukko 11.2: Ominaisuudet SKOS-käsitteen eri nimikkeiden ilmaisemiseksi.

Alla on esimerkkinä määritelty biologinen eläinsanasto, jolla on kaksi juurikäsitettä `linnut` ja `kalat`, joista linnut-käsitteellä on suppeampi käsite lokit.

```
@prefix : <http://www.esimerkki.fi/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix dct: <http://purl.org/dc/terms/> .

<elainsanasto> rdf:type skos:ConceptScheme ;
  dct:title "Eläinsanasto"@fi ;
  skos:hasTopConcept <linnut>, <kalat> .

<linnut> rdf:type skos:Concept ;
  skos:prefLabel "linnut"@fi ;
  skos:altLabel "lintu"@fi ;
  skos:hiddenLabel "tipu"@fi ;
  skos:topConceptOf <elainsanasto>

<kalat> rdf:type skos:Concept ;
  skos:prefLabel "kalat"@fi ;
  skos:altLabel "kala"@fi ;
  skos:topConceptOf <elainsanasto>

<lokit> rdf:type skos:Concept ;
  skos:prefLabel "lokit"@fi ;
  skos:altLabel "lokki"@fi ;
  skos:inScheme <elainsanasto> .
```

Esimerkistä näkyy, että SKOS-käsitteet kuten `linnut` ovat yksilöitä luokasta `skos:Concept`. Jos halutaan esittää tietoa yksittäisestä linnusta, pitäisi määritellä luokka `Linnut` ja luoda yksilö siitä RDF(S):n avulla:

```
:Linnut a rdfs:Class .
:lintu23 rdf:type :Linnut .
```

SKOS-järjestelmän ideana on siis mallintaa käsitejärjestelmän, ei niinkään reaalimaailman yksilöitä, mihin käytetään yleensä RDF(S)-sanastoa.

SKOS:n nimikkeet ovat literaaleja. Tämä merkitsee sitä, ettei nimikkeillä voi olla ominaisuuksia (paitsi kielimääre, joka ei ole RDF-ominaisuus). Monasti tämä olisi kuitenkin tarpeellista. Esimerkiksi henkilön nimi voi naimisiin menon yhteydessä muuttua tai samalla paikalla voi olla eri aikoina eri nimi, jolloin nimike pitäisi voida *kvalifioida* (qualify) ajan suhteen ja kertoa sen voimassaoloaika.

kvalifioida

Ongelma on ratkaistu SKOS-standardin laajennuksessa *SKOS eXtension for Labels* (SKOS-XL), joka on määritelty nimiavaruudessa

```
@prefix skosxl: http://www.w3.org/2008/05/skos-xl# .
```

Siinä nimikkeet luodaan yksilöinä luokasta `skosxl:Label`, jotka liitetään sen emokäsitteen ominaisuuden arvoksi `skosxl:prefLabel`,

`skosxl:altLabel` ja `skosxl:hiddenLabel` ominaisuuksilla. Nimikeluokan yksilölle voidaan sitten antaa ominaisuudella `skosxl:literalForm` ninimikkeen literaaliarvo, ja samaan tapaan voidaan nimikkeisiin liittää muutakin tietoa, esimerkiksi nimikkeen voimassaoloaika. Nimikkeitä voidaan myös linkittää toisiinsa `skosxl:labelRelation`-ominaisuuden aliominaisuuksilla, joita voi määritellä tarpeen mukaan.

11.3 Notaatiot

Luokituksissa ja sanastoissa käytetään usein erityisiä tunnisteita eli *notaatioita* niissä esiintyvien termien yksilöintiin. Nimikkeistä poiketen notaatiot eivät yleensä ole luonnollista kieltä vaan erilaisia koodeja. Esimerkiksi Helsingin kaupunginkirjaston HKLJ-luokituksessa on kategoria *notaatio*

`Atk. Tietotekniikka. Tietoliikenne. Viestintätekniikka`,

jonka notaatio on `627.7`. Notaatioita käytetään tiedon indeksoinnissa.

SKOS-suosituksessa notaatiot esitetään ominaisuudella `skos:notation`, jonka arvona on merkkijono. Notaatiolle voidaan lisäksi antaa sen tyyppiä tarkentava tietotyyppi, sillä yhdellä käsitteellä voi olla useita eri notaatioita eri sanastoissa. Esimerkiksi:

```
<kasite45> skos:notation "627.7"^^<HKLJ-notaatio>
```

11.4 Semanttiset relaatiot

Ominaisuus	**Merkitys**
`skos:broader`	Viittaus tarkemmasta käsitteestä yleisempään.
`skos:narrower`	Viittaus yleisemmästä käsitteestä tarkempaan.
`skos:related`	Muu assosiatiivinen viittaus käsitteestä siihen liittyvään toiseen käsitteeseen.

Taulukko 11.3: SKOS-käsitemalliin kuuluvien käsitteiden väliset semanttiset relaatiot.

SKOS-käsitteiden merkityksen määrittely tietokoneille perustuu joukolle yksinkertaisia *semanttisia relaatioita*, jotka yhdistävät käsitteet laa- *semanttinen relaatio*

hierarkkinen relaatio
laajempi käsite
suppeampi käsite

jemmaksi käsiteverkoksi. Keskeisin suhdetyyppi on *hierarkkinen relaatio laajemman* (broader) ja *suppeamman* (narrower) käsitteen välillä. Sen avulla käsitteistä muodostuu puumaisia rakenteita. Lisäksi voidaan kertoa kahden käsitteen välillä olevan joku muu tarkemmin määrittelemätön semanttinen assosiaatiosuhde. Nämä semanttiset relaatiot on lueteltu taulukossa 11.3.

Hierarkian avulla voidaan suorittaa yksikertaista päättelyä: `skos:broader` suhteesta seuraa automaattisesti käänteinen `skos:narrower` suhde ja yhteen suuntaan määritelty `skos:related` suhde on voimassa toiseenkin suuntaan. Semanttisten suhteiden avulla voidaan toteuttaa esimerkiksi kyselyn laajennusta, jossa yleisempää hakutermiä voidaan laajentaa tarkemmiksi ja ottaa mukaan myös termiin vain assosiatiivisesti liittyviä termejä.

Huomionarvoista kuitenkin on, että hierakkinen suhde (`skos:broader`/-`skos:narrower`) ei ole transitiivinen. Jos esimerkiksi metso tiedetään linnuksi ja linnut ovat eläimiä, ei voida automaattisesti päätellä, että metso on eläin. Tämä hierarkkisen suhteen konservatiivinen tulkinta SKOS-standardissa johtuu siitä, että olemassa olevissa luokituksissa ja sanastoissa transitiivisuus ei myöskään välttämättä toimi. Esimerkiksi hierarkiasta

```
<huonekalut> skos:narrower <peilit>.
<peilit> skos:narrower <meikkipeilit>.
```

ei voi päätellä, että meikkipeilit ovat huonekaluja!

Monasti kyselyn laajennuksen kannalta tärkeä transitiivisuus kuitenkin toimii ja silloin tämä on esitettävissä käyttämällä suhteita `skos:broaderTransitive` ja `skos:narrowerTransitive`, jotka on määritelty `skos:broader`- ja `skos:narrower`-ominaisuuksien yläominaisuuksiksi. SKOS:n semanttiset ominaisuuden muodostavat näin seuraavan `rdfs:subPropertyOf`-ominaisuushierakian:

```
skos:semanticRelation
  skos:related
    skos:broaderTransitive
      skos:broader
    skos:narrowerTransitive
      skos:narrower
```

Koska suppeammasta ominaisuudesta voidaan päätellä yleisemmän omi-

naisuuden olevan myös voimassa, voitaisiin ylläolevassa huonekaluesimerkissä päätellä kolme uutta kolmikkoa:

```
<huonekalut> skos:narrowerTransitive <peilit> .
<peilit> skos:narrowerTransitive <meikkipeilit> .
<huonekalut> skos:narrowerTransitive <meikkipeilit> .
```

Transitiivisen ominaisuuden aliominaisuus ei kuitenkaan välttämättä ole transitiivinen, eikä tässä tapauksessa `skos:narrowerTransitive`-suhdetta voi käyttää esim. kyselyn laajentamiseen. Alla olevassa tapauksessa transitiivisuustulkinta kuitenkin olisi mahdollinen:

```
<eläin> skos:narrower <lintu> .
<lintu> skos:narrower <metso> .
```

Ratkaisuna on jättää päätös siitä, päätelläänkö transitiiviset ominaisuudet vai ei tehtäväksi sovelluskohtaisesti. Transitiivisia ominaisuuksia ei ole tarkoitettu käytettäväksi hierarkian kuvaamisessa, vaan ne päätellään aina `skos:broader/narrower` suhteiden kautta.

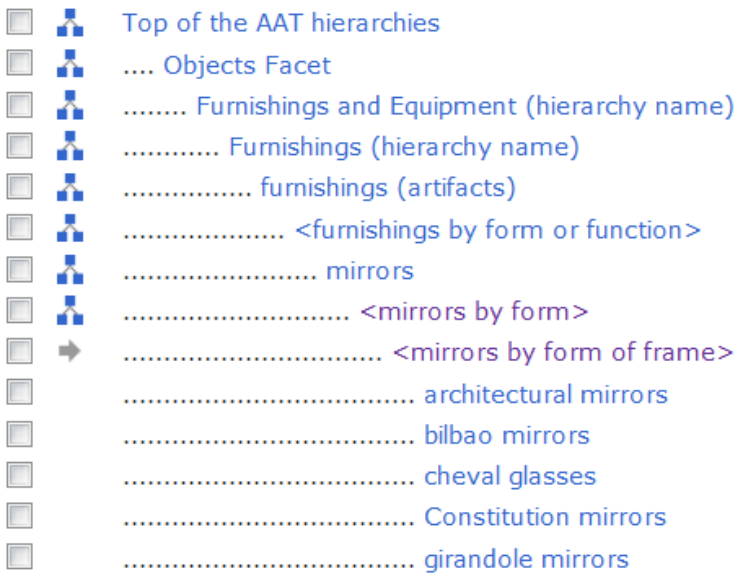
Hierarkiaa voidaan laajentaa uusilla sovelluskohtaisille suhdetyypeillä. Voidaan esimerkiksi erottaa toisistaan *osa-kokonaisuus-suhde* (part-of), esimerkiksi että *Helsinki* on osa *Suomea*, ja yläluokkasuhde (subclass-of) esimerkiksi että luokka *huonekalu* on *seinäpeili*-luokan yläluokka).

11.5 Ohjaustermit

Monissa sanastoissa on varsinaisten indeksoinnissa käytettävien termien ja luokitusten ohella käsitteitä ryhmitteleviä luokituksia eli *ohjaustermejä* (guide term), jotka on tarkoitettu helpottamaan sanaston jäsentämistä ihmiskäyttäjän kannalta, mutta joita ei käytetä tietojen kuvailussa. Esimerkiksi Getty-säätiön Art and Arhitecture Thesaurus (AAT) -sanastossa[4] on ohjaustermi <*Mirrors by form or frame*>, jonka alta löytyy erilaisia peilityyppejä niiden kehyksen laadun mukaan ryhmiteltynä, kuten *architectural mirrors*, *bilbao mirrors*, *cheval glasses*, *Constitution mirrors*, ja *girandole mirrors* (ks. kuva 11.1). Tällaiset käsitteet voidaan esittää SKOS-mallissa *kokoelmina* (collection) luokan `skos:Collection`

ohjaustermi

[4] http://www.getty.edu/research/tools/vocabularies/aat/



Kuva 11.1: Getty-säätiön AAT-sanaston peileihin liittyviä hierarkkisia käsitteitä ja ohjauskäsitteitä, jotka on esitetty kulmasuluissa.

yksilöinä. Kokoelmaan kuuluvat käsitteet tai toiset kokoelmat luetellaan kokoelmayksilön ominaisuuden `skos:member` arvoina.

järjestetty kokoelma

Mikäli kokoelman jäsenet halutaan esittää jossain tietyssä järjestyksessä, esimerkiksi luetella viikonpäivät maanantaista sunnuntaihin, voidaan käyttää *järjestetyn kokoelman* (ordered collection) luokkaa `skos:OrderedCollection`. Tässä tapauksessa kokoelman jäsenet esitetään yhtenä järjestettynä listana ominaisuuden `skos:memberList` arvona. SKOS-päättelyn avulla voidaan päätellä, että järjestetty kokoelma on samalla myös tavallinen kokoelma, jolla on listassa luetellut `skos:member`-jäsenet. Näin erilaiset kokoelmat saadaan muutettua yhteentoimivaan muotoon.

11.6 SKOS-sanastojen siltaaminen

Edellä on käsitelty yhden sanaston käsitteiden kuvaamista SKOS-mallissa. Eri aineistojen sisällönkuvailussa käytetään usein monia eri

Ominaisuus	Merkitys
`skos:closeMatch`	Lähes sama käsite kahdessa eri käsitemallissa
`skos:exactMatch`	Sama käsite kahdessa eri käsitemallissa
`skos:broadMatch`	Laajempi käsite toisessa käsitemallissa
`skos:narrowMatch`	Suppeampi käsite toisessa käsitemallissa
`skos:relatedMatch`	Merkitykseltään liittyvä käsite toisessa käsitemallissa

Taulukko 11.4: Eri käsitemalleihin kuuluvien SKOS-käsitteiden siltaamiseen tarkoitetut ominaisuudet

sanastoja. Jotta aineistot saataisiin yhteentoimiviksi ja esimerkiksi yhtenäisen haun piiriin, täytyy tietokoneen ymmärtää miten eri sanastojen käsitteet liittyvät toisiinsa. Tätä varten SKOS-mallissa on määritelty yleinen ominaisuus `skos:mappingRelation` ja sille joukko tarkempia *siltausominaisuuksia* (mapping property) taulukon 11.4 mukaisesti. Näiden avulla on mahdollista kuvata kahden *eri* sanaston käsitteiden välisiä suhteita.[5]

siltausominaisuus

SKOS:n sanastojen väliset semanttiset relaatiot on määritelty suhteutettuina yhden sanaston sisäisiin semanttisiin ominaisuuksiin alla olevan `rdfs:subPropertyOf`-hierarkian avulla:

```
skos:semanticRelation
  skos:related
    skos:relatedMatch
  skos:broaderTransitive
    skos:broader
      skos:broadMatch
  skos:narrowerTransitive
    skos:narrower
      skos:narrowMatch
  skos:mappingRelation
    skos:closeMatch
      skos:exactMatch
    skos:relatedMatch
    skos:broadMatch
    skos:narrowMatch
```

Tämä mahdollistaa päättelyn RDF(S)-semantiikan sääntöjen kautta: SKOS-päättelyssä voidaan esimerkiksi kolmikosta

```
<A> skos:broadMatch <B> .
```

[5]Ontologioiden *siltaus* (mapping, alignment, matching) on oma semanttisen webin tutkimusalueensa, jota on esitelty esimerkiksi teoksessa [13].

päätellä seuraavat uudet kolmikot:

```
<A> skos:mappingRelation <B> .
<A> skos:broader <B> .
<A> skos:broaderTransitive <B> .
<A> skos:semanticRelation <B> .
<A> rdf:type skos:Concept .
<B> rdf:type skos:Concept .
```

11.7 Dokumentointiominaisuudet

dokumentointi-ominaisuus

SKOS-malliin on määritelty myös joukko *dokumentointiominaisuuksia* (documentation property), joiden avulla resursseihin voidaan liittää niihin liittyviä tekstuaalisia kuvauksia ihmislukijoita ja sanastojen kehittäjiä varten. Taulukossa 11.5 on esitty yhteenveto käytettävissä olevista dokumentointiominaisuuksista.

Ominaisuus	**Merkitys**
`skos:note`	Yleinen dokumentointiominaisuus
`skos:changeNote`	Käytetään käsitteen pienten muutosten kuvaamiseen käsitemallin ylläpitoa varten
`skos:definition`	Käsitteen merkityksen kuvaus tai määritelmä
`skos:editorialNote`	Käsitteen hallintaan liittyvä kommentointi, esimerkiksi huomautus keskeneräisestä työstä
`skos:example`	Ominaisuudella annetaan esimerkkejä käsitteen käytöstä
`skos:historyNote`	Merkittävät käsitteeseen tehdyt aiemman muutokset
`skos:scopeNote`	Erityisesti käsitteen käyttöön liittyvää kuvausta

Taulukko 11.5: SKOS-käsitteiden dokumentointiin tarkoitetut ominaisuudet. Näiden arvona on ihmisen luettavaksi tarkoitettu merkkijono.

11.8 Päättelysäännöt ja integriteettiehtoja

integriteettiehto

SKOS-malli sisältää 46 *integriteettiehtoa* (integrity constraint), joiden oletataan olevan voimassa valideissa SKOS-sanastoissa. Esimerkiksi ehto S9 edellyttää, että sama resurssi ei voi olla samanaikaisesti luokan

`skos:ConceptScheme` ja `skos:Concept` yksilö. Integriteettisääntöjen avulla voidaan määritellä SKOS-sanaston semantiikkaa, täsmentää sen käyttöä sekä validoida sanastoja.

Luku 12

Web Ontology Language OWL

RDF Schema and SKOS ovat yksinkertaisia kieliä semanttisen webin (aihe)ontologioiden esittämistä varten. Niiden ilmaisuvoima on kuitenkin melko rajoittunutta, ja vastaavasti näillä kielillä kehitettyjen ontologioiden avulla ei voida tehdä kovin syvällistä päättelyä, joka voi olla tarpeen älykkäiden verkkopalveluiden kehittämisessä.

Näitä haasteita varten on W3C:n piirissä standardoitu ilmaisuvoimainen kieli ja standardi, Web Ontology Language OWL[1]. OWL:n ensimmäinen suositus julkaistiin vuonna 2004 ja uusimmasta vuonna 2012 julkaistusta suosituksesta käytetään nimitystä OWL 2. Koska OWL 2 korvaa aiemman OWL-spesifikaation, käytetään siitä tässä teoksessa jatkossa yksinkertaisesti nimeä OWL.

Ontologiakielen päävaatimukset ovat:

- Helppokäyttöinen syntaksi, jolla ihminen määrittelee käsitteet.
- Formaali semantiikka, johon perustuen tietokone voi tehdä päätelmiä.
- Riittävä ilmaisuvoima sovellusten tietämyksen esittämistä varten.
- Laskennallinen tehokkuus.

[1] https://www.w3.org/2001/sw/wiki/OWL

Seuraavassa esitellään näiden vaatimusten pohjalta ensin OWL:n perustana olevan lause- ja predikaattilogiikan perusteet ja sitten itse OWL.

12.1 Johdatus logiikkaan

Logiikka muodostaa semanttisen webin semanttisen perustan. Sen avulla määritellään verkon datan ja ontologioiden merkitys täsmällisellä tavalla, joka voidaan ymmärtää yksikäsitteisesti sekä ihmisten että koneiden toimesta. Logiikka mahdollistaa datan automattisen rikastamisen uutta tietoa päättelemällä sekä automaattisen *ongelmien ratkonnan* (problem solving). Tässä mielessä logiikka voidaan rinnastaa aritmetiikkaan, jonka avulla myöskin voidaan esittää tietoa, tuottaa mekaanisesti laskemalla uutta tietoa sekä ratkoa ongelmia.

ongelmien ratkonta

Tässä luvussa esitellään ensin loogisen päättelyn kannalta keskeinen syllogismin (syllogism) käsite modernin logiikan perustana. Tämän jälkeen luodaan katsaus semanttisessa webissä käytettyihin logiikkajärjestelmiin, lauselogiikkaan ja predikaattilogiikkaan, joka muodostaa semanttisen webin semanttisen perustan.

12.1.1 Logiikan idea

Aristotelesta voidaan pitää sekä ensimmäisenä ontologina että loogikkona, sillä hän esitti ensimmäisen mallin päättelystä mekaanisena järjestelmänä. Aristoteleen järjestelmä perustuu tiedon esittämiseen väittäminä eli *propositioina* (proposition). Propositiot esittävät sovellusmaailmaa (domain) kuvaavia *tosiasioita* (fact) ja Aristoteleellä niitä oli neljää eri tyyppiä, jotka on lueteltu alla (suomennettuina):

propositio

tosiasia

1. “Kaikki A ovat B”
2. “Joku A on B”
3. “Mikään A ei ole B”
4. “Joku A ei ole B”

Propositioista voidaan johtaa uusia propositioita *syllogismien* (syllogism) avulla. Syllogismit ovat muotoa

syllogismi

$$C_1, \ldots, C_n \rightarrow H$$

olevia sääntöjä ja ne voidaan tulkita seuraavalla tavalla: jos säännön ehdot eli *premissit* (premise)[2] C_1,...,C_n pitävät paikkansa, niin silloin myös *päätelmä* (conclusion) H pitää paikkansa Esimerkiksi kolmesta muotoa *Kaikki A ovat B* olevasta propositiosta voidaan muodostaa syllogismi

ehto

päätelmä

Kaikki X ovat Y, Kaikki Z ovat X $\rightarrow$ Kaikki Z ovat Y,

jolla voidaan tehdä seuraava päätelmä kreikkalaisten kuolevaisuudesta:

"Kaikki ihmiset ovat kuolevaisia", "Kaikki kreikkalaiset ovat ihmisiä" $\rightarrow$
"Kaikki kreikkalaiset ovat kuolevaisia"

Syllogismit kuvaavat inhimillisen ajattelun rakennetta ja olemusta. Aristoteleen suuri oivallus oli, että tiettyä muotoa olevat syllogismit pitävät *aina* paikkansa propositioiden varsinaisesta sisällöstä riippumatta. Tällaisia universaaleja totuuksia kutsutaan nykyisin *tautologioiksi* (tautology).

tautologia

Periaatteessa sama idea on käytössä moderneissa logiikoissa ja semanttisessa webissä: metadata ja ontologiat kuvataan propositioina, jotka vastaavat RDF-verkon kaaria ja rakenteita, ja syllogististen sääntöjen avulla voidaan RDF-verkkoon päätellä lisää tietoa, ts. uusia kaaria.

12.1.2 Lauselogiikka

Semanttinen web perustuu *tekoälytutkimuksen* (Artificial Intelligence)[3] piirissä paljon käytettyyn *predikaattilogiikkaan* (predicate logic). Predikaattilogiikan perustana on yksinkertaisempi *lauselogiikka* (propositional

tekoäly

predikaattilogiikka

[2] Aristotelesin syllogismeissa oli vain kaksi ehtoa.

[3] Hyvä laajempi johdanto tekoälyyn on teos [41]. Suomeksi aiheesta ilmestynyt aiemmin mm. teos [23].

lauselogiikka logic), jota se laajentaa ilmaisuvoimaltaan. Lauselogiikassa tietoa esitetään *väitelauseina* (assertion) eli *propositiona* (proposition), jotka kuvaavat sovellusmaailmaa ja ovat sen tilasta riippuen joko tosia (T) tai epätosia (E). Esimerkiksi:

```
Helsinki on Suomen pääkaupunki
Eilen Espoossa satoi vettä
Sibelius sävelsi rock-musiikkia
```

teoreema Tosia propositioita kutsutaan *teoreemoiksi* (theorem) .

Lauselogiikan lauseet eivät jakaudu osiin ja niitä kutsutaankin usein *ato-*
atomilause *milauseiksi* .

Konnektiivi	**Nimi**	**Merkitys**
$\neg A$	negaatio	A ei ole tosi
$A \vee B$	disjunktio	A tai B (tai molemmat) on totta
$A \wedge B$	konjunktio	A ja B ovat totta
$A \Rightarrow B$	implikaatio	Jos A on totta niin myös B on totta
$A \Leftrightarrow B$	ekvivalenssi	A ja B ovat totta/epätotta samanaikaisesti

Taulukko 12.1: Lauselogiikan konnektiivit

Lauselogiikan ideana on tarjota mahdollisuus atomilauseiden yhdistämiseen konnektiiveilla, jolloin maailmaa voidaan kuvailla rikkaammilla tavoilla. Yleisimmin käytetyt konnektiivit ja niiden merkitys on esitetty taulukossa 12.1.

A	B	$A \Rightarrow B$
T	T	T
T	E	E
E	T	T
E	E	T

Taulukko 12.2: Implikaation ($\Rightarrow$) totuustaulu

Lauselogiikan lauseita voidaan rakentaa yhdistelemällä konnektiiveilla atomilauseita ja muita lauseita sisäkkäisiksi rakenteiksi. Näiden totuusarvot voidaan laskea kaikilla atomilauseiden arvokombinaatioilla käyttäen

taulukon 12.1 konnektiivien merkityksiä, jotka määritellään *totuustauluna* (truth table). Esimerkiksi implikaation merkityksen määrittelevä totuustaulu on esitetty taulukossa 12.2. *totuustaulu*

A	B	$A \Rightarrow B$	$(A \Rightarrow B) \wedge A$	$((A \Rightarrow B) \wedge A) \Rightarrow B$
T	T	T	T	T
T	E	E	E	T
E	T	T	E	T
E	E	T	E	T

Taulukko 12.3: Modus Ponens -lauseen totuustaulun laskeminen

Tarkastellaan esimerkkinä totuustaulujen käytöstä lausetta

$$((A \Rightarrow B) \wedge A) \Rightarrow B,$$

joka on nimeltään *Modus Ponens*. Sen osien totuustaulut on esitetty taulukossa 12.3. Siitä huomataan, että Modus Ponens on aina tosi maailman tilaa kuvaavien atomilauseiden totuusarvoista riippumatta. Kyseessä on siis tautologia: implikaatiolla voidaan päätellä, että jos säännön ehto A on tosi, pätee myös seuraus B. Vastaavasti lause, joka on aina epätosi, on *ristiriitainen* (contradiction). Jos lause on tosi ainakin jollain atomilauseiden arvokombinaatiolla, se on *toteutuva* (satisfiable) ja *kontingentti* (contingent), jos lause on lisäksi epätosi ainakin yhdellä kombinaatioilla. *ristiriita* *toteutuva* *kontingentti*

12.1.3 Predikaattilogiikka

Predikaattilogiikka lisää olennaisesti lauselogiikan ilmaisuvoimaa. Keskeinen idea on atomilauseiden korvaaminen tietoa hienosyisemmin esittävillä *predikaateilla*: *predikaatti*

$$predikaatti(Arg_1, \ldots, Arg_n)$$

Esimerkiksi:

```
yliopisto(Aalto)
on-osa(Otaniemi, Espoo)
vaimo(Jean, Aino)
antoi(Jean, Aino, sormus)
```

vakio *muuttuja*

Predikaatti kuvaa sen argumenttien välistä suhdetta, relaatiota. Argumentit voivat olla vakiosymboleita eli *vakioita* (constant), kuten yllä olevissa esimerkissä, mutta myös *muuttujia* (variable), joita merkitsemme jatkossa $x, y, \ldots$. Esimerkiksi se, että Akseli on naimisissa jonkun tuntemattoman henkilön x kanssa, voidaan ilmaista muuttujalla:

vaimo(Akseli, x)

funktio

Predikaatin argumenttina voi vakion ja muuttuja ohella olla myös *funktion* (function) kutsu, joka edustaa funktion argumenteista laskettua arvoa. Esimerkiksi aritmeettisia lausekkeita, kuten

f(x,y)=y*sin(35)/ln(x)

voidaan käyttää predikaattien argumentteina. Samoin funktio

hetu(*henkilö*)

voidaan määritellä henkilön henkilötunnuksen arvon palauttavaksi funktioksi ja käyttää sen kutsua edustamaan henkilön henkilötunnusta.

termi *perustermi* *peruskaava*

Matemaattisesti funktiolla $f(x_1, \ldots, x_n)$ tarkoitetaan relaatiota, joka kuvaa joukon syötteitä (argumenttien x_i arvokombinaatiot) yksikäsitteiselle funktion arvolle. Funktion argumentin arvona voi olla joko vakio, muuttaja tai toinen funktio. Näitä kutsutaan yhdessä *termeiksi* (term). Termiä, jossa ei ole muuttujia, kutsutaan *perustermiksi* (ground term); muuttujaton predikaatti on vastaavasti *peruskaava* (ground formula).

kvanttori *universaalikvanttori*

Predikaattien lisäksi toinen predikaattilogiikan keskeinen innovaatio on *kvanttorit* (quantifier), joilla voidaan ilmaista väittämiä muuttujan eri arvojen yli. *Universaalikvanttorilla* (universal quantifier)

$\forall$ x *lause*

eksistentiaalikvanttori

voidaan kertoa, että *lause* pitää paikkansa kaikilla (for all) arvoilla x. *Eksistentiaalikanttorilla* (existential quantifier)

$\exists$ x *lause*

taas voidaan kertoa, että *lause* pitää paikkansa ainakin jollain x:n arvolla. Esimerkiksi käsitteet *Aikuinen* ja *Poikamies* voidaan määritellä näin:

$\forall x$ Aikuinen(x) $\Leftrightarrow$ Henkilö(x) $\wedge$ ikä(x)>=18

$\forall x, y$ Poikamies(x) $\Leftrightarrow$ Mies(x) $\wedge$ Aikuinen(x) $\wedge\neg\exists y$ Naimisissa(x,y)

lause

Predikaattilogiikassa tietämys esitetään joukkona *lauseita* (statement), joiden syntaksi eli rakenne voidaan määritellä seuraavien sääntöjen avulla:

$$
\begin{aligned}
Lause &\mapsto Atomilause|Lause\ Konnektiivi\ Lause|\\
&\quad Kvanttori\ Muuttuja\ Lause|\\
&\quad \neg Lause|(Lause)\\
Atomilause &\mapsto Predikaatti(Termi, Termi, \dots)|Termi = Termi\\
Termi &\mapsto Vakio|Muuttuja|Funktio(Termi, Termi, \dots)\\
Konnektiivi &\mapsto \vee|\wedge|\Rightarrow|\Leftrightarrow\\
Kvanttori &\mapsto \exists|\forall\\
Vakio &\mapsto \text{a|b}|\dots vakioidennimet\\
Muuttuja &\mapsto x|y|\dots muuttujiennimet\\
Predikaatti &\mapsto \text{omistaa|myi}|\dots predikaattiennimet\\
Funktio &\mapsto \text{ikä|sotu|summa}|\dots funktioidennimet
\end{aligned}
$$

Pystyviivalla '|' kuvataan tässä vaihtoehtoisia muotoja. Esimerkiksi *Lause* on joko *Atomilause*, *Lauseen*, *Konnektiivin* ja toisen *Lauseen* jono, kvanttorilause, *Lauseen* negaatio tai *Lause* suluissa. Näiden muotojen rakenne selviää tarkemmin muiden sääntöjen avulla.

Lauseita voidaan em. sääntöjen mukaan yhdistää mutkikkaammiksi lauseiksi konnektiiveilla "ja" $\wedge$, "tai" $\vee$, implikaatio $\Rightarrow$ ja ekvivalenssi $\Leftrightarrow$ lauselogiikan tavoin. Atomilause, jonka argumentteina ei ole muuttujia (peruskaava), esittää *väittämän* mallinnuksen kohteena olevaan maailmaan (domain) liittyen soveltamalla predikaattia vakiotermeihin tai -funktiokutsuihin. Esimerkiksi: *väittämä*

```
omistaa(Matti,Auto521)
myi(Matti,Auto521,Pekka)
```

Predikaatin tulkinnasta riippuen väittämä on joko tosi tai epätosi. Atomilauseiden totuusarvon määrittely perustuu predikaattien, funktioiden ja vakioiden sovelluskohtaisiin määritelmiin, jossa tarkastelun kohteena oleva maailma on kuvattu matemaattisena joukkona M olioita, joihin viitataan vakioilla. Funktiot määritellään matemaattisina funktioina[4] joukossa M ja predikaatit relaatioina $R \subseteq M_1 \times M_2 \times \dots M_n$. Vakioiden,

[4]Funktion avulla voidaan laskea yksikäsitteinen arvo sen argumenteista; tässä argumentit ja funktion arvo otetaan joukosta M.

tulkinta

funktioiden ja predikaattien sovellusaluekohtaista määritelmää kutsutaan *tulkinnaksi* (interpretation).

Mutkikkaampien lauserakenteiden totuusarvo voidaan määrittää luontevasti semanttisilla säännöillä, joita soveltamalla mielivaltaisen logiikan lauseen totuusarvo voidaan selvittää palauttamalla tarkastelu atomilauseiden totuusarvojen tarkasteluun niiden tulkinnan kautta:

1. $\neg A$ on tosi, jos ja vain jos A ei ole tosi.
2. $A \vee B$ on tosi, jos ja vain jos A tai B tai molemmat ovat tosia.
3. $A \wedge B$ on tosi, jos ja vain jos sekä A että B ovat tosia.
4. $A \Rightarrow B$ on tosi, jos ja vain jos B on tosi aina kun A on tosi.
5. $A \Leftrightarrow B$ on tosi, jos ja vain jos $A \Rightarrow B$ ja $B \Rightarrow A$
6. $\forall x A$ on tosi, jos ja vain jos A on tosi kaikilla x:n arvoilla.
7. $\exists x A$ on tosi, jos ja vain jos A on tosi ainakin yhdellä x:n arvolla.

Sääntöjä voidaan soveltaa esimerkiksi seuraavien lauseiden totuusarvojen laskemiseen:

omistaa(Matti,Auto521) $\vee$ omistaa(Pekka,Auto521)
myi(Matti,Auto521,Pekka) $\Rightarrow$
($\neg$ omistaa(Matti,Auto521) $\wedge$ omistaa(Pekka,Auto521))

Predikaatin argumenttina oleva termi voi olla paitsi vakio myös muuttuja tai funktiokutsu. Silloin väittämän totuusarvo riippuu siitä, mikä arvo muuttujalle valitaan tai funktiokutsulle saadaan. Tähän perustuu predikaattilogiikan käyttäminen kyselykielenä, jossa kysely muotoillaan joukkoja muuttujia sisältäviä lauseita. Vastauksena on joukko muuttujasijoituksia, joissa muuttujille on annettu sellaiset vakiot arvoksi, että kysely on tosi. Esimerkiksi kyselyyn {myi(x,auto521,y)} voidaan päätellä vastaukseksi muuttujasijoitus, jossa luetellaan ne muuttujien x ja y arvokombinaatiot, joilla kysytty lause on tosi, kuten {x/Matti, y/Pekka}. Jos tiedossa on aiempia myyntitapahtumia samalle autolle, saadaan ne vaihtoehtoisina vastauksina. Mekanismi on vastaavanlainen kuin SPARQL-kyselykielessä.

OWL:n tapauksessa predikaattilogiikkaa ei kuitenkaan hyödynnetä kyselykielenä, johon on käytettävissä SPARQL, vaan uusien RDF-kolmikoiden päättelemiseksi tietämyskantaan. Ontologiset luokkakäsitteet määritellään yksipaikkaisina predikaatteja. Esimerkiksi käsitteet *Aikuinen* ja *Poikamies* voidaan määritellä näin:

$\forall$x Aikuinen(x) $\Leftrightarrow$ Person(x) $\wedge$ ikä(x)>=18
$\forall$x,y Poikamies(x) $\Leftrightarrow$ Mies(x) $\wedge$ Aikuinen(x) $\wedge\neg\exists$y Naimisissa(x,y)

Aikuinen(x) ja *Poikamies(x)* vastaavat luokkia eli niiden yksiöiden (vakioiden) joukkoja, joille predikaatti on tosi. Tulkinnassa määriteltyjen vakioiden, funktioiden ja predikaattien avulla voidaan päätellä, mihin luokkiin yksiöt kuuluvat.

Oletetaan esimerkiksi, että on määritelty *Naimisissa*-predikaatilla, kuka on naimisissa kenenkin kanssa, henkilöiden miehuus, ja funktio *ikä* määrittelee kunkin henkilön iän:

naimisissa(Pekka, Paula), naimisissa(Kalle, Kerttu),
Mies(Pekka)=T, Mies(Paula)=F, Mies(Kalle)=T,
Mies(Kerttu)=F, Mies(Matti)=T,
ikä(Pekka)=34, ikä(Paula)=31, ikä(Matti)=17, ikä(Kalle)=25,
ikä(Kerttu)=24

Tästä voidaan päätellä (tietyin oletuksin), ketkä henkilöt ovat poikamiehiä ja aikuisia ja lisätä tuo tieto tietämyskantaan:

Aikuinen(Pekka)=T, Aikuinen(Matti)=F, ...
Poikamies(Pekka)=F; Poikamies(Matti)=T, Poikamies(Kerttu)=F, ...

Tämä tarkoittaa käytännössä `rdf:type` kaarien lisäämistä yksilöiden ja luokkien resurssien välille (esim. `:Pekka rdf:type :Aikuinen`).

OWL-kieleen sisältyy joukko etukäteen sovittuja loogisia aksioomeja, jotka määrittelevät kieleen kuuluvien predikaattien merkityksen. Näiden avulla on mahdollista suorittaa päättelyä. Esimerkiksi RDF(S):n `rdfs:subClassOf` on kaksipaikkainen transitiivinen predikaatti, jolle pätee:

$\forall$x,y,z rdfs:subClassOf(x,y) $\wedge$ rdfs:subClassOf(y,z) $\Rightarrow$ rdfs:subClassOf(x,z)

aksiomi *Aksiooma* on perusmääritelmä, joka on universaalisesti tosi (tautologia) kaikissa mahdollisissa eri tulkinnoissa (maailmoissa). Aksiooma mahdollistaa näin uusien kaarien päätellyn RDF-verkkoon samalla tavalla kaikissa semanttisen webin sovelluksissa.

TBox *RBox* *ABox*

OWL-ontologian luokkakäsitteiden eli termien määrittelystä käytetään nimitystä *TBox* (terminological box). Ominaisuuksien määritelmien osalta nimitys on *RBox* (relational box). Termien ja ominaisuuksien määritelmät voidaan yhdistää niiden avulla kuvattuun dataan, joka on joukko sovellusmaailman asioita kuvaavia kiinnitettyjä atomilauseita. Tästä datasta käytetään nimitystä *ABox* (assertional box). OWL:ssa data esitetään RDF:n avulla eri luokkien yksilöinä ja näiden ominaisuuksina.

Sovelluksissa ontologia ja data tyypillisesti yhdistetään ja yhdistettyyn dataan sovelletaan päättelykonetta, joka rikastaa dataa uusilla kolmikoilla ontologisen tietämyksen mukaisesti. Lopputuloksena on rikastettu datajoukko, johon voidaan tehdä kyselyitä SPARQL-kielellä. Tässä mallissa älykäs toiminnallisuus perustuu päätelmien tekemiseen etukäteen tietoa rikastamalla. Päättelyä voidaan myös tehdä ajon aikana kyselykohtaisesti. Lopputulos näyttää käyttäjän kannalta molemmissa tapauksissa samalta, mutta etukäteen dataa rikastamalla voidaan monasti nopeuttaa ajonaikaista laskentaa.

12.2 OWL-kielen syntaksit

Semanttisen webin standardit on rakennettu toistensa varaan. RDF:n alkuperäinen määrittely perustuu XML-kieleen. RDF Schema laajentaa RDF-kieltä, ja samalla tavalla OWL rakentuu RDF Scheman varaan seuraavana käsitteellisenä kerroksena. Kaikki OWL-kielen ilmaukset esitetään viimekädessä semanttisena RDF(S)-verkkona ja voidaan kirjoittaa esimerkiksi Turtle-notaatiolla.

OWL:n logiikan lauseiden kirjoittaminen verkkona on kuitenkin monasti kömpelöä ja siksi OWL-kielelle on kehitetty yksinkertaisempia notaatioita samaan tapaa kuin Turtle kehitettiin yksinkertaisemmaksi RDF-verkkojen esittämisnotaatioksi kuin alkuperäinen RDF-XML. OWL-kielelle on käytössä seuraavia syntakseja:

1. **RDF(S)** OWL-ilmaukset voidaan esittää RDF(S):n avulla verkkoina esimerkiksi Turtle- tai RDF-XML-notaatiolla[5].

2. **OWL-XML** XML-perustainen erityiskieli OWL:n ilmaisemiseksi.

3. **Funktionaalinen syntaksi**, jota käytetään erityisesti OWL-spesifikaatioissa, mm. W3C:n OWL-määrittelydokumenteissa. *funktionaalinen syntaksi*

4. **Manchesterin syntaksi**[6], joka pyrkii olemaan mahdollisimman yksinkertainen notaatio niin, että sen käyttö vaatii mahdollisimman vähän logiikan osaamista. *Manchesterin syntaksi*

Lisäksi OWL-kieltä varten on käytettävissä graafisia editoreita, kuten Protégé[7] ja TopBraid Composer[8], joiden avulla OWL-kielen monia rakenteita voi muokata interaktiivisesti vain vähän tekstiä editoiden.

Kaikissa tapauksissa kielen semantiikka on kuitenkin sama predikaattilogiikan avulla määritelty. Vaikka OWL perustuu melko sofistikoituun logiikkaan, on sen tavoitteena olla kieli, jota voitaisiin käyttää ilman syvällistä logiikan osaamista. Käytännön sovelluksissa OWL:sta otetaan kuitenkin usein käyttöön vain sen yksinkertaisempia muotoja.

OWL laajentaa RDF Scheman ilmaisuvoimaa merkittävästi tarjoamalla useita uusia tapoja luokkien, yksilöiden ja ominaisuuksien määrittelemiseen. Seuraavassa annetaan yleiskuva näistä esimerkkien avulla. Esitys perustuu W3C:n OWL Primer -dokumenttiin[9], jossa määritellään monipuolisella tavalla sukulaissuhteiden ontologiaa. OWL Primer -dokumentti on toteutettu siten, että samat esimerkit voi nähdä esitettynä kaikilla OWL:n eri syntakseilla (funktionaalinen, RDF/XML, Turtle, Manchester ja OWL/XML). Käytämme seuraavassa erityisesti funktionaalista syntaksia. Siinä esitysmuotona ovat sisäkkäiset funktiokutsut, joiden muoto on:

$$funktio(argumentti_1\ argumentti_2\ ...\ argumentti_n)$$

[5] https://www.w3.org/TR/owl2-mapping-to-rdf/

[6] http://www.w3.org/TR/2012/NOTE-owl2-manchester-syntax-20121211/

[7] Protégé on Stanfordin yliopistossa USA:ssa kehitetty, paljon käytetty open source ontologiaeditori, joka on ladattavissa ohjeineen osoitteesta http://protege.stanford.edu.

[8] Kaupallinen vaihtoehto ontologiaeditorille.

[9] https://www.w3.org/TR/owl2-primer/

Esimerkiksi:

```
EquivalentClasses(
  :Äiti
  ObjectIntersectionOf( :Nainen :Vanhempi )
)
```

Kaikissa syntakseissa käytetään RDF(S) ja OWL-spesifikaatiossa määriteltyjä luokkia ja ominaisuuksia, jotka on määritelty alla olevassa osoitteessa:

`http://www.w3.org/2002/07/owl`

Nämä resurssit tunnistaa jatkossa nimiavaruudesta `owl`:

`prefix owl: http://www.w3.org/2002/07/owl# .`

Esimerkiksi **owl:Class** on OWL-luokkien luokka, joka on määrittely luokan **rdfs:Class** aliluokkana. OWL:n nimiavaruus voidaan määritellä funktionaalisella syntaksilla näin:

`Prefix(owl:=<http://www.w3.org/2002/07/owl#>)`

Tarkempi kuvaus funktionaalisesta syntaksista on julkaistu W3C:n suosituksessa[10].

12.3 Ontologia-dokumentti

OWL-ontologia määritellään abstraktina OWL-dokumenttina, jolla on oltava URI-tunniste. Sen kautta ontologiaan voidaan viitata ja muodostaa dokumenttia vastaava ontologia. Tyypillisesti ontologia määritellään tiedostossa, jolla on URI-osoite ja jossa on kerrottu:

- Ontologian määrittelyssä käytetyt nimiavaruuslyhenteet prefix-lauseilla
- Ontologian mahdollinen versiotieto
- Ontologian käyttämät toiset ontologiat import-lauseilla ilmaistuina
- Ontologian kuvailu ihmislukijalle annotointi-ominaisuuksien avulla

[10] https://www.w3.org/TR/owl2-syntax/

Ontologian perustana on sen *entiteettien* (entity), eli luokkien, yksilöiden ja ominaisuuksien määrittelyt. Ontologian entiteettien tyyppi (luokka) on joko

entiteetti

1. yksilö (`owl:NameIndividual`),
2. luokka (`owl:Class`),
3. *tietotyyppiominaisuus* (`owl:DataTypeProperty`), jonka arvo on data-arvo, tai
4. *objektiominaisuus* (`owl:ObjectProperty`), jonka arvo on resurssi (URI).

tietotyyppiominaisuus

objektiominaisuus

Kaikkien ontologiassa kuvattujen entiteettien tyyppi ilmaistaan ontologiassa deklaraatioiden avulla[11]. Esimerkiksi:

```
Declaration( NamedIndividual( :Jean ) )
Declaration( Class( :Henkilö ) )
Declaration( ObjectProperty( :vaimo ) )
Declaration( DataProperty( :ikä ) )
```

Turtle-notaatiolla sama asia ilmaistaan näin:

```
:Jean rdf:type owl:NamedIndividual .
:Henkilö rdf:type owl:Class .
:vaimo rdf:type owl:ObjectProperty .
:ikä rdf:type owl:DatatypeProperty .
```

Sekä luokat että ominaisuudet voivat muodostaa hierarkioita RDFS:n tapaan. Lisäksi niille voidaan OWL:ssä määritellä monenlaisia ominaisuuksia ja keskinäisiä riippuvuuksia. Ideana on, että kun ontologian terminologian (TBox ja RBox) avulla kuvataan reaalimaailman yksilöitä ja niiden välisiä suhteita datana (ABox), voidaan kuvauksen perusteella datasta päätellä uusia suhteita (kolmikoita) entiteettien välille, eli rikastaa dataa älykkäästi.

Seuraavassa esitellään esimerkkien avulla ontologisten käsitteiden määrittelyä OWL-kielellä.

[11] Manchesterin syntaksissa deklaraatioita ei kuitenkaan käytetä, vaan kone päättelee ne automaattisesti.

12.4 Luokkien määrittely

OWL perustuu RDFS:n tapaan luokkien ja ominaisuuksien hierarkiaan, joiden avulla määritellään tietämyksen esittämisessä käytettävät käsitteet eli terminologia (TBox). Terminologisia käsitteitä määritellään kahdella tavalla:

1. *Ekvivalenssilla* termin ja sen kuvauksen välillä. Esimerkiksi *leski* voidaan määritellä samaksi asiaksi kuin henkilö, jonka aviopuoliso on kuollut. Tällöin yksilön leskeydestä voidaan päätellä, että hänen aviopuolisonsa on kuollut, ja kääntäen aviopuolison kuolemasta voidaan päätellä henkilön leskeys.

2. *Aliluokkasuhteella* termin ja kuvauksen välillä. Esimerkiksi *opiskelija* voidaan määritellä sellaisten henkilöiden aliluokkana, jotka opiskelevat. Osa opiskelevista henkilöistä ei siis välttämättä ole opiskelijoita, vaan esimerkiksi tutkijoita. Tällöin opiskelijuudesta voidaan päätellä, että yksilö opiskelee, mutta käänteisesti henkilön opiskelemisesta ei voida päätellä hänen olevan opiskelija, kuten ekvivalenssimäärittelyn tapauksessa.

Ontologiassa määritellyn terminologian mukainen data, esimerkiksi että resurssi `x-4564` on luokan `Henkilö` yksilö ja että hän opiskelee, esitetään sitten luokkien yksilöinä ja näiden välisinä suhteina (ABox) RDF:n avulla.

12.4.1 Samuus ja eriys

URI-tunnisteet yksilöivät luokat, yksilöt ja ominaisuudet. Eri aineistoja verkossa yhdistettäessä ovat aineistojen tuottajat kuitenkin voineet luoda samoille asioille eri tunnisteita. Esimerkiksi maamme pääkaupungilla Helsinki on eri tunniste Maanmittauslaitoksen Paikannimirekisterissä, Geonames-rekisterissä ja DBpediassa. Tästä johtuen OWL:ssa ei tehdä ns. *yksilöivien tunnisteiden oletusta* (Unique Name Assumption, UNA), jonka mukaan eri tunnisteilla kuvatut resurssit kuvaavat eri asioita. Tämä tieton esittämistä yksinkertaistava oletus voidaan yleensä tehdä tietokantajärjestelmissä, joissa data on peräisin yhdestä lähteestä ja tunnisteet on muodostettu systemaattisesti koko tietokannan perusteella.

yksilöivien tunnisteiden oletus

OWL:ssa kaksi eri tunnisteilla varustettua luokkaa voidaan merkitä samoiksi `owl:equivalentClass`-ominaisuudella. Tämä tarkoittaa sitä, että luokilla on täsmälleen samat yksilöt. Tarkastellaan esimerkiksi seuraavaa funktionaalisella syntaksilla esitettyä ontologiaa:

```
Prefix(:=<http://esimerkki.fi/owl/perhe/>)
Ontology(<http://esimerkki.fi/owl/perhe/>
  EquivalentClasses (:Henkilo :Ihminen)
)
```

Määrittely vastaa seuraavaa graafia Turtle-notaatiolla esitettynä.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix : <http://esimerkki.fi/owl/perhe/> .
@base <http://esimerkki.fi/owl/perhe/> .

<http://esimerkki.fi/owl/perhe/> rdf:type owl:Ontology .

:Ihminen rdf:type owl:Class .
:Henkilo rdf:type owl:Class ;
  owl:equivalentClass :Ihminen .
```

OWL-ontologioiden muunnoksia eri muotojen välillä (Manchester OWL Syntax, OWL/XML, OWL Functional Syntax, RDF/XML, Turtle) voi kokeilla verkossa Manchesterin yliopiston OWL Syntax Converter -palvelulla[12]. Esimerkissä 'ö' on korvattu kirjamelle 'o', koska em. verkkopalvelu käyttää perinteistä ASCII-merkistöä.

Vastaavasti voidaan kertoa, että joukko luokkia ovat pareittain erillisiä, jolloin millään joukon luokalla ei ole yhtään yhteistä yksilöä minkään toisen luokan kanssa:

```
DisjointClasses( :Mies :Nainen :Koira :Kissa )
```

Tämä ilmaistaan RDF-verkolla luomalla yksilö luokasta `owl:AllDisjointClasses` ja luettelemalla erilliset luokat listana, joka asetetaan `owl:members` ominaisuuden arvoksi:

```
[] rdf:type owl:AllDisjointClasses ;  # [] on nimetön resurssi
  owl:members ( :Mies  :Nainen :Koira :Kissa ) .
```

[12] http://owl.cs.manchester.ac.uk/tools/webapps/owl-syntax-converter/

12.4.2 Luokkien määrittely joukko-opillisesti

Luokat ovat niihin kuuluvien yksilöiden joukkoja. Yksinkertaisimmillaan luokka voidaan OWL:ssa määritellä luettelemalla sen yksilöt eksplisiittisesti. Esimerkiksi viikonpäivän käsite voidaan määritellä seuraavalla tavalla ekvivalenssin avulla:

```
EquivalentClasses(
  :Viikonpäivä
   ObjectOneOf( :Maanantai :Tiistai :Keskiviikko :Torstai
                :Perjantai :Lauantai :Sunnuntai )
)
```

arvorajoite

Tämän ohella luokka voidaan määritellä implisiittisesti määrittelemällä sen ominaisuuksille *arvorajoitteita* (property restriction). Luokan yksilöitä ovat silloin ne resurssit, jotka ovat rajoitteiden mukaisia.

Yksinkertaisin arvorajoite on vaatimus, että luokan yksilöillä on tietty ominaisuuden arvo. Esimerkiksi käsite *Nokian tuote* voidaan määritellä niiden yksilöiden joukoksi, joiden valmistaja on Nokia.

```
EquivalentClasses(
  :NokianTuote
   ObjectHasValue( :valmistaja :Nokia )
)
```

Teknisesti arvorajoitteet ilmaistaan RDF-muodossa luomalla luokasta `owl:Restriction` yksilö, jonka ominaisuudet kertovat, millaisesta rajoitteesta on kysymys. Edellinen esimerkki RDF-muodossa voidaan esittää näin:

```
:NokianTuote owl:EquivalentClass [
   rdf:type owl:Restriction ;
   owl:onProperty  :valmistaja ;
   owl:hasValue    :Nokia
] .
```

Määrittelyssä muodostuu hakasulkujen sisällä kuvattu nimeämätön arvorajoiteyksilö.

ObjectHasValue-muodon ohella käytössä on ObjectHasSelf-muoto, jolla voi esittää vaatimuksen siitä, että yksilön ominaisuus viittaa yksilöön itseensä (self restriction). Esimerkiksi narsistin käsite voidaan määritellä näin:

```
EquivalentClasses (
  :Narsisti
   ObjectHasSelf ( :rakastaa )
)
```

Arvorajoite voi koskea myös ominaisuuden arvojen lukumäärää eli olla *kardinaliteettirajoite* (cardinality restriction) tai rajata arvojen tyyppiä eli olla *tyyppirajoite*. Lukumäärälle voidaan asettaa minimi (`owl:ObjectMaxCardinality`), maksimi (`owl:ObjectMaxCardinality`) ja tarkka arvo (`owl:ObjectExactCardinality`). Esimerkiksi se tieto, että Matilla on korkeintaan kaksi poikaa, voidaan ilmaista funktionaalisella syntaksilla näin:

kardinaliteettirajoite

```
ClassAssertion(
  ObjectMaxCardinality ( 2 :lapsi :Poika )
  :Matti
)
```

Predikaattilogiikan universaali- ja eksistentiaalikvanttoreita käytetään OWL:ssa luokkamäärittelyissä esittämään luokkakohtainen vaatimus ominaisuuksien arvojen tyypistä ja pakollisuudesta. Muodolla `Object-AllValuesFrom` voidaan vaatia, että tietyn ominaisuuden kaikki arvot ovat annetun luokan yksilöitä. Esimerkiksi onnellinen henkilö voidaan määritellä sellaiseksi henkilöksi, jonka kaikki lapset ovat onnellisia henkilöitä näin:

```
EquivalentClasses(
 :OnnellinenHenkilö
  ObjectAllValuesFrom( :lapsi :OnnellinenHenkilö )
)
```

Predikaattilogiikan avulla ilmaistuna tämä tarkoittaa:

$$\forall x, y \; \mathit{OnnellinenHenkilö}(x) \Leftrightarrow \mathit{lapsi}(x, y) \wedge \mathit{OnnellinenHenkilö}(y)$$

Huomattavaa on, että universaali kvantifiointi ei edellytä, että yksilöllä on ainakin yksi kvantifioitu omaisuus. Esimerkiksi edellisen määrittelyn mukaan myös henkilö, jolla ei ole lainkaan lapsia, on onnellinen. Jos halutaan asettaa onnellisuuden ehdoksi myös vaatimus ainakin yhdestä lapsesta, pitää onnellisen henkilön määritelmään liittää myös eksistentiaalirajoite (`ObjectSomeValuesFrom`):

```
EquivalentClasses(
 :OnnellinenHenkilö
  ObjectAllValuesFrom( :lapsi :OnnellinenHenkilö )
  ObjectSomeValuesFrom( :lapsi :OnnellinenHenkilö )
)
```

Eksistentiaalinen kvantifiointirajoite on sama asia kun minimikardinaliteettirajoite arvolla yksi.

12.4.3 Luokkien määrittely joukko-opilla

Luokkia voi määritellä myös joukko-opillisten operaatioden unioni (`ObjectUnionOf`), leikkaus (`ObjectIntersectionOf`) ja komplementti (`ObjectComplementOf`) avulla. Esimerkiksi `Äiti`-luokka voidaan määritellä luokkien `Vanhempi` ja `Nainen` leikkauksena:

```
EquivalentClasses(
 :Äiti
  ObjectIntersectionOf( :Nainen :Vanhempi )
)
```

Edellä on määritelty luokkia ekvivalenssin (EquivalentClasses) avulla. Määrittelyissä voi käyttää myös `subClassOf`-suhdetta, jolloin aliluokkasuhde ilmaisee välttämättömän, muttei riittävää ehtoa luokkaa kuulumiselle. Esimerkiksi määrittelyn

```
SubCLassOf(
  :Isoisä
  ObjectIntersectionOf( :Mies :Vanhempi )
 )
```

mukaan `Isoisä`-luokan yksilöt ovat välttämättä mies- ja vanhempiluokan yksilöitä, mutta mies- ja vanhempi-luokkiin kuulumisesta ei käänteisesti välttämättä seuraa, että yksilö on isoisä.

12.5 Ominaisuuksien määrittely

OWL:ssa on kolmenlaisia ominaisuusluokkia:

objektiominaisuus

1. *Objektiominaisuudet* ovat luokan `owl:ObjectProperty` yksilöitä ja ilmaisevat resurssien välisiä suhteita.

2. *Tietotyypiominaisuudet* ovat `owl:DataTypeProperty`-luokan ominaisuuksia ja antavat resurssille jotain tietotyyppiä olevan arvon, kuten merkkijonon tai lukuarvon. *tietotyyppiominaisuus*

3. *Annotointiominaisuudet* instantioidaan luokasta `owl:AnnotationProperty` ja ovat ominaisuuksia, joita käytetään vain datan dokumentoinnissa ihmisille. Niiden avulla siis ei suoriteta mitään päättelyä. *annotointiominaisuus*

Objektiominaisuuksille voidaan määritellä seuraavassa kuvattavia matemaattisia suhteita ja piirteitä, jotka ovat varsin hyödyllisiä päättelyssä.

12.5.1 Kääteisominaisuus

Ominaisuudelle voidaan määritellä *kääteisominaisuus*. Esimerkiksi `aviomies`-ominaisuuden käänteisominaisuus on `aviovaimo`: *kääteisominaisuus*

```
:aviomies owl:inverseOf :aviovaimo .
```

Ideana on, että jos jompikumpi suhde tiedetään, toinen voidaan automaattisesti päätellä ja lisätä vastaava kolmikko RDF-tietämyskantaan.

12.5.2 Ominaisuuksien poissulkevuus

Kahdesta ominaisuudesta voidaan sanoa, että ne ovat toisensa *poissulkevia* (disjoint). Ajatuksena on, että kahta yksilöä ei koskaan voi yhdistää kahdella toisensa poissulkevalla ominaisuudella. Esimerkiksi ominaisuudet *puoliso* ja *lapsi* ovat toisensa poissulkevia, koska oman lapsensa kanssa ei voi mennä naimisiin: *poissulkeva ominaisuus*

```
:puoliso owl:propertyDisjointWith :lapsi .
```

12.5.3 Symmetrinen ominaisuus

Ominaisuus voidaan määritellä symmetriseksi luomalla ominaisuusyksilö luokasta `owl:SymmetricProperty`. Esimerkiksi *aviopuoliso* on symmetrinen suhde: *symmetrinen ominaisuus*

```
:aviopuoliso rdf:type owl:SymmetricProperty .
```

Symmetrinen suhde voidaan päätellä automaattisesti "molempiin suuntiin": jos x on y:n aviopuoliso niin symmetrisesti myös y on x:n aviopuoliso.

OWL:ssa voi myös ekplisiittisesti kertoa, että joku ominaisuus ei ole symmetrinen. Esimerkiksi vanhemmat eivät voi olla lapsiensa lapsia:

```
:lapsi rdf:type owl:AsymmetricProperty .
```

12.5.4 Refleksiivinen ominaisuus

refleksiivinen ominaisuus

Refleksiivinen omaisuus liittää yksilön automaattisesti itseensä. Esimerkiksi *sukulainen* on refleksiivinen ominaisuus, koska jokainen on itsensä sukulainen:

```
:sukulainen rdf:type owl:ReflexiveProperty .
```

irrefleksiivinen ominaisuus

Käänteisesti ominaisuus voidaan myös ilmoittaa *irrefleksiiviseksi* (irreflexsive property). Esimerkiksi kukaan ei voi olla itsensä isä:

```
:isä rdf:type owl:IrreflexiveProperty .
```

12.5.5 Funktionaalinen ominaisuus

funktionaalinen ominaisuus

Funktionaalisella ominaisuudella voi olla vain yksi arvo kuten funktiolla. Esimerkiksi se yksiavioisuuden vaatimus, että jokaisella henkilöllä voi olla vain yksi puoliso, voidaan ilmaista kertomalla, että *puoliso* on funktionaalinen ominaisuus:

```
:puoliso rdf:type owl:FunctionalProperty .
```

Funktionaalinen ominaisuus ei ole samalla pakollinen ominaisuus: edellinen määrittely sallii sen, että henkilöllä ei ole lainkaan puolisoa.

12.5.6 Käänteisesti funktionaalinen ominaisuus

käänteisesti funktionaalinen ominaisuus

Käänteisesti funktionaalinen ominaisuus tarkoittaa sitä, että yksilön ominaisuuden arvo yksilöi yksilön yksikäsitteisesti, eli että millään kahdella yksilöllä ei voi olla samaa käänteisesti funktionaalisen ominaisuuden arvoa. Esimerkki tällaisesta ominaisuudesta on vaikkapa puhelinnumero.

```
:puhelinnumero rdf:type owl:InverseFunctionalProperty .
```

Yksiavoisessa maailmassa myös *aviomies* on käänteisesti funktionaalinen ominaisuus.

12.5.7 Transitiivinen ominaisuus

Transitiivinen ominaisuus, esimerkiksi *perillinen* sukupuussa, mahdollistaa päättelyn ominaisuudesta muodostuvien ketjujen kautta: jos x on y:n perillinen ja y on z:n perillinen, niin voidaan päätellä, että x on myös z:n perillinen:

transitiivinen ominaisuus

```
:perillinen rdf:type owl:TransitiveProperty .
```

12.5.8 Ominaisuusketjut

OWL-spesifikaatioon on lisätty mahdollisuus määritellä ominaisuuksia *ominaisuusketjujen* (property chain) avulla. Esimerkiksi isovanhemmuus voidaan määritellä muodolla

ominaisuusketju

```
:isovanhempi owl:propertyChainAxiom ( :vanhempi :vanhempi ) .
```

jossa `isovanhempi`-ominaisuus tarkoittaa sitä, lapsenlapsesta lähtee ominaisuuskaari `vanhempi` tämän vanhempaan ja tästä samanlainen kaari edelleen isovanhempaan.

12.5.9 Avaimet

Yksilön tunnistaminen sen ominaisuuksien perusteella on OWL-päättelyn yksi keskeinen tehtävä. Yksi hyvä väline tähän ovat edellä kuvatut käänteisesti funktionaaliset ominaisuudet. Toinen vielä geneerisempi tapa on käyttää OWL:n *avainmekanismia*, jossa luokkaan liitetään lista ominaisuuksia, joiden arvokombinaatio yksilöi luokan yksilöt yksikäsitteisesti. Esimerkiksi henkilön yksilöinti etunimen, sukunimen ja syntymävuoden avulla voidaan ilmaista näin:

avainmekanismi

```
:Henkilö owl:hasKey ( :etunimi :sukunimi :syntymäaika ) .
```

12.6 Tietotyyppiominaisuudet

OWL tarjoaa välineitä paitsi objektiominaisuuksien myös tietotyyppiominaisuuksien määrittelyyn. Tietotyyppien arvoalue voidaan esimerkiksi määritellä rajoituksilla ja toisten tietotyyppien kombinaatioilla vastaavaan tapaa kuin XML Schemassa. Alla on esimerkkinä määritelty henkilön ikä kokonaisluvuksi välillä 0...150.

```
DatatypeDefinition(
  :ikä
  DatatypeRestriction( xsd:integer
    xsd:minInclusive "0"^^xsd:integer
    xsd:maxInclusive "150"^^xsd:integer
  )
)
```

Tietotyyppiominaisuuksille voidaan myös määritellä vastaavia ominaisuuksia kuin objektiominaisuuksille, esimerkiksi funktionaalisuus, ja tietotyyppi voidaan määritellä myös luettelemalla sen mahdolliset arvot. Esimerkiksi:

```
DatatypeDefinition(
  :taaperoikä
  DataOneOf( "1"^^xsd:integer "2"^^xsd:integer )
)
```

Arvoalueidän määrittämisessä voidaan käyttää myös joukko-opillisia operaatioita.

12.7 Annotointiominaisuudet

12.7.1 Entiteettien ja aksioomien annotointi

annotointiominaisuudet

Annotointiominaisuuksien (annotation property) voidaan OWL-ontologiassa määriteltyihin entiteettehin lisätä ihmislukijaa varten niitä kuvaavaa dokumentointia. Esimerkiksi:

```
AnnotationAssertion( rdfs:comment :Orpo
  "Henkilö, jonka vanhemmat ovat kuolleet."
)
```

Entiteettien lisäksi voidaan kommentteja lisätä myös aksioomeihin, kuten:

```
SubClassOf(
  Annotation( rdfs:comment "Kaikki naiset ovat henkilöitä." )
  :Nainen
  :Henkilö
)
```

Tämä annotointi vastaa alla olevaa graafia Turtle-notaatiolla esitettynä:

```
:Nainen rdfs:subClassOf :Henkilö .
[] rdf:type owl:Axiom ;
  owl:annotatedSource :Nainen ;
  owl:annotatedProperty rdfs:subClassOf ;
  owl:annotatedTarget :Henkilö ;
  rdfs:comment "Kaikki naiset ovat henkilöitä."^^xsd:string .
```

12.7.2 Ontologia ja sen osat

Laajempi esimerkki sukulaisuussuhteiden määrittelystä OWL-dokumenttina on esitetty liitteessä A.

12.8 OWL-profiilit

Kaikkien OWL-kielen ilmaisujen muodostamasta kielestä käytetään nimitystä *OWL Full*, täysi OWL. OWL Full sisältää RDF(S)-kielen ja kaikki sen rakenteet voidaan kuvata RDF(S):n avulla. Haasteena on kuitenkin, että kieli ei ole loogisesti ratkeava, ts. on olemassa OWL-ontologioita, joissa päättelijä ei koskaan päädy lopputulokseen, ja päättely voi olla laskennallisesti hidasta. Ratkeavuus ja tehokkuusongelmat voidaan ratkaista rajoittamalla kielen muotoja erilaisiin tapauskohtaisiin OWL:n *profiileihin* (profile), joita ovat OWL EL, OWL QL ja OWL RL. Näiden unioista, joka on laajin tehokkaasti ratkeava OWL-profiili, käytetään nimitystä *OWL DL* (Description Logic).

OWL Full

profiili

OWL DL

Kukin profiili on suunniteltu käytettäväksi omantyyppisille ontologioille, jollaisten käsittelyyn on toteutettavissa tehokkaita päättelykoneita. Samalla kuitenkin joudutaan rajoittamaan laskennallisista syistä kielen ilmaisvoimaa OWL DL profiiliin nähden tavoilla, joiden perusteet eivät

välttämättä ole kovin intuitiivisia sovellusten kannalta. Esitämme seuraavassa lyhyet luonnehdinnat OWL:n profiileista; näissä käytettävissä olevat OWL ilmaisut ja rajoitteet on kuvattu tarkemmin W3C:n OWL-spesifikaatiossa[13].

12.8.1 OWL 2 EL

OWL EL soveltuu erityisesti ontologioille, jossa on paljon luokkia. Murretta käytetään esimerkiksi isojen lääketieteellisten ontologioiden kuten Snomed CT käsittelyssä, joiden hierarkioissa voi olla jopa satoja tuhansia luokkia.

OWL EL -päättelykoneet kykenevät polynomisessa ajassa tarkistamaan ontologian toteutuvuuden (satisfiability), laskemaan luokkasuhteet ja päättelemään yksilöiden kuulumisen luokkiin. Tehokkuuden hintana on rajoitukset mm. negaation, disjunktion ja kaikki-kvanttorin käytössä, eikä esimerkiksi käänteisiä ominaisuuksia voida käyttää.

12.8.2 OWL 2 QL

OWL QL puolestaan soveltuu sellaisten ontologioiden esittämiseen, joissa on paljon yksilöitä ts. dataa. Murteen päättelykone voidaan ohjelmoida tehokkaasti relaatiotietokannan päälle, mikä soveltuu hyvin juuri isojen datajoukkojen käsittelyyn.

Tehokkuuden hintana on mm. rajoituksia kaikki-kvanttorin käytössä ja että ominaisuusketjuja ei voi käyttää eikä ilmaista resurssien samuutta (equality).

12.8.3 OWL 2 RL

OWL RL:n keskeinen käyttötapaus on päättelyn tukeminen. Profiilia on siksi rajoitettu siten, että kieli voidaan toteuttaa tehokkaasta logiikkaohjelmoinnin avulla sääntöjärjestelmänä[14].

[13] https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/

[14] Palaamme sääntöihin ja logiikkaohjelmointiin tarkemmin tuonnempana.

RL-profiililla on kuitenkin monia rajoituksia. Sillä ei voida esittää tilanteita, joissa yksilön olemassaolosta seuraa toisen yksilön olemassaolo. Ei voida esimerkiksi ilmaista, että jokaisella henkilöllä pitää olla vanhemmat.

Näiden OWL 2 -profiilien lisäksi käytössä saattaa vielä olla alkuperäiseen OWL 1 -määrittelyyn kuuluvia profiileja: OWL 1 DL, on osajoukko laajemmasta OWL 2 DL -profiilista, ja OWL Lite, on yksinkertainen hierarkioiden käsittelyyn sopiva profiili.

Luku 13

Päättelysäännöt

Logiikka muodostaa semanttisen webin perustan. Sen avulla määritellään verkon datan ja ontologioiden merkitys täsmällisellä tavalla, joka voidaan ymmärtää yksikäsitteisesti sekä ihmisten että koneiden toimesta. Logiikka mahdollistaa datan automaattisen rikastamisen uutta tietoa päättelemällä sekä automaattisen *ongelmien ratkonnan* (problem solving). *ongelmien ratkonta*

13.1 Semanttisen webin looginen tulkinta

Semanttisen webin kielten RDF, RDFS ja OWL merkitys määritellään edellisessä luvussa kuvatun klassisen ensimmäisen kertaluvun predikaattilogiikan (predicate logic) avulla. Tämä logiikka muodostaa nykyisten loogisten järjestelmien perustan, ja sen monia yksinkertaistettuja muotoja voidaan käsitellä tehokkaasti tietokoneilla päättelytehtäviä varten.

RDF-verkon looginen tulkinta on suoraviivainen: jokainen kolmikko

<*subject*, *predicate*, *object*>

esimerkiksi <Suomi, on-osa, Eurooppa>, voidaan tulkita tosiasian tai väittämänä esittävänä propositiona

$predicate(subject, object)$

kuten:

on-osa(Suomi, Eurooppa)

tietämyskanta

RDF-verkko on silloin joukko samanaikaisesti voimassa olevia väittämiä, jotka muodostavat *tietämyskannan* (knowledge base).

W3C on julkaissut RDF ja RDFS-kielten semantiikan formaalin loogisen spesifikaation[1]. Siinä RDF(S)-rakenteet, kuten `rdfs:subClassOf`, on määritelty *päättelysääntöinä* (entailment rule), joiden periaatteellinen muoto on:

> jos RDF-tietämyskanta E sisältää kolmikot $C_1, \ldots, C_n$, niin kolmikko H voidaan lisätä kantaan E uutena tosiasiana.

Taulukossa 13.1 on esimerkkinä esitetty kaksi RDFS-määrittelyn päättelysääntöä, joilla kuvataan luokkahierarkian merkitys päättelyn kannalta.

Taulukko 13.1: Kaksi RDF(S):n semantiikan määrittelyn päättelysääntöä

Sääntö	**Jos tietämyskanta E sisältää**	**niin lisää:**
rdfs9	x `rdf:type` y, y `rdfs:subClassOf` z	x `rdfs:subClassOf` z
rdfs11	x `rdfs:subClassOf` y, y `rdfs:subClassOf` z	x `rdfs:subClassOf` z

Nämä säännöt tarkoittavat, että jos esimerkiksi tietämyskanta sisältää tosiasiat

```
// Yksilödata
:tuoli-1324 rdf:type :nojatuoli .

// RDFS-ontologia
:nojatuoli rdfs:subClassOf :tuoli .
:tuoli rdfs:subClassOf :istuin .
:istuin rdfs:subClassOf :huonekalu .
```

niin seuraavat uudet tosiasiat voidaan lisätä tietämyskantaan:

```
:tuoli1324 rdf:type :tuoli .
:tuoli1324 rdf:type :istuin .
:tuoli1324 rdf:type :huonekalu .

:nojatuoli rdfs:subClassOf :istuin .
:nojatuoli rdfs:subClassOf :huonekalu .
:tuoli rdfs:subClassOf :huonekalu .
```

Uusi tieto siitä, että nojatuolit ovat huonekaluja, voidaan tämän jälkeen hyödyntää vaikkapa hakukoneessa, kun etsitään erilaisia huonekaluja.

Päättely voidaan tehdä kahdessa eri vaiheessa:

[1] http://www.w3.org/TR/2004/REC-rdf-mt-20040210/

1. Etukäteen ennen tiedon käyttämistä (preprocessing).

2. Dynaamisesti kyselyn aikana.

Etukäteen tehtävä päättely mahdollistaa tehokkaan ajonaikaisen toiminnan, kun osa työstä on silloin jo tehtynä. Haittana on, että tietämyskannan koko kasvaa yleensä merkittävästi ja kaiken mahdollisen päätteleminen voi kestää kauan. Jos jotain päätelmiä ei myöhemmin tarvitakaan, on niiden laskenta ja indeksointi ollut turhaa. Lisäksi päätelmät täytyy muodostaa aina uudelleen kun tietämyskanta muuttuu. Nykyiset kolmikkotietokannat ja hakukoneet kuten Apache Lucene/Solr[2] pystyvät kuitenkin käsittelemään tehokkaasti varsin isojakin RDF-verkkoja, joissa voi olla miljardeja kolmikoita, joten päättelyssä ei tässä mielessä tarvitse yleensä pihistellä.

Päätelmien laskenta kyselyn aikana on joustavampaa, mutta voi olla laskennallisesti haastavaa erityisesti isommilla tietämyskannoilla. Lisäksi samoja päätelmiä, esimerkiksi yksilön luokkien päättelyä luokkahierarkian avulla, joudutaan tekemään uudelleen ja uudelleen eri kyselyjen yhteydessä.

Yksinkertaista päättelyä kyselyn aikana voidaan tehdä paitsi tietämyskantaa kolmikoilla rikastamalla myös SPARQL-kyselykielen avulla, jossa voidaan käyttää esimerkiksi ominaisuusketjuhahmoa

```
?x rdf:type/rdfs:subClassOf* ?y
```

yksilön x kaikkien yläluokkien y päättelemiseksi.

Edellisiä tapoja voidaan myös yhdistellä hyvän kompromissin löytämiseksi niin, että esimerkiksi päätellään etukäteen usein toistuvat tehtävät, kuten yksilö-luokkasuhteiden laskeminen, ja ajon aikana keskitytään vain tapauskohtaisempaan päättelyyn.

OWL-ontologiakieli variaatioineen tarjoaa RDF(S)-kieltä rikkaamman välineistön tietämyksen esittämistä varten ja päätelmien tekemiseen. OWL-profiilien looginen perusta[3] ovat *kuvailulogiikat* (description logics). Vastaavaan tapaan kuin RDF(S)-kielellä, myös OWL:lla on logiikkaan perustuva semantiikka, joka viimekädessä tarkoittaa sitä, että

kuvailulogiikka

[2] http://lucene.apache.org/

[3] Hyvä teos tästä on [20].

OWL-ontologian avulla voidaan päättelemällä tuottaa uutta tietoa, ts. uusia kaaria, RDF-tietokantaan.

OWL:n tyypillinen käyttötapaus on ontologisen tiedon rikastaminen: Ontologian kehittäjä siirtää OWL-malliinsa tietämystä sovelluksen kohteena olevan maailman käsitteistä yleisinä luokka- ja ominaisuusmäärittelyinä (TBox ja RBox) ja kuvaa näiden avulla sovelluksen dataa RDF-väittäminä (assertion) (ABox). Luokkamäärittelyt pyritään esittämään yksinkertaisena hierarkiana ja välttämään moniperintää. Loppukäyttöä varten määrittelyjä rikastetaan OWL-päättelykoneen avulla, joka lisää ontologiaan uusia rakenteita ja mahdollistaa näin älykkäämpien järjestelmien toiminnan. Ideana on, että ontologi kehittää yksinkertaista määrittelyjoukkoa, joka sitten rikastetaan semanttisesti sovellusten käyttöä varten. Tämä helpottaa ontologian kehitystyötä ja ylläpitoa.

13.2 Sääntöjen käyttö

Ontologinen päättely RDF(S)- ja OWL-kielissä kohdistuu terminologisten ja ontologisten relaatioiden, kuten yksilö-luokkasuhteiden (`rdf:type`), aliluokkasuhteiden (`rdfs:subClassOf`), aliominaisuuksen (`rdfs:subPropertyOf`) ja samuuden päättelyyn. Tällaisten yleisten, sovellusriippumattomien suhteiden ohella sovelluksissa on tarvetta käsitellä myös ala- ja sovelluskohtaisia suhteita, kuten vaikkapa kreikkalaisten kuolevaisuutta. Tämä on mahdollista sääntöjärjestelmien avulla.

Hornin logiikka

Hornin klausuuli
literaali (logiikassa)

Ehkä käytetyin looginen järjestelmä sääntöperustaiseen päättelyyn on *Hornin logiikka* (Horn Logic) (HL). HL on predikaattilogiikan osajoukko, joka käsittelee vain tietynlaisia loogisia lauseita, ns. *Hornin klausuuleja* (Horn clause). Klausuulilla (clause) tarkoitetaan logiikassa *literaalien* (literal) disjunktiota

$$C_1 \vee \cdots \vee -C_n$$

missä $\vee$ merkitsee disjunktiota ("tai"-operaattori) ja C_i ovat literaaleja. Logiikassa literaalilla tarkoitetaan RDF-kielestä poiketen *atomikaavaa* (atomic formula) tai sen negaatiota. Atomikaava, esimerkiksi

atomikaava

veli(Juhani,Simeoni)

koostuu yhdestä predikaatista ja sen argumenteista ilman konnektiivien (negaatio, konjunktio jne.) avulla luotua syvempää rakennetta.

Hornin klausuuli (Horn clause) on klausuuli, jossa voi olla korkeintaan yksi positiivinen literaali, jota on merkitty alla H:

$\neg C_1 \vee \cdots \vee \neg C_n \vee H$

Tällainen klausuuli voidaan logiikan sääntöjen mukaan esittää implikaationa, jonka muoto on

$C_1 \wedge \cdots \wedge C_n \Rightarrow H$

eli Hornin klausuulit ovat loogisia sääntöjä.

Päätelmää H kutsutaan säännön *pääksi* (head) ja C_i ovat säännön *ehtoja* (condition). *pää* *ehto*

Sääntöjen avulla voidaan määritellä uusia n-paikkaisia relaatiota (predikaatteja). RDF-tietomallissa relaatiot ovat binaarisia ja vastaavat verkon kaaria. Esimerkiksi alla on määritelty sukulaissuhteet *veli*, *sisko*, *setä*, ja *isoäiti* toisten sukulaissuhteiden avulla:

mies(X) ∧ vanhempi(P,X) ∧ vanhempi(P,Y) ∧ eri(X,Y) ⇒ veli(X,Y)
nainen(X) ∧ vanhempi(P,X)∧ vanhempi(P,Y) ∧ eri(X,Y) ⇒ sisko(X,Y)
veli(X,P) ∧ vanhempi(P,Y) ⇒ setä(X,Y)
äiti(X,P) ∧ vanhempi(P,Y) ⇒ isoäiti(X,Y)

Jos Hornin klausuulilla ei ole ehtoja, silloin H on *tosiasia* (fact). Näiden avulla voidaan kuvata sovellusmaailmaan tietty tila, kuten jonkun perheen sukupuu alla: *tosiasia*

⇒ mies(Jussi)
⇒ mies(Paavo)
⇒ nainen(Maija)
⇒ nainen(Jaana)
⇒ vanhempi(Jussi,Maija)
⇒ vanhempi(Jussi,Paavo)
...

Jos Hornin klausuulilla ei ole päätä H, sanotaan sen ehtojen muodostavan *tavoitteen*(goal). Tavoite voidaan tulkita logiikassa *kyselyksi* (query), jonka vastaukset ovat tavoitteessa olevien muuttujien arvosijoituksia. Esimerkiksi kysely *tavoite* *kysely*

```
veli(X,Y) ⇒
```

palauttaisi ylläolevassa esimerkissä kaikki veljesparit (X, Y) ottaen huomioon tietämyskannan tosiasiat ja säännöt Tavoitteen muuttujasijoitusten päättely siis vastaa SPARQL-kyselyä.

Hornin logiikan merkitys tietotekniikassa perustuu siihen, että siinä esitettyjen kyselyiden ratkaisemiseen on olemassa tehokkaita algoritmeja. Itse asiassa Hornin logiikan perustalle on kehitty kokonainen ohjelmoinnin paradigma, *logiikkaohjelmointi* (logic programming).[4] Tunnetuin logiikkaohjelmoinnin kieli on *Prolog*. Logiikkaohjelma koostuu joukosta Hormin klausuuleja, joista yksi on kysely. Varsinaista perinteistä algoritmia logiikkaohjelmassa ei kuvata, vaan ohjelman ajo eli sen *proseduraalinen tulkinta* (procedural interpretation), syntyy automaattisesti kyselyyn vastaamisesta. Tämä tapahtuu teoreemantodistamisen menetelmällä, jossa haetaan kaikki klausuulit toteuttavat muuttujien arvosijoitukset tarkastelemalla klausuuleja ylhäältä alas ja niiden atomeja vasemmalta oikealle. Todistusmenetelmänä on *käänteinen todistusmenetelmä* (proof by contradiction), jossa tutkitaan, millä sijoituksilla kyselyn negaatio johtaa loogiseen ristiriitaan, mistä voidaan päätellä juuri noiden sijoitusten toteuttavan tietämyskannassa kuvatut ehdot.

logiikkaohjelmointi
Prolog
proseduraalinen tulkinta
käänteinen todistusmenetelmä

Logiikkaohjelmointi on luonteva tapa esittää RDF-muotoista dataa ja tehdä siitä päätelmiä: tosiasiat *predicate*(*subject*,*object*) vastaavat suoraan kolmikoita <*subject*, *predicate*, *object*> kuten RDF:n loogisessa spesifikaatiossa. Hornin säännöillä voidaan näin kuvata tietämystä ja myös toteuttaa tarvittavat päättelyt. Prolog-järjestelmiin kuten SWI-Prolog[5] on integroitu kirjastoja, joiden avulla voidaan lukea RDF-muotoista dataa tietämyskannaksi sekä rikastaa ja käyttää sitä päättelysäännöillä.

Dataa voidaan lisäksi rikastaa käyttämällä erilaisia tekoälyn *sääntöjärjestelmiä* (rule-based system). Esimerkiksi semanttisen webin Apache Jena -kehitysympäristöön[6] sisältyy mahdollisuus päättelysääntöjen esittämiseen ja soveltamiseen ehdoista päätelmiin (forward chaining) tai päätelmistä ehtoihin (backward chaining).

sääntöjärjestelmä

Toinen varsin käytännönläheinen tapa toteuttaa sääntöperustaista päättelyä on käyttää hyväksi SPARQL-kyselyitä. SPARQL Upadate -kielellä

[4] Logiikkaohjelmoinnin teoriaan voi tutustua esimerkiksi teoksessa [35] ja käytäntöön teoksessa [8].

[5] http://www.swi-prolog.org/

[6] https://jena.apache.org/

voidaan tunnistaa tietämyskannan rakenteita (SELECT) ja generoida (CONSTRUCT) ja lisätä sen mukaisesti uusia kolmikoita tietämyskantaan. SPARQL-kielen hahmonsovitus muistuttaa päättelyä ja kieleen sisältyy päättelyä tukevia ominaisuuksia, kuten ominaisuusketjut. SPARQL-sääntöjen avulla on kehitetty erityinen päättelyjärjestelmä SPARQL Inferencing Notation SPIN[7], joka on käytössä esimerkiksi TopBraid Composer-ontologiaeditorissa. Sen avulla voidaan mm. kontrolloida, miten ja missä järjestyksessä sääntöjä sovelletaan. SPIN-järjestelmään sisältyy tapa esittää SPARQL-kyselyt RDF-muodossa, mikä mahdollistaa niiden tallentamisen ja muokkaamisen kolmikkokannassa. Systeemin sisältyy mekanismeja uusien funktioiden määrittelemiseksi kyselyissä ja uudelleenkäyttöä varten toisissa säännöissä.

Päättelyä voidaan suorittaa myös OWL-päättelykoneilla. Käytettävissä on lukuisia OWL-päättelykoneita, jotka saavat syötteenä OWL-ontologian ja rikastavat sitä uusilla kolmikoilla. Tällaisia ovat esimerkiksi Pellet, FaCT++ ja HermiT[8]. Näitä voidaan käyttää vaivattomasti esimerkiksi Protégé ontologiaeditorissa valmiiksi integroituina toiminnallisuuksina.

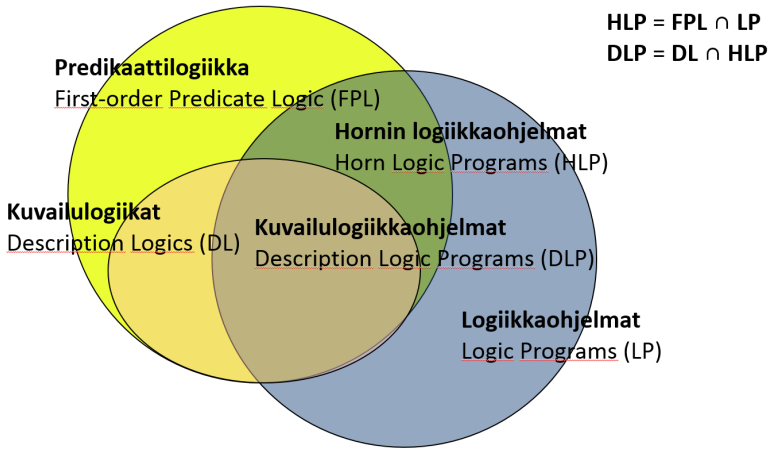
13.2.1 Hornin logiikan suhde kuvailulogiikoihin

Hornin logiikka ja logiikkaohjelmointi tuntuisivat muodostavan luontevan perustan semanttisen webin kerroskakkumallille. Haasteena kuitenkin on, että Hornin logiikalla ei pystytä esittämään kaikkea sitä, mitä OWL:n eri murteet voivat esittää kuvailulogiikoissa. Lisäksi logiikkaohjelmoinnissa ja OWL-standardin mukaisessa päättelyssä tehtävät loogiset oletukset eivät vastaa täysin toisiaan. Tämä on johtanut tilanteeseen, jossa soveltajan on tapauskohtaisesti hahmotettava terminologialogiikoiden ja sääntöjärjestelmien taustalla olevien logiikkajärjestelmien eroja. Tarkastelemme seuraavassa näitä eroja hieman tarkemmin.

OWL-profiilit perustuvat kuvailulogiikoihin (description logic). Ne ovat Hornin logiikan tavoin predikaattilogiikan tehokkaasti laskettavissa olevia osajoukkoja, jotka on tarkoitettu ontologiseen terminologiseen päättelyyn käsitehierarkioiden avulla. Keskeinen kysymys ontologisen päät-

[7] http://spinrdf.org/

[8] Ks. esimerkiksi W3C.n keräämä lista OWL-toteutuksista: https://www.w3.org/2001/sw/wiki/OWL/Implementations



Kuva 13.1: Predikaattilogiikan ja logiikkaohjelmien osa-alueita ja niiden leikkauksia [4]

telyn ja sääntöpohjaisen päättelyn yhdistämisen kannalta on, miten eri kuvailulogiikat suhtautuvat Hornin logiikkaan. Lisäksi haasteena on kysymys siitä, miten OWL-päättelykoneet ja logiikkaohjelmoinnissa käytetyt teoreemantodistimet suhtautuvat toisiinsa. Voidaanko esimerkiksi OWL-päättelykone toteuttaa Hornin logiikan ja logiikkaohjelmoinnin avulla? Silloin pitkään käytössä ollut logiikkaohjelmointi voitaisiin ottaa käyttöön semanttisen webin ohjelmointikielenä. Toisaalta jos Hornin logiikka voitaisiin esittää OWL-kielellä, olisi OWL:n laajentaminen sääntöjärjestelmiin luontevaa.

Valitettavasti OWL:n perustana olevat kuvailulogiikat ja Hornin logiikka kattavat toisiaan vain osittain kuvan 13.1 osoittamalla tavalla. Vaikka molemmat ovatkin predikaattilogiikan osajoukkoja, on olemassa propositioita, jotka voidaan esittää kuvailulogiikoilla, mutta ei Hornin logiikalla. Esimerkiksi se, että ihmiset ovat joko miehiä tai naisia voidaan esittää helposti OWL:ssa, mutta ei sääntöinä. Toisaalta Hormin sääntöjä ei voida aina esittää kuvailulogiikan keinoin.

Kuvailulogiikoiden ja Hornin logiikoiden yhdistämistä voidaan kuitenkin tehdä erilaisilla strategioilla[9]:

1. Käytetään vain Hornin klausuuleita. Tällöin kuvailulogiikoista ote-

[9]Ks. esimerkiksi [20]

taan käyttöön vain ne piirteet, jotka voidaan esittää sääntöinä. Tällaista logiikoista käytetään nimitystä Description Logic Programs (ks. kuva 13.1). Esimerkiksi OWL RL -profiili voidaan toteuttaa Hornin sääntöjen avulla.

2. Laajennetaan OWL:a säännöillä. Tämä lähestymistapa onkin jo otettu käyttöön OWL 2:ssa, jossa on mahdollista ketjuttaa predikaatteja. Esimerkiksi $isovanhempi(X, Y)$ voidaan määritellä kahden $vanhempi(X, Y)$ predikaatin avulla Tässä lähestymistavassa on mahdollista formuloida ehtoja sille, millaisia sääntöjä OWL-ilmaisujen yhteydessä voidaan käyttää, mutta haasteena on, että nämä muotoilut eivät ole kovin intuitiivisia käyttää ja laajennukset voivat olla laskennallisesti haasteellisia.

Ilmaisuvoimaltaan erityisen rikas ehdotus OWL:n ja Hornin logiikan yhdistämiseksi on Semantic Web Rule Language SWRL[10], jonka säännöt voivat olla muotoa:

$$A_1, \ldots, A_n \Rightarrow B_1, \ldots, B_m$$

Tässä A_i ja B_j ovat atomilauseita, joista osaa on voitu käyttää kuvailulogiikassa ja osaa vain säännöissä. SWRL-sääntöihin perustuva logiikka ei ole ratkeava, mutta ongelma on ratkaistavissa rajoittumalla vain ns. *DL-turvallisiin* (DL-safe) sääntöihin. Sääntö on DL-turvallinen, jos sen kaikki muuttujat esiintyvät säännön jonkun vasemman puolen sellaisen atomilauseen argumenttina, jota ei ole käytetty kuvailulogiikan määrittelyissä.

DL-turvallinen

Vaikka matemaattinen ratkaisu ontologisen ja sääntöperustaisen päättelyn yhdistämiseksi olisi olemassa, OWL-päättelyn ja logiikkaohjelmoin yhdistämisessä on vielä lisähaasteita. Logiikkaohjelmoinnissa tehdään perinteisesti kaksi käytännöllistä oletusta, joita OWL-maailmassa ei tehdä: suljetun maailman oletus ja yksikäsitteisten nimien oletus. Tarkastelemme näitä seuraavaksi erikseen.

13.2.2 Suljetun maailman oletus

Logiikkaohjelmoinnissa ja Prolog-kielessä tehdään yleensä *suljetun maailman oletus* (Closed World Assumption) (CWA). Se tarkoittaa, että

suljetun maailman oletus

[10] https://www.w3.org/Submission/SWRL/

väittämät, joita ei tiedetä tai voida todistaa todeksi oletetaan epätodeksi, ts. oletetaan, että kaikki tarpeellinen maailman tieto tunnetaan. Tämä vahva ja usein erittäin käyttökelpoinen oletus on tyypillinen logiikkaohjelmoinnin ohella myös perinteisissä tietokantajärjestelmissä. Jos esimerkiksi emme voi päätellä Jussin olevan nainen – hänet esimerkiksi tiedetään mieheksi – tiedämme, että hän ei ole nainen. Ilman suljetun maailman oletusta tätä ei voida tietää, koska voi olla niin, että tietomme Jussista ovat vain puutteellisia.

Suljetun maailman oletus on kuitenkin haasteellinen tilanteissa, jossa maailma ei ole suljettu, vaan tietomme siitä ovat puutteellisia. Ja avoimessa webissä maailman tieto ei välttämättä ole suljettua vaan frakmentaalista. Silloin oletus siitä, että sitä mitä ei tiedetä on epätotta, johtaa virheellisiin päätelmiin. Jos emme esimerkiksi tiedä, satoiko Pihtiputaalla eilen vai ei ja kysymme *Satoiko Pihtiputaalla eilen?*, suljetun maailman olettava tekoäly vastaa *ei*. *Avoimen maailman olettava* (open world assumption, OWA) järjestelmä taas ei tee mitään päätelmää eli vastaa *en tiedä*. Koska tieto webissä ei yleisesti ottaen voi olla suljettua, OWL-kielessä päädyttiin käyttämään avoimen maailman oletusta.

avoimen maaiman oletus

Monissa tilanteissa tällainen OWL-päättelijän toiminta saattaa kuitenkin tuntua hämmentävältä ja itsestään selvältä tuntuvien asioiden kertominen turhalta. Jos esimerkiksi tiedämme, että Jussi on mies, emme voi päätellä, että hän ei ole nainen, vaan tämä pitää erikseen kertoa. Pitää esimerkiksi määritellä, että miesten ja naisten joukot ovat toisensa poissulkevia.

Perinteiset logiikat kuten predikaattilogiikka, Hornin logiikka ja kuvauslogiikat voivat ainoastaan lisätä tietoa tietämyskantaan, eivät poistaa sieltä tietoa. Tällaisia logiikoita kutsutaan *monotonisiksi* (monotonic). Logiikkaohjelmoinnissa käytetty suljetun maailman oletus johtaa kuitenkin epämonotoniseen päättelyyn, sillä tietämyskantaan lisätty uusi tieto saattaa kumota aiemmin pääteltyä tietoa, eli oletuksen siitä, että asia jota ei tiedetä, on epätosi. Epämonotonisessa päättelyssä tietämyskannan koko voi myös pienetä päättelyn tuloksena. Tällaisia logiikoita kutsutaan *epämonotonisiksi* (non-monotonic).

monotoninen logiikka

epämonotoninen logiikka

13.2.3 Yksikäsitteisten nimien oletus

yksikäsitteisten nimien oletus

Toinen keskeinen ero OWL-päätelyn (ja yleisemmin loogisten teoreemantodistimien) ja logiikkaohjelmoinnin välillä on *yksikäsitteisten nimien oletus* (Unique Name Assumption, UNA). UNA tarkoittaa, että jos kahdella objektilla on sama nimi, objektit ovat samoja, ja käänteisesti erinimiset objektit ovat eri objekteja. Oletetaan esimerkiksi, että monikielisellä ihmisellä voi olla vain yksi äidinkieli, ja että Hannan äidinkieli on suomi. Jos sitten käy ilmi, että hänen äidinkielensä onkin ruotsi, on päädytty loogiseen ristiriitaan. Näin tapahtuukin logiikkaohjelmoinnissa, jossa tehdään yksikäsitteisten nimien oletus: suomi ja ruotsi ovat eri asioita, koska niillä on eri nimet. OWL-päättelyssä UNA-oletusta ei kuitenkaan tehdä, eikä tietokantaa siksi tulkita tässä tapauksessa ristiriitaiseksi. Ilman UNA-oletusta on nimittäin täysin mahdollista, että suomi ja ruotsi ovat sama asia, joilla vaan sattuu olemaan eri nimi. OWL-päättelijä tekee siksi päätelmän siitä, että suomen ja ruotsin on oltava sama asia, koska muuten tilanne olisi ristiriitainen. Moinen päätelmä on hämmentävä, mutta on loogisesti oikein ilman lisätietoja. Jotta OWL-päättelijä ymmärtäisi tilanteen olevan ristiriitainen, sille pitää erikseen kertoa, että suomi ja ruotsi eivät voi olla sama asia.

OWL-ontologioissa joudutaan tästä johtuen usein tilanteeseen, jossa tietokantaan joudutaan lisäämään suuria määriä tietoa siitä, mitkä tunnisteet viittaavat eri objekteihin ja mitkä samoihin. Jos kannassa on esimerkiksi 10 tunnistetta henkilöille (:Hanna, :Jussi jne.) ja halutaan niiden todella viittaavan eri henkilöihin, niin eriys joudutaan jokaisen nimiparin osalta erikseen kertomaan `owl:differentFrom` ominaisuudella, ellei asianlaita ole muuten pääteltävissä. Jos joukossa on n nimeä, joudutaan lisäämään $n * (n - i)/2$ kaarta, eli esimerkkimme tapauksessa 45 kaarta. Käytännössä tämä voidaan onneksi tehdä OWL-kielessä yhdellä `owl:allDifferent`-ilmaisulla, jossa luetellaan kaikki pareittain eri ohjekteihin viittaavat tunnisteet. UNA-oletusta käyttämällä kaikki tämä on kuitenkin tarpeetonta, koska silloin henkilötunnisteiden voidaan automaattisesti olettaa tarkoittavan eri henkilöitä.

OWL-kielessä ei tehdä UNA-oletusta, koska yleisessä tapauksessa webissä ei voida olettaa, että eri tahojen luomat erilaiset URI-tunnisteet viittaisivat eri asioihin. Hyvin usein samalle asialle, vaikkapa Jean Sibeliukselle, on käytössä erilaisia tunnisteita. Tällöin yksi tärkeä tehtävä on voida päätellä, viittaavatko kaksi URI-tunnistetta samaan asiaan vai ei, mikä

on mahdollista OWL:n avulla. Toisaalta monissa käytännön sovelluksissa hyvin usein tiedetään asiat nimetyiksi yksikäsitteisesti, jolloin tietojenkäsittelyä merkittävästi yksinkertaistava UNA-oletus voitaisiin tehdä ja OWL mutkistaa tarpeettomasti asioita.

13.2.4 Yhteenveto

Yhteenvetona voidaan todeta, että puhtaissa loogisissa päättelykoneissa, kuten OWL-päättelijät, ei tehdä mitään ylimääräisiä oletuksia, kuten suljetun maailman tai yksikäsitteisten nimien oletusta. Logiikkaohjelmoinnissa molemmat oletukset tehdään. Sovelluksen kehittäjän on ymmärrettävä käyttämänsä päättelykoneen tekemät oletukset ja toimintatapa ja kuvattava tietoa sen mukaisesti, jotta järjestelmä toimisi halutulla tavalla. Kaikissa tapauksissa päättelyn ideana on lisätä tietämyskantaa uusia kolmikoita, mikä rikastaa tietoa semanttisesti ja näkyy sovelluksissa älykkyytenä. Uudet kolmikot voidaan lisätä tietämyskantaa etukäteen joko ennen sovelluksen käyttöä tai niitä voidaan päätellä vasta sovelluksen ajon aikana tarpeen mukaan. On myös mahdollista yhdistää näitä strategioita ja laskea etukäteen vain osa päätelmistä.

13.3 Käyttötapauksia säännöille

Sääntöjärjestelmiä voidaan hyödyntää mm. seuraavilla tavoilla.

- *Sisällön rikastaminen.* Sovelluskohtaisen *arkipäättelyn* (common sense) avulla voidaan automaattisesti rikastaa tietämyskannan sisältöä. Voidaan esimerkiksi päätellä erilaisia sukulaisuussuhteita henkilöiden välillä sukupuun isä- ja äitisuhteiden perusteella

- *Sisällönkuvailun yksinkertaistaminen.* Yksi tyypillinen käyttötapaus on kohteiden metadatan rikastaminen, mikä voi vähentää tarpeetonta sisällönkuvailutyötä. Esimerkiksi kokoelmassa olevan *tuolin* kohdalla on tarpeetonta käyttää laajempaa termiä *huonekalu*, jos tämä on pääteltävissä automaattisesti esineontologian perusteella.

- *Metadataformaattien muunnokset* Säännöillä voidaan toteuttaa metadataformaattien muunnoksia. Jos esimerkiksi tiedetään Dublin Core -metadata perusteella, että maalauksen loi (`dc:creator`) John Lennon Tokiossa, voidaan kolmikkokantaan päätellä CIDOC CRM -mallin mukainen tapahtuma Tokiossa, johon osallistui John Lennon. Tapahtuma voidaan näin lisätä vaikkapa John Lennonin elämäkertaan muiden tapahtumien joukkoon, ja näin yhdistää biografista tietoa ja taidemuseon kokoelmatietoja.

- *Datan yhdistäminen ja linkitys.* Sääntöjen avulla voidaan luontevasti tunnistaa ja yhdistää moninkertaisia samoja tietoja eri lähteistä.

- *Semanttiset suositukset ja niiden selitykset.* Yksinkertainen tapa semanttisten suosituslinkkien luomiseksi on käyttää SPARQL-kyselyitä. Jos käyttäjä esimerkiksi hakee Victor Hugon kirjoittamia teoksia, voidaan hakea suosituksiksi muita saman maan (Ranska) kirjailijoiden teoksia samalta aikakaudelta. Sääntöjä ketjuttamalla voidaan löytää mutkikkaampiakin yhteyksiä. Sääntöjen etuna on, että säännön tai niiden ketjujen logiikka voidaan ilmaista selväsanaisesti *selityksenä* (explanation) sille, miksi kyseinen suositus on annettu. Tätä ideaa on käytetty aiemmin tekoälyn asiantuntijajärjestelmissä. *selitys*

- *Aineistojen projisointi faseteille.* Fasettihaussa sääntöjä voidaan käyttää hakukohteiden projisoimiseen eli "ripustamiseen" hakukategoroihin älykkäästi. Tekninen etu tässä on mahdollisuus erottaa fasettiontologiat loogisesti datasta, jolloin samaa fasettihakukonetta voidaan käyttää eri tavoilla annotoitujen aineistojen haussa.

- *Tietämyksen muodostaminen.* Sääntöjen avulla voidaan muodostaa datasta uudenlaisia rakenteita, esimerkiksi hakea tietojen välisiä assosiaatioketjuja, mikä mahdollistaa *tietämyksen automaattista muodostamista* (knowledge discovery).

Sääntöjä käytettäessä päättelykoneen ei välttämättä tarvitse olla loogisesti *täydellinen* (complete), ts. kyetä johtamaan kaikki loogiset seuraukset. Hyödyllistä voi olla, jos edes osa mahdollisista päätelmistä voidaan tehdä. Päättelykoneen pitäisi kuitenkin olla *oikeellinen* (sound), eli se ei saisi tehdä virheellisiä päätelmiä. *täydellinen* *oikeellinen*

Osa IV

Sovellukset ja tietoinfrastruktuuri

Luku 14

Sovellusten kehittäminen

Tässä luvussa esitellään linkitetyn datan sovellusten kehittämistä, rakennetta ja toimintaa. Keskeinen ero moniin perinteisiin web-jarjestelmiin on sovellusten datalähtöisyys: sovellusten perustana on rajapintojen kautta käytettävä linkitetty data, joka voi olla hajautetettu fyysisesti ympäri maailmaa sijoitetuille palvelimille. Datalähtöisyys ja hajautetun heterogeenisen tiedon linkitys asettaa uusia haasteita organisaatioden sisällöntuotannon prosesseille ja keskinäiselle yhteistyölle: semanttisessa webissä tarvitaan aiempaa enemmän yhteisesti sovittuja pelisääntöjä ja aineistojen avointa jakamista datan linkittämiseksi kustannustehokkaasti yli organisaatiorajojen ja datasiilojen. Luvussa esiteltävän julkaisumallin keskiössä onkin yhteisen semanttisen webin tietoinfrastruktuurin kehittäminen, jota Suomessa on kehitetty vuodesta 2003 alkaen.

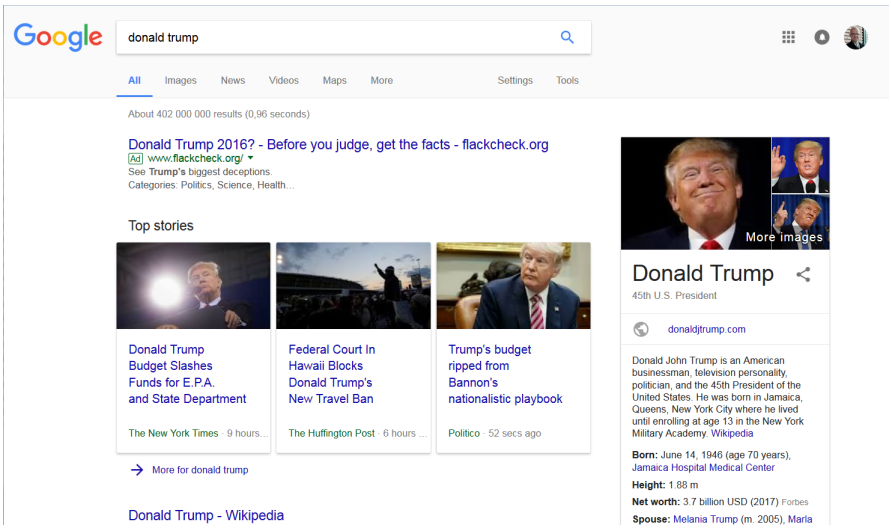
14.1 Yleiset ja alakohtaiset sovellukset

Linkitetyn datan sovellusten kirjo on yhtä laaja kuin webin sovellusten sateenkaari. Esimerkiksi Googlen hakukone hyödyntää semanttisia merkkauksia ja linkitettyä dataa: sisällön tuottajat voivat lisätä WWW-sivuille Schema.org-spesifikaation[1] mukaista semanttista metatietoa sivujen sisällöstä, jotta hakukone voisi paremmin ymmärtää niiden sisältöä. Tämä mahdollistaa mm. *entiteettihakua* (entity search), jossa hakukone

entiteettihaku

[1] http://schema.org

etsii WWW-sivujen sijasta semanttisen webin entitettejä, kuten paikkoja, henkilöitä elokuvia jne., ja voi tarjota ne erikseen osana hakutulosta. Metatieto ja linkitetyn datan tietovarastot mahdollistavat tulosten esittämisen havainnollisessa rakenteellisessa muodossa, johon sisältyy myös linkkejä muuhun hyödylliseen tietoon. Kuvassa 14.1 on esimerkiksi tehty haku sanoilla *donald trump*, ja entiteettihaku on palauttanut kuvauksen presidentti Trumpista oikealla palstalla hänestä kertovien WWW-sivujen ja muiden aineistojen lisäksi (vasemmalla). Kuvauksessa on valmiina linkit mm. hänen syntymäsairaalaansa, kouluihin joita hän kävi sekä linkit lisätietoon Melanie-vaimosta ja lapsista.



Kuva 14.1: Google hyödyntää linkitettyä dataa entiteettihaussa sekä tulosten ja lisätietojen esittämisessä.

Faceook taas käyttää linkitettyä dataa sosiaalisten verkostojen ja niihin liittyvän tiedon esittämiseen. Soveltajat pääsevät käyttämään dataa avoimen Graph API -rajapinnan[2] kautta. Voidaan sanoa, että Googlen ja Facebookin kaltaisten suurten toimijoiden kautta linkitetty data on jo nyt keskeinen globaali webin teknologia; tutkimusten mukaan sellaisten WWW-sivujen osuus, joista löytyy semanttisia merkkauksia, lasketaan jo kymmenissä prosenteissa. Esimerkiksi vuonna 2015 analysoitiin n. 10 miljardin WWW-sivun satunnainen otos, joista n. 31% sisälsi metada-

[2] https://developers.facebook.com/docs/graph-api

taa. Määrä oli kasvanut n. 10%-yksikköä edellisen vuoden vastaavasta testistä.

Globaalin käytön ohella linkitettyä dataa hyödynnetään mitä erilaisimmissa alakohtaissa sovelluksissa. Esimerkkejä linkitetyn datan käyttötapauksista ja sovelluksista on koottu mm. W3C:n sivuille[3] aiheina mm. lääketieteeseen ja terveyteen liittyvät sovellukset, kulttuurialan sovellukset, musiikkiala, teollisuus, paikkatieto ja mobiilijärjestelmät. Havainto linkitetyn datan soveltuvuudesta mitä moninaisempien sovellusalojen käyttöön huomattiin myös Suomessa kansallisessa FinnONTO-hankesarjassa (2003–2013) ja sitä seuranneessa Linked Data Finland -kehitystyössä ja siihen liittyvissä hankkeissa. Näissä on mm. kehitetty kulttuurialan semanttisia portaaleja (MuseoSuomi.fi, Kultuurisampo.fi, Kirjasampo.fi, ja Sotasampo.fi), Terveyden ja hyvinvoinnin laitoksen verkkopalveluita (TerveSuomi.fi), ohjelmoitu demonstraattoreita Wärtsilän voimalaitosten dokumentointiin ja Nokian 5G-verkkojen halintaan, julkaistu Semanttinen Kalevala kansanrunouden saralla, tutkittu valistuksena ajan kirjeenvaihtoa Euroopassa osana laajaa euroopplaista konsortiota, sovellettu teknologiaa kielikorpusten tutkimuksessa sekä käytetty teknologiaa lainsäädännön ja oikeustapausten julkaisemiseen avoimena datana (Semanttinen Finlex). Kaiken tämän työn taustalla ollut laaja-alainen FinnONTO:n visio yhteisen kansallisen tietoinfrastruktuurin kehittämisestä osana kansainvälistä semanttisen webin kehittymistä: erityisesti on keskitytty ontologiseen työhön eri aloilla tarvittavien asiasanastojen, auktoriteettitietojen (henkilöt, organisaatiot), historiallisten tapahtumien, paikkatietojen, biologian taksonomioiden ja lääketieteelisten luokitusten julkaisemikseksi avoimena linkitettynä data sekä yhteentoimivien metatietomallien kehittämiseen.[4]

Seuraavassa tarkastellaan esimerkkinä linkitetyn datan kehittämisestä ja sovelluksista erityisesti semanttisia portaaleja, joiden ideana on kerätä, linkittää ja rikastaa eri lähteistä saatavaa dataa yhteiseksi laajemmaksi palveluksi. Esitys perustuu erityisesti SeCo-tutkimusryhmän tutkimuksissa ja käytännön sovellushankkeissa saatuihin omakohtaisiin kokemuksiin “Sampo”-sarjan portaalien Kulttuurisampo, Kirjasampo, Matkailusampo ja Sotasampo ja muiden sovellusten kehittämisestä, ajatukseen ontologiainfrastruktuurin rakentamisesta ja tarjoamisesta avoimi-

[3] https://www.w3.org/2001/sw/sweo/public/UseCases/
[4] http://journal.fi/tt/article/view/41559

na verkkopalveluina asiakaskäyttöön ONKI/Finto-palveluissa sekä datan julkaisemiseen avoimina verkkopalveluina Linked Data Finland -alustalla.

14.2 Semanttinen portaali: osat ja kokonaisuus

Linkitetyn datan semanttisissa portaaleissa voidaan erottaa kolme pääosaa kuvan 14.2 mukaisesti.

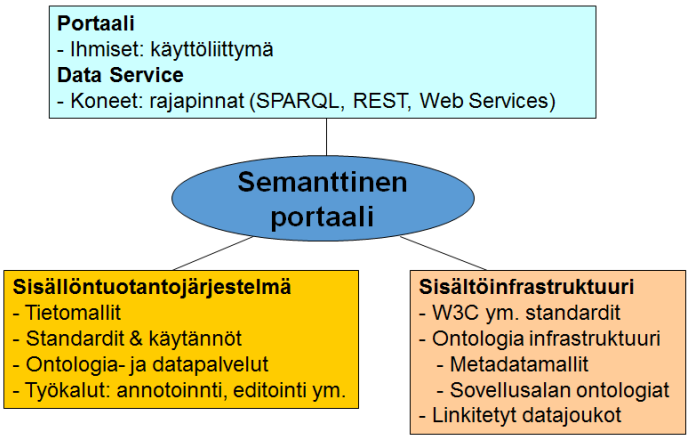
1. Järjestelmän ihmiskäyttäjille datapalvelut tarjoillaan **käyttöliittymän** kautta. Myös koneet voivat käyttää järjestelmän dataa **rajapintojen** kautta. Ihmisille kehitetyt sovellukset toimivat näiden rajapintojen varassa.

2. Käyttöliittymät ja rajapinnat ovat kuitenkin vain jäävuoren huippu kokonaisuudessa, johon kuuluu myös **sisällöntuotannon järjestelmä**. Siinä on sovittu tiedon esittämistavoista ja tavoista tuottaa tietoa. Sisällöntuotannossa hyödynnetään sovellusalueen standardeja, käytäntöjä ja työvälineitä datan tuottamiseksi alkulähteistä.

3. Koko systeemi käyttää **tietoinfrastruktuuria**, joko perustuu toisaalta 1) W3C:n sovellusalasta riippumattomiin web-standardeihin ja toisaalta 2) alakohtaisiin metatietomalleihin ja ontologioihin sekä käytettävissä oleviin muihin datajulkaisuihin.

Sovelluksissa on hyödyllistä erottaa selkeästi datapalvelu varsinaisista sovelluksista, jotka käyttävät dataa yleensä SPARQL-rajapinnan kautta. Ideana on, että saman datan varaan voidaan silloin rakentaa erilaisia sovelluksia dataa muuttamatta. Tämä on kustannustehokasta, sillä dataa on palveluissa tyypillisesti miljoonia kolmikoita, laajimmissa palveluissa jopa miljardeja. Datan muokkaaminen voi siksi olla työlästä ja virhealtista puuhaa.

Kuva 14.3 havainnollistaa tilannetta, jossa on kehitetty erilaisia sovelluksia yhteisen SPARQL-palvelupisteen varaan. Tällaisista sovelluksista, joiden varsinainen toiminallisuus syntyy selaimessa, käytetään nimitystä *Rich Internet Application (RIA)*. RIA-sovellusten etuna on mahdollisuus nopeatoimisten käyttöliittyminen kehittämiseen ilman kokonaisten

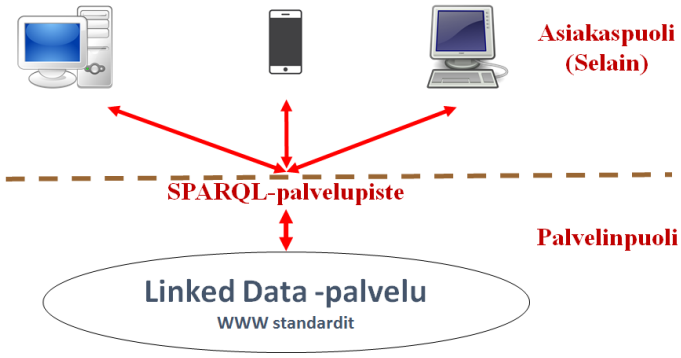


Kuva 14.2: Semanttisen webin portaalin kolme komponenttia

sivujen hidasta latausta palvelimelta (AJAX-tekniikat) sekä laskennan hajauttaminen asiakkaiden selaimille, mikä vähentää palvelimen kuormitusta. Malli soveltuu ohjelmistojen kehittämisessä laajasti käytettyyn *Model-View-Control-arkkitehtuuriin* (MVC), jossa erotetaan käyttöliittymä sovellusalueen tiedosta: *Malli* (model) kuvaa tiedon esittämisen, ylläpidon ja käsittelyn; *näkymä* (view) käyttöliittymän ulkoasun ja *käsittelijä* (controller) toteuttaa käyttäjältä saadut käskyt muuttamalla mallia ja näkymää.

MVC-arkkitehtuuri
malli
näkymä
käsittelijä

Seuraavassa esitellään kuvan 14.2 kompotenteista ensin semanttisen portaalin idea ja ansaintalogiikka, jossa kaikki voittavat. Sitten käsitellään sisällöntuotantoon liittyviä kysymyksiä. Keskitymme tapauksiin, jossa tietoaineistoina voivat olla painetut tekstit ja rakenteista tietoa sisältävät tietovarastot, kuten tietokannat. Myös äänestä, kuvista ja elokuvista voidaan louhia tietoa ja esittää sitä semanttisesti, mutta silloin käytettävät hahmontunnituksen menetelmät poikkeavat olennaisesti tekstin käsittelyssä käytetyistä menetelmistä eikä niitä tässä teoksessa tarkemmin käsitellä. Lopuksi esitellään semanttisen webin sisältöinfrastruktuuriin liittyviä kysymyksiä ja Suomessa tehtyä työtä sekä tarkastellaan tarkemmin portaalisovelluksen kehittämistä käytännön esimerkkinä Sotasampo.fi-palvelu.



Kuva 14.3: SPARQL-datapalvelun rajapinnan varaan voidaan kehittää useita linkitetyn datan sovelluksia kustannustehokkaasti dataa muuttamatta.

Luku 15

Portaalimalli yhteisölliseen julkaisemiseen

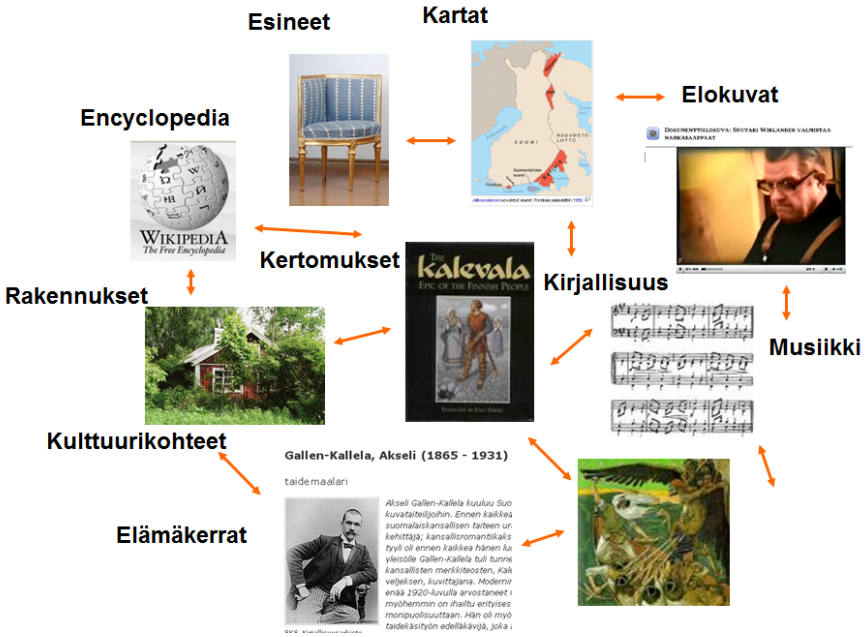
Semanttisen portaalin idean avulla voidaan tarttua yhteisölliseen sisällöntuotantoon ja julkaisemiseen liittyvään keskeiseen haasteeseen: *miten tietoa ja tiedon tuotannon prosesseja voidaan yhdistää yli organisaatiorajojen ja datasiilojen?* Tässä luvussa tarkastellaan esimerkkinä kulttuurialan sisältöjen julkaisemista kansallisella tasolla verkossa. [1]

15.1 Kulttuuriaineistot verkossa

Julkisen sektorin museoilla, kirjastoilla, arkistoilla ja mediaogranisaatiolla kuten YLE on paljolti yhtenäinen kansallinen tehtävä ja tavoite julkaista kulttuuriaineistoja ja palvella asiakkaita nopeasti muuttuvassa digitaalisessa toimintaympäristössä. Eri ogranisaatioiden toisiinsa eri tavoin liittyvien sisältöjen saaminen asiakkaiden ja tutkijoiden käyttöön joustavasti ja kustannustehokkaasti verkon välityksellä on kaikkien yhteinen tahtotila, niin tiedon julkaisijoiden kuin asiakkaidenkin. Lisäksi yhteistyöllä voitaisiin vähentää kustannuksia ja saada käyttöön kriittistä IT-osaamista ja resursseja, joiden hankkiminen on iso haaste erityisesti pienemmille organisaatioille. Samalla voitaisiin selkeyttää organisaatioiden työnjakoa, eliminoida tarpeetonta päällekkäistä työtä ja kohdistaa

[1] Kulttuuriaineistojen julkaisemista ja käyttöä semanttisessa webissä sekä tässä luvussa esitettävää mallia on esitelty tarkemmin teoksessa [21].

eri ogranisaatioiden niukkoja reursseja optimaalisemmin niiden omille osaamisalueille.



Kuva 15.1: Kulttuurisisällöt ovat hetorogeenisiä ja linkittyvät toisiinsa monin tavoin.

Tähän visioon sisältyy kuitenkin isoja haasteita liittyen sisältöjen yhteentoimivuuteen ja organisaatioiden toimintaan. Kultuurialan sisällöt ovat formaateiltaan monimuotoisia, niissä käytetään eri kieliä, tiedot on kuvattu erilaisilla metadatamalleilla ja käytössä on monia luettelointitapoja. Semanttisena haasteena on, että erilaiset sisällöt liittyviä toisiinsa sisällöllisesti mutkikkailla tavoilla. Tilannetta on havainnollistettu kuvassa 15.1. Esimerkiksi henkilön tekstimuotoinen elämäkerta sisältää viittauksia paikkoihin, sukulaisiin ja kollegoihin, tapahtumiin sekä tieteellisen ja taiteellisen työn tuloksiin, joista kaikista on tyypillisesti lisätietoa olemassa jossain muualla, mutta eri muodossa.

Lisäksi tietojen kohteet identifioidaan eri tavoilla eri maissa, organisaatioissa ja kielissä. Jo pelkästään samojen henkilöiden tunnistaminen nimien perusteella eri tietokannoissa on osoittautunut kirjastoissa ja museoissa vaikeaksi haasteeksi. Kuvassa 15.2 on esimerkkinä listattu

Kuva 15.2: Pjotr Iljitš Tšaikovskin tunniste DBpediassa ja kirjoitusasuja eri kielillä Getty-säätiön Union List of Artis Names -ontologiassa

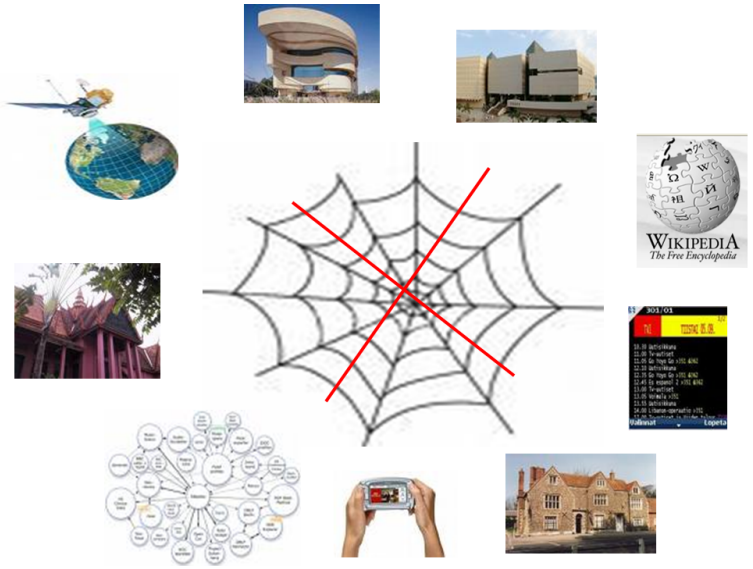
Getty-säätiön ULAN rekisterissä käytettyjä erikielisiä kirjoitusasuja Pjotr Iljitš Tšaikovskista, ja lisäksi tulevat vielä erilaiset lyhenneilmaisut. Lisäksi henkilöt voivat käyttää pseudonimiä, lempinimiä ja nimet voivat muuttua vaikkapa avioliiton seurauksena.

Tunnistejärjestelmien ja tietomallien sekamelska on seurasta tavasta, jolla kulttuurisisältöjä tuotetaan kuvan 15.3 havainnollistamalla tavalla toisistaan riippumatta. Tiedon esitysmuotoja ja luettelointikäytäntöjä on pyritty yhtenäistämään mm. erilaisten standardointihankkeiden avulla, mutta kokonaisuutena tässä riittää vielä paljon tehtävää.

15.2 Hajautettu haku vai hajautetun tiedon aggregointi

Eri toimijoiden tuottamia sisältöjä voidaan yhdistää kahdella eri strategialla.

1. **Hajautettu haku**. Data voidaan yhdistää tiedon hakuvaiheessa



Kuva 15.3: Kulttuurialan sisältöjä tuottavat eri toimijat yleensä toisistaan riipumatta ilman yhteistyötä.

hakukoneen toimesta eri tietokantoihin tehtyjen kyselyiden perusteella.

2. **Tiedon aggregointi**. Data voidaan yhdistää ennen tiedon hakua ja tehdä haku yhteen harmonisoituun tietomassaan.

hajautettu haku

Hajautetussa haussa (federated search), josta käytetään myös nimityksiä *metahaku* (metasearch) ja *monihaku* (multi-search), haku suoritetaan seuraavalla tavalla:

1. Hakukone muuntaa kyselyn eri tietokannoille sopiviin muotoihin ja lähettää kyselyt niille.

2. Hakukone kokoaa tulokset.

3. Tulokset yhdistetään ja mahdolliset päällekkäisyydet vastauksissa minimoidaan.

4. Tulokset esitetään loppukäyttäjälle.

Menettelyn etuna on, että tietokantoja ei tarvitse yhdistää, vaan ainostaan sopia yhtenäisestä tavasta esittää kyselyitä ja hakutuloksia. Mekanismi on erityisen sopiva tilanteisiin, jossa hakukohteet ovat samanlaisia, kuten vaikkapa teokset eri kirjastojen kokoelmissa. Hajautettu kysely voidaan suorittaa kahdella tavalla: *Local as View* (LAV) -strategiassa kukin paikallinen tietokanta tarjoaa oman paikallisen näkymän tietokantaan, johon kysely pitää muuntaa. *Global as View* (GAV) -strategiassa taas sama globaali näkymä jaetaan paikallisille tietokannoille vastattavaksi. Muunosprosesseissa tarvittavia ohjalmistoja kutsutaan *kääreiksi* (wrapper).

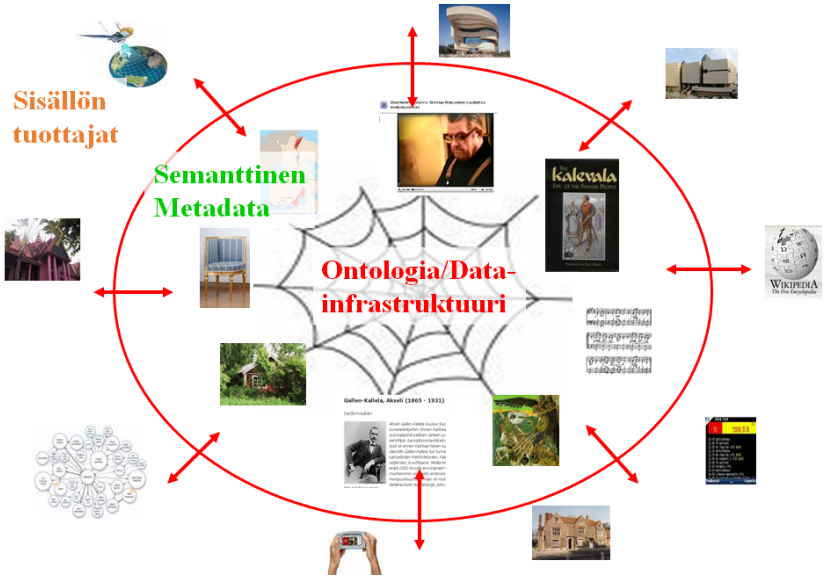
kääre

Yksi periaatteellinen rajoite federoidussa haussa on, että koska kyselyt tehdään paikallisesti toisistaan riippumatta, ei ole helppoa löytää tietojen välisiä globaaleja riippuvuuksia, mikä on yksi semanttisen webin lupauksista. Federoitu haku ei myöskään ratkaise tietojen yhteentoimivuuteen liittyviä kysymyksiä. Jos kahdessa paikallisessa kirjaston tietokannassa esimerkiksi käytetään eri nimimuotoja kirjailijoista, on saman kirjailijan teosten löytäminen globaalisti eri tietokannoista haasteellista.

Ratkaisumalli näihin ongelmiin on tiedon aggregointi eri lähteistä ja haun kohdistaminen näin aikaan saatuun yhteen harmonisoituun tietokantaan. Semanttinen web tarjoaa tässä lähestymistavassa uudenlaisen tavan tarttua samalla kertaa sekä sisällöllisen yhteentoimivuuden että hajautettun sisällöntuotannon haasteisiin kuvan 15.4 havainnollistamalla tavalla.

Tämän ratkaisumallin ytimessä ja kuvan keskellä on ajatus jaetusta tietoinfratruktuurista, jonka avulla eri tahoilla tuotettu tieto saadaan luotua jo sisällön alkuperäisen tuottajan toimesta luetteloinnin ja sisällönkuvailun yhteydessä yhteentoimivassa muodossa. Tällöin yhteisöllinen datan yhdistäminen laajemmaksi kokonaisuudeksi voi tapahtua automaattisesti. Esimerkiksi Ateneumin taidemuseosta saatava tieto Akseli Gallen-Kallelan maalauksesta *Sammon ryöstö* yhdistyy automaattisesti taiteilijan elämäkertaan Suomalaisen Kirjallisuuden Seuran kansallisbiogragiassa ja sen tapahtumiin. Samoin voi syntyä yhteys Kalevalan runoon, josta maalaus sai innoituksensa, Kalevala teemaan liittyvään Sibeliuksen musiikkiin, kirjastoissa oleviin teoksiin kultakauden taiteesta ja Wikipediassa (DBpedia ja Wikidata) oleviin näitä asioita taustoittaviin artikkeleihin ja tietoihin.

Tämän vision toteuttaminen alkoi Suomessa vuonna 2002 MuseoSuomihankeessa, jossa kehitettiin pilottihankkeena semanttisen webin tekno-

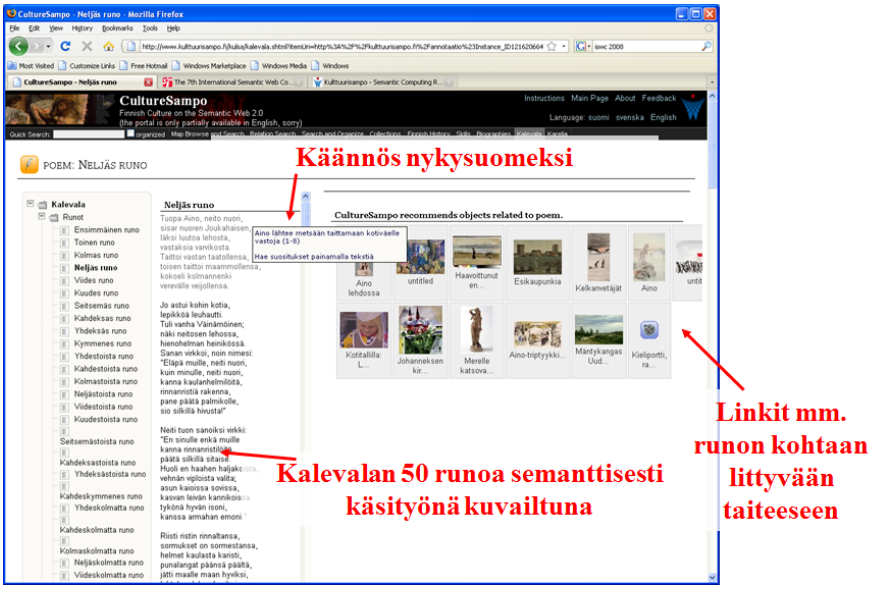


Kuva 15.4: Sisällöntuotannon hajautettu Sampo-malli, jossa tieto yhdistyy jaetun tietoinfrastruktuurin kautta

logialla MuseoSuomi-palvelu (2004). Siinä yhdistettiin Kansallismuseon, Espoon kaupunginmuseon ja Lahden kaupunginmuseon kokoelmatietoja semanttiseksi verkoksi, jonka varaan kehitettiin uudenlainen semanttinen hakukone ja aineistojen suosittelujärjestelmä (semanttinen linkittäjä) logiikkaohjelmointiin perustuen. Pilotti oli ensimmäinen laatuaan maailmassa, sillä semanttisen webin visio oli tuolloin vielä nuori. Käytettävissä ei tuolloin ollut työssä tarvittavia ontologioita, mikä johti ajatukseen luoda kansallisen tason yhteinen semanttisen webin tietoinfrastruktuuri seuraavia projekteja avittamaan. Työ käynnistyi 2003 FinnONTO-hankkeena, joka laajeni sittemmin kymmenvuotiseksi FinnONTO-projektien sarjaksi. Työ jatkuu vieläkin uusina Linked Data Finland -datajulkaisualustaan liittyvinä hankkeina.

Kuvan 15.4 Sampo-malliksi ristittyä konseptia on testattu käytännössä useissa Sampo-järjestelmissä:

- *Kulttuurisampo.fi – Suomalainen kulttuuri semanttisessa webissä* julkaisiin vuonna 2008. Se yhdistää parinkymmenen eri sisällöntuottajan sisältöjä. Siinä datan integraatio perustuu joukkoon jaet-



Kuva 15.5: Kulttuurisammon Semanttinen Kalevala, jossa Kalevalan tarinan tapahtumat, henkilöt ja paikat mallinnettiin ja esitettiin linkitettynä datana ja yhdistettiin automaattissti mm. Kalevala-aiheiseen taiteeseen Ateneumin kokoelmissa ja videoihin YLE:n Elävässä arkistossa.

tuja ontologiota, kuten kotimaisista asiasanastoista luotu kymmenien tuhansien käsitteiden KOKO-ontologia ja aineistojen perusteella rakennetut paikka- ja henkilöontologiat. Metadatamallit on yhdistetty Dublin Core -standardin ja sen dumb down -periaatteen avulla. Yhtenä ideana oli käyttää Kalevalaa ja siitä luotua ontologiaa kultakautemme kulttuuriaineistoja yhdistävänä "semanttisena liimana". Datapalvelun varaan toteutettiin semanttinen haku ja yhdeksän erilaista sovellusnäkymää, kuten Semanttien Kalevala, jota havainollistettu kuvan 15.5 kuvankaappauksella sovelluksen käyttöliittymästä. Kulttuurisampo-demonstraattorin semanttisessa verkossa on n. 11,4 miljoonaa tietojen välistä yhteyttä (kolmikkoa). Järjestelmää ei ole olennaisesti päivitetty vuoden 2008 jälkeen, mutta se on edelleen verkossa.

- *Kirjasampo – Suomen kaunokirjallisuus semanttisessa webissä* oli alun perin osa Kulttuurisampoa, mutta on nykyisin yleisten kir-

jastojen (Kirjastot.fi) ylläpitämä oma palvelunsa, jolla oli v. 2016 1,6 miljoonaa kävijää. Järjestelmän metadata kattaa käytännössä kaiken Suomessa julkaisun kaunokirjallisuuden erittäin rikkaana semanttisena verkkona.

- *Matkailusampo*-pilotissa konseptoitiin kulttuuridatan hyödyntämistä matkailun tukena lähtökohtana paikkatieto ja siihen linkittyvä kulttuuri- ja muu data. Verkossa ollut demonstraattori joutui hakkereiden hyökkäyksen kohteeksi eikä ole enää toiminnassa.

- *Sotasampo – talvi- ja jatkosota semanttisessa webissä* on Sampomallin uusin ikarnaatio. Se yhdistää lukuista eri lähteistä saatavaa sotahistoriaan liittyvää dataa linkitetyksi avoimen datan pilveksi. Tätä kirjoitettaessa Sotasammossa on 12 miljoonaa kolmikkoa, kuten että talvisota päättyi 13.3.1940. Palvelu sisältää mm. metatiedot puolustusvoimien SA-kuva-arkiston 160 000 autenttisesta valokuvasta, Kansallisarkiston tietokannan viime sodissa menehtyneistä 95 000 sotilaista ja tuhansista muista, 35 000 historiallisen paikan ontologian luovutetuilta alueilta vanhoine karttoineen, kymmeniä tuhansia sotapäiväkirjoja ym. Palvelun ytimessä on sota-ajan tapahtumista luotu sotahistoriallinen tapahtumaontologia. Sotasammon tiedot on mallinnettu, harmonisoitu ja yhdistetty laajentaen CIDOC CRM -ontologiaa, jota esitelty tarkemmin luvussa 8.3.

15.3 Edut käyttäjille

Edellä esitetty semanttisen portaalin Sampo-malli tarjoaa loppukäytäjälle monia etua:

- *Yksi globaali näkymä hajautettuihin aineistoihin.* Sisältöihin pääsee käsiksi yhden yhtenäisen käyttöliittymän kautta tarvitsematta opiskella yhdistettyjen tietokantojen erilaisia käyttöliittymiä.

- *Automattinen sisällön aggregointi.* Eri lähteissä oleva tieto kyselyyn liittyen voidaan yhdistää automaattisesti loppukäytäjälle yhdeksi vastaukseksi.

- *Semanttinen haku.* Haku voidaan tehdä älykkäästi semanttisena hakuna.

- *Semanttinen selailu.* Haun ohella voidaan käyttäjälle tarjota tuloksiin linkittyviä muita tietoja semanttisten suosittelulinkkien avulla.
- *Muut palvelut.* Semanttisen datapalvelun päälle voidaan helposti kehittää muitakin palveluita, kuten data-analyysin työkaluja ja visualisointeja.

15.4 Edut julkaisijoille

Portaalimalli on edullinen myös datan julkaisijoiden kannalta:

- *Hajautettu sisällöntuotanto.* Malli mahdollistaa työn hajauttamisen eri tahoille.
- *Automaattinen linkitys.* Työlästä tietojen ylläpitoa kuten linkitystä voidaan automatisoida.
- *Jaettu julkaisukanava.* Eri organisaatiot voivat hyödyntää samaa julkaisukanavaa, jolloin tarve omalle kehitystyölle vähenee.
- *Sisältöjen rikastaminen.* Kaikkien mukana olevien sisällöntuottajien aineistot rikastuvat automaattisesti ja ilmaiseksi toisensa kautta.
- *Aggregoidun sisällön käyttö uudelleen.* Portaalin sisältöä voidaan käyttää helposti uudelleen rajapintojen kautta tai dataa kopioimalla uusissa sovelluksissa, myös ulkoisten toimijoiden kautta.

15.5 Uusia haasteita

Uuden mallin käyttöönottossa on kuitenkin myös omat vaikeutensa.

Aineistojen semanttinen yhteentoimivuus edellyttää ogranisaatioilta aiempaa enemmän yhteistyötä ja sopimista yhteisistä tietomalleista, tunnisteista ja sisällön tuotannon käytännöistä. Tämä ongelma on toisaalta organisatoorinen ja sosiaalinen, sillä perinteisten mallien ja toimintatapojen muuttaminen ei välttämättä ole helppoa. Toisaalta ongelmat voivat

olla teknisiä ja käytännöllisiä. Jos esimerkiksi käytössä olevat tietojärjestelmät eivät tue semanttisen tiedon esittämistapoja ja uusia työnkulkuja. Eteenpäin ei ehkä päästä ilman isompia tietojärjestelmäremontteja, mikä hidastaa kehitystä, vaikka tahtoa uusien menetelmien käyttöönottoon olisikin.

Haasteita tarjoaa myös älykkäiden sovellusten kehittäminen semanttisen datan avulla. Työ vaatii uusien menetelmien ja työkalujen osaamista, mitä organisaatiossa ei välttämättä ole tarjolla. Ala on myös nuori ja tarjolla olevissa työkaluissa on usein puutteita ja niitä pitää kehittää itse. Myös käytettävissä olevan linkitetyn datan laadussa, kuten vaikkapa DBpediassa, on vielä paljon parannettavaa.

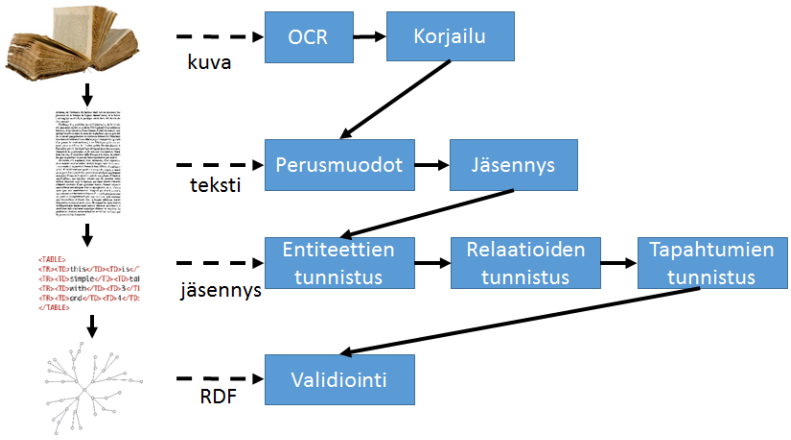
On kuitenkin selvää, että haasteista riippumatta linkitetyn datan teknologia on se suunta, johon kulttuuridatan julkaisemisessa ollaan menossa.

Luku 16

Sisällöntuotanto

Tässä luvussa luodaan katsaus RDF-muotoisen metadatan tuotantoon ja sen työvaiheisiin. Lähtökohtana on tekstimuotoinen data. Se kuvailee sovelluksen kohteena olevia asioita tai tietosisältöjä, jotka voi liittyä esineisiin, kirjoihin, äänitteisiin, filmeihin, tapahtumiin tms. Kuvaus voi olla rakenteetonta tekstiä, puolirakenteista dataa kuten WWW-sivuja, tai olla jo rakenteisessa muodossa esimerkiksi taulukkona tai tietokantadumppina. Haasteena on muodostaa tällaisesta tekstuaalisesta kuvauksesta rakenteista RDF-muotoista dataa, joka on linkitetty sisäisesti, ja jota on rikastettu linkeillä muihin tietoaineistoihin. Esimerkkinä käytetään Sotasampo.fi portaalin aineistoja ja niistä tuotettua dataa.

Kuva 16.1 havainnollistaa prosessia, joka tarvitaan fyysisen tekstidokumentin kuten kirjan sisällön muuttamiseksi ensin tekstiksi, joka voidaan sitten analysoida ja muuntaa kieliteknologian avulla tavalla rakenteiseksi dokumentiksi ja lopulta linkitetyksi dataksi. Esimerkiksi tekstissa olevat sanat voidaan yhdistää niille merkityksen antaviin ontologisiin käsitteisiin ja tunnistaa sekä esittää laajempia merkitysrakenteita semanttisina RDF-verkkoina. Automaattisesti tuotetun lopputuloksen laatua on lisäksi syytä tarkistaa dataa validoimalla. Seuraavassa tarkastellaan näitä vaiheita hiemen tarkemmin.



Kuva 16.1: Sisällöntuotannon vaiheita

16.1 Tekstin tunnistaminen (OCR)

Monessa tapauksessa tietosisältö on saatavilla vain painetussa muodossa, esimerkiksi kirjana. Sotasammossa tällaisia aineistoja ovat mm. Kansa Taisteli -lehden PDF-muodossa olevat vuosikerrat sekä Kansallisarkistossa olevat koneella kirjoitetut joukko-osastokortit.

OCR-tunnistus

Silloin datan muodostamisen ensimmäisenä vaiheena on tekstin tunnistus *OCR-tunnistuksella* (Optical Character Recognition), jossa dokumentti digitoidaan kuvaamalla ja siinä olevat tekstit käsitellän hahmontunnistuksen menetelmien avulla. Tuloksena on tyypillisesti dokumentin tekstit sivukohtaisesti erotettuina. Joissain tapauksissa myös kuvia voidaan erottaa.

Tekstin irroitukseen[1] on saatavilla runsaasti työkaluja, kuten esimerkiksi avoimen koordin Tesseract[2] ja kaupallinen Abbyy FineReader[3], ja moniin yleistyökaluihin, kuten PDF-dokumenttien luomiseen tarkoitettuun Adobe Reader -ohjelmaan, sisältyy OCR-mahdollisuus. Kaikkien OCR-työkaluen haasteena on, että OCR-prosessi tuottaa enemmän tai vähemmän virheellisiä tulkintoja painetun tekstin ja algoritmien laadusta riippuen. Monet tällaiset virheet ovat systemaattisia ja niitä voidaan korjata

[1] Aihetta käsitellään esimerkiksi teoksessa [40].
[2] https://github.com/tesseract-ocr
[3] https://www.abbyy.com/en-eu/finereader/

esimerkiksi aineistokohtaisilla säännöillä, joissa virheelliset tekstihahmot korvataan virheettomillä. Haasteena tässä on varmistaa, että muunnokset kohdistuvat vain todellisiin virheisiin eivätkä vanhingossa korjaa oikein tulkittuja tekstikohtia vääräksi.

Esimerkiksi Sotasammossa käytettiin tekstien irrottamiseen digitoituista Kansa Taisteli -lehdistä Abbyy-työkalua. OCR-virheiden korjaamiseen luotiin n. 160 erilaista korjaussääntöä, jotka toteutettiin säännöllisten lausekkeiden avulla.[4]

Toisena OCR-tekniikan haasteena on yhdellä sivulla mahdollisesti olevien eri tekstien, sivunumeroiden ja otsakkeiden, kuvatekstien, mainosten yms. erottaminen toisistaan.

Korjauksia voidaan periaatteessa tehdä käsityönä, mutta esimerkiksi tuhansien tai miljoonien artikkeleiden tarkastaminen ja korjaaminen käsin ei ole mahdollista vaan on luotettava koneen tekemiin päätöksiin ja pistokokeina tehtyihin tarkistuksiin. Viimekädessä on yleensä hyväksyttävä se tosiasia, että automaattisesti käsitellyissä aineistossa on yleensä enemmän virheitä kuin käsityönä tehdyissä.

16.2 Datamuunnokset ja linkitys

Tekstimuotoiseen dataan sisältyy usein säännönmukaisuuksia, kuten luetteloita ja taulukoita, joita voidaan käyttää hyväksi datan rakenteistamisessa. Myös tällaisten rakenteiden tunnistamisessa ja rakenteiden tuottamisessa voidaan käyttää säännöllisiä lausekkeita ja muunnossääntöjä. Esimerkiksi Suomalaisen Kirjallisuuden Seuran (SKS) julkaisemasta Kansallisbiografiasta on tuotettu Semanttinen Kansallisbiografia tällaisten muunnosten avulla: elämäkertojen loppuun on listattu melko määrämuotoisella tavalla tietoa tekstin kohteena olevan henkilön elämänkaaren keskeisistä tapahtumista, joiden rakenteistaminen on huomattavasti helpompaa kuin vapaamuotoisen tekstin.[5]

Jos teksti on vapaamuotoista ja rakenteetonta, voidaan siitä tuottaa merkityksiä *automaattisen annotoinnin* (automatic annotation)

[4] Työtä on kuvattu tarkemmin viitteessä [48].

[5] Semanttinen kansallisbiografia -hanketta on esitelty tarkemmin sivulla http://seco.cs.aalto.fi/projects/biographies/, josta löytyy myös aiheeseen liittyviä julkaisuja.

automaattinen annotointi

menetelmillä. Automaattisen annotoinnin tutkimus kuuluu *tietämyksen irroittamisen* (knowledge extraction alaan, jossa tutkitaan sekä rakenteettoman että rakenteisen datan muuntamista semanttiseen muotoon.[6]

tietämyksen irroitus

Annotoinnin tavoitteena on löytää tekstistä sen merkitystä kuvaavia termejä (esimerkiksi henkilöitä, paikkoja, teoksia jne.), mahdollisesti näiden välisiä suhteita (esimerkiksi että tietty henkilö on tietyn teoksen kirjoittaja) ja laajempia tapahtumia (esimerkiksi että henkilö sävelsi teoksen tietyyn aikaan tietyssä paikassa).

Automaatinen annotointi alkaa taivutusmuodot normalisoivalla morfologisella analyysilla, jonka avulla teksti perusmuotoistetaan. Tämän jälkeen voidaan tunnistaa teksissä sitä kuvailevia, siinä esiintyviä nimiä (Named

NER

Entiry Recognition, NER) ja avainsanoja. Jos tunnistetut käsitteet pystytään yhdistämään järjestelmän taustalla oleviin ontologioihin, voidaan tekstiä ja dokumentin metatietoja alkaa linkittää semanttisesti ja rikastaa niitä muiden aineistojen avulla.

16.3 Merkitysten erottelu

Haasteena kuitenkin on taivutusmuotojen ja sanojen merkisten homonymia. Esimerkiksi onko sanan “alusta” perusmuoto “alku”, “alusta”, “alustaa”, vai “alus”. Lisäksi sama termi voi merkitä erikseen tarkasteluna montaa eri asiaa, esimerkiksi “kuusi” lukumäärää ja puulajia tai “Varkaus” kaupunkia tai tapahtumaa. Eri merkitysten tunnistaminen onnistuu ihmiseltä sanojen käyttöympäristön (kontekstin) ja erilaisten arkijärjen heuristiikkojen avulla. Esimerkiksi Pariisista puhuttaessa tarkoitetaan oletusarvoisesti Ranskan pääkaupunkia, mutta jos puheena on Wim Wendersin elämä, tilanne on toinen ohjaajan vuonna 1984 valmistuneesta elokuvasta “Paris, Texas” johtuen. Lisäksi paikannimi Paris nimi voi mennä sekaisin kuuluisan miljoonaperijättären Paris Hilton nimen kanssa. Kauhuskenaariona on teksti, jossa Paris Hilton katsoo Pariisin Hiltonissa elokuvaa Paris, Texas.

semanttinen disambiguointi

Merkitysten erotteluun eli *semanttiseen disambiguointiin* (semantic disambiguation) on kehitetty paljon menetelmiä ja työkaluja erityisesti

[6] Alueella kehitettyjä lukuisia työkaluja on lueteltu mm. Wikipediassa https://en.wikipedia.org/wiki/Knowledge_extraction.

englannin kielelle, kuten vaikkapa DBpedia Spolight[7] DBpedian käsitteiden tunnistamiseen tekstistä. Suomen kielelle ne eivät yleensä sovellu.[8]

Tekstien automaattista sisällönkuvailua (automaattista annotointia) tarvitaan, vaikka data olisi saatavilla rakenteisessa muodossa, sillä metadatan elementtien arvot ovat yleensä tekstimuotoisia. Esimerkiksi Sotasammon kuvan tekstissä saattaa lukea, että kuvassa "kapteeni Korhonen laskee mäkeä" tai että kuvan on ottanut "Ojanen". Se, kenestä rintamalla olleista sadoista korhosista tai ojasista on kyse, ei välttämättä ole helppoa.

Toinen automaattiseen annotontiin liittyvä haaste on, että tekstiä kuvaavaa asiasanaa ei tekstissä välttämättä ole erikseen kirjoitettu, jos se asiayhteydestä muutenkin selviää. Mailasta ja kiekosta puhuttaessa ei tarvitse välttämättä tarvitse kertoa, että kyse on jääkiekosta. Tällaisten implisiittisten käsitteiden löytämiseen voidaan käyttää taustalla olevien ontologioiden yhteyksiä, jolloin puhutaan dokumentien laajentamisesta. Erimerkiksi Suomesta kertova artikkeli voi kertoa jotain myös Pohjoismaista. Laajennus voi parantaa tiedon löydettävyyttä, vaikka se ei aina pitäisikään paikkaansa. Esimerkiksi Pohjoismaat-termillä tapahtuvan tiedonhaun kohteena voi olla vain Pohjoismaat kokonaisuutena, eikä Pohjoismaihin kuuluvat valtiot erikseen, jolloin haun tarkkuus huononee, vaikka saanti paranisikin.

Tekstien aiheiden löytämiseen voidaan käyttää myös tilastollista tiedonlouhintaa, jossa samantyyppisten tekstien voidaan olettaa puhuva samoista asioista, vaikka joissakin niistä eivät kaikki yhteiset termit esiintyisikään. Esimerkki tällaisesta tekniikasta on *aiheiden tunnistus* (topic modeling).[9]

aiheiden tunnistus

RDF-datan tuotannon kannalta ongelmattominta tietoa on data, jonka kuvailussa on käytetty standardoituja muotoja tai jopa yksilöiviä tunnisteita, esimerkiksi että henkilönimet on aina kirjoitettu tietyllä tavalla tai henkilöt yksilöity henkilötunnusten avulla. Käytännössä tällaisissakin datoissa joudutaan usein kuitenkin tekemään datan normalisointia, kun esimerkiksi eri aineistoissa käytettyjä erilaisi käytäntöjä vaikkapa päi-

[7] http://www.dbpedia-spotlight.org/

[8] Sotasammossa käytettiin entiteettien linkitykseen tutkimusryhmässä itse kehitettyä työkalua ARPA [39] ja disambiguointiin erilaisia alakohtaisia heuristiikkoja ja sääntöjä [19].

[9] Johdatus aiheiden tunnistamiseen on esimerkiksi [9].

väysten esittämisessä harmonisoidaan. Historiallisten päiväysten osalta lisäharmia tuottaa eri maissa eri aikoina käytetyt erilaiset kalenterijärjestelmät. Esimerkiksi Englannissa ja Venajällä luovuttiin juliaanisesta ajanlaskusta myöhemmin kuin Italiassa tai Suomessa. Erityisesti XML- ja taulukkomuotoisen datan puhdistamista ja muunnoksia varten on kehitetty monia näppäriä työkaluja, kuten Open Refine[10] ja Karma[11].

16.4 Datan semanttinen validointi

datan validointi

Kun data on saatu teknisesti haluttuun muotoon, on vielä syystä tarkastaa, ettei datassa ole virheitä tai sen muunnoksessa ei ole syntynyt niitä, eli data pitää *validoida* (validate). Metadataskeemoissa esimerkiksi usein määritellään joku tietty elementti pakolliseksi tai kentän arvoille rajoitteita. Datan automaattista validointia varten metadatamalli voidaan määritellä esimerkiksi käyttämällä W3C:n SHACL-suositusta[12].

tietämysperustainen validointi

Teknisen validoinnin ohella dataa on mahdollista validoida myös *tietämysperustaisesti* sovellusalueen tietämyksen avulla (knowdege based validation). Esimerkiksi Sotasammossa havaittiin, että järjestelmään sisältyneessä, viime sodissa kaatuneiden tietokannassa Menehtyneet 1939–1945 oli henkilöitä, jotka olivat haavoittuneet kuolemansa jälkeen alkuperäisessä datassa olevasta kirjoitus- tai muusta virheestä johtuen.

Isojen aineistojen osalta ei ole mahdollista tarkistaa kaikkia automaattisti tuotettuja linkityksiä ja rakenteita. Voidaan kuitenkin tehdä pistokokeita ja arvioida tällä tavalla linkityksen onnistumista laajemminkin. Esimerkiksi Sotasammon kuvien automaattisessa annotoinnissa valitettiin satunnaisesti pieni otos kuvia, jonka linkitykset tarkistettiin käsin. Tuloksen mukaan aannotointinnassa onnistuttiin n. 75% tarkkuudella, mitä voidaan pitää kohtuullisena tuloksena [19].

[10] http://openrefine.org/
[11] http://usc-isi-i2.github.io/karma/
[12] https://www.w3.org/TR/shacl/

Luku 17

Semanttisen webin tietoinfrastruktuurit

W3C:n standardit eivät ota kantaa ontologioiden tai metadatan varsinaiseen sisältöön, vaan ovat luonteeltaan sovellusriippumattomia ja perustuvat logiikkaan. Esimerkiksi `rdfs:subClassOf`-suhde mahdollistaa leijonan ja kissaeläimen välisen yleisen yläluokkasuhteen ilmaisemisen, mutta varsinaiseen eläinlajien taksonomian muodostamiseen ei oteta kantaa. Datan mallinnus- ja ontologiatyö jää kunkin alan asiantuntijatahojen ja sovellusten kehittäjien tehtäväksi. Tässä luvussa esitellään periaatteita, malleja ja verkkopalveluita, joilla W3C:n standardeja voidaan laajentaa kohti sovellusalakohtaista infrastruktuuria, joka muodostaa esimerkiksi Sampo-mallissa perustan tietojen yhteentoimivuudelle ja linkitykselle. Infrastruktuuria voidaan kehittää vielä laaja-alaisemminkin kohti monialaista kansallista tietoinfrastuktuuria ja luoda palveluita, joilla se voidaan ottaa käyttöön verkon välityksellä tietojärjestelmissä.

17.1 Tietoinfrastruktuurin osat

Ajatusta kansallisesta semanttisen webin ontologiainfrastruktuurista voi verrata tie-, sähkö- ja puhelinverkkojen muodostamiin perinteisiin infrastruktuureihin. Semanttisten käsitteiden verkosto on luonnollisesti abstrakti ja näkymätön, mutta mahdollistaa hieman vastaavaan tapaan yh-

teyksiä kuin vaikkapa tie- tai puhelinverkko.[1] Visiona on, että jos yhteisen infrastruktuurin pelisäännöillä tuotettu sisältö julkaistaan verkossa, voidaan se kytkeä ja hyödyntää automaattisesti muiden toimijoiden semanttisissa verkoissa hieman vastaavaan tapaan kuin uusi maantienpätkä tieverkoston osana. Etuna on, että uuden sisällön arvo rikastuu ”ilmaiseksi” infrastruktuurin avulla verkon muusta sisällöstä vastaavasti kuin uuden tieosuuden arvo syntyy sen kytkeytymisestä muihin teihin. Samalla myös muiden verkon julkaisijoiden sisältöjen arvo rikastuu uuden tiedonsirpaleen avulla vastaavasti kuin uusi tie parantaa muiden teiden keskinäistä saavutettavuutta. Sekä itse infrastruktuuri että siinä jo oleva tieto voidaan hyödyntää toisissa sovelluksissa, mikä säästää merkittävästi järjestelmien kehityskustannuksia.

Tietoinfrastruktuurissa voidaan erottaa seuraavia osia:

1. Eri alat kattavat **sovellusriippumattomat W3C:n semanttisen webin standardit** (RDF, RDFS, SPARQL, SKOS, OWL, RIF ym.).

2. **Metadatamallit** ja niiden yhteentoimiva kokonaisuus. Käytössä on esimerkiksi tässä teoksessa aiemmin kuvatut Dublin Core laajennuksineen ja dumb down -periaate sekä datan harmonisointimallit, kuten CIDOC CRM ja FRBR.

3. **Sovellusaluekohtaiset aiheontologiat** (domain ontology). Joukko toisiinsa linkitettyjä, W3C:n standardeihin ja kansalliseen käsitteistöön perustuvia ontologioita, joilla on linkkejä myös kansainvälisiin sanastoihin ja tietosisältöihin. Käsiteontologioiden resursseja käytetään metadatamalleissa tietojen kuvailussa.

aiheonotologia

Semanttisen webin standardeja on esitelty jo tämän teoksen muissa luvuissa. Työtä koordinoidaan kansainvälisesti W3C:n toimesta aktiivisesti. Metatietomallien osalta (ks. luku 8) työtä tehdään lukuisissa kansainvälisissä yhteisöissä kuten museoalan ICOM (International Council of Museums) ja kirjastojen IFLA (International Federation of Library Associations and Institutions). Lisäksi lukuisilla muillakin aloilla ja yhteisöillä on omia vastaavantyyppisiä datamalleja ja käytäntöjä vaikkapa potilastietoja tai luontohavaintoja varten.

[1] Tätä ajatusta on kehitelty viitteessä [26].

Samoin työ edistyy aiheontologioiden kehittämiseksi. Jo Carl von Linnén (1707–1778) ajoista lähtien esimerkiksi biologit ovat ryhmitelleet kasveja ja eläimiä hierarkkisiksi taksonomioiksi, elämän puuksi, ja lajien kehitystä aika-akselilla kuvaaviksi fylogeneettisiksi evoluutiopuiksi. Lääketieteen saralla on kehitetty laajoja ontologioita mm. anatomiaan, lääkeaineisiin ja tautiluokituksiin liittyen. Esimerkkejä erilaisista luokituksista ja sanastoista löytyy kaikilta aloilta. Tarkastelemme seuraavassa hieman tarkemmin kulttuurisisältöjen esittämisessä tarvittavia keskeisimpiä ontologioita. Näitä voidaan ryhmitellä seuraavasti:

1. **Yleiskäsiteontologiat**. Nämä ontologiat vastaavat käsitteistöltään karkeasti nykyisiä asiasanastoja ja tesauruksia (ilman vapaan indeksoinnin termejä), kuten YSA, Museoalan asiasanasto MASA tai Getty-säätiön Art and Architecture Thesaurus (AAT), jossa on yli 51 000 käsitettä ja 269 000 termiä.

2. **Toimijaontologiat**. Toimijaontologiat ovat henkilö- ja organisaatiorekistereitä, ja muistuttavat kirjastoissa käytettyjä ns. auktoriteettitietokantoja. Toimijaontologian avulla samannimiset toimijat voidaan yksilöidä ja erottaa toisistaan eri tunnisteilla ja lisätietojen avulla, esimerkiksi henkilökaimat syntymävuoden ja -paikan avulla. Esimerkiksi Getty-säätiön Union List of Artist Names (ULAN) -sanastossa on n. 120 000 toimijaa (kuten Akseli Gallen-Kallela), joilla on n. 293 000 erilaista nimeä. Semanttisen kuvailun osalta toimijoita on luokiteltu mm. satojen eri kansallisuus- ja ammattinimikkeiden avulla.

3. **Paikkaontologiat**. Paikkaontologiat vastaavat kansallisten maanmittauslaitosten ylläpitämiä paikannimirekistereitä. Niiden avulla voidaan yksilöidä paikat, sijoittaa ne koordinaatistoon ja tallentaa paikkoihin liittyviä lisätietoja kuten paikkatyyppi (kylä, järvi, asema jne). Historiallisten paikkojen osalta erityisenä haasteena on paikoissa ja kartoilla tapahtuvat muutokset[2].

4. **Aikaontologiat**. Monella alalla keskeistä on ajan esittäminen. Lineaarisen kalenteriajan ohella voidaan viitata myös päivän ja vuodenkierron aikoihin (esimerkiksi aamu, kevät) sekä nimettyihin

[2]Lisätietoa Suomen historiallisten paikkojen ja karttojen kuvaamisesta ontologiana on mm. projektisivulla http://seco.cs.aalto.fi/projects/histoplaces/.

aika- ja tyylikausiin (esimerkiksi rauta-aika, valistuksen aika, art deco). Aikakausien esimerkiksi rautakauden merkitys voi riippua paikasta.

5. **Tapahtumaontologiat**. Tapahtumat, kuten Porvoon valtiopäivät tai Napoleonin kruunajaiset, liittävät toisiinsa henkilöitä, paikkoja, aikoja ja toisia tapahtumia, ja mahdollistavat kulttuuristen sekä provenienssitietoon liittyvien ilmiöiden kuvaamisen semanttisesti yhteentoimivalla tavalla. Tästä johtuen mm. museoalalla standardoidun CIDOC-CRM-järjestelmän perustaksi on valittu tiedon esittäminen tapahtumina, eikä esimerkiksi Dublin Coren dokumenttiperustaista tietomallia.

6. **Nimistöontologiat**. Monilla tieteenaloilla on käytössä laajoja nimistöjä, joita voidaan liittää yleisontologioihin. Tällaisia ovat esimerkiksi biologian eliöiden lajilistat, geologian mineraalit, lääketieteen taudit ja lääkeaineet, kielitieteen kielet yms. Esimerkiksi FinnONTO:n YSO-ontologian yleiskäsite ”linnut” lavenee kattavaksi, yli 11 000 maailman lintulajin taksonomiaksi (lajilistaksi) AVIO-ontologian[3] kautta.

Lisäksi tarvitaan verkkopalveluita, joiden avulla tietoinfrastruktuurin osat voidaan julkaista ja tukea niiden käyttöä asiakkaiden järjestelmissä:

- **Ontologiapalvelut**. Ontologiakirjastopalveluiden kautta ajantasaiset aiheontologiat voidaan ottaa kustannustehokkaasti käyttöön eri organisaatioissa verkkopalveluina. Palveluun voi sisältyä myös yhteisöllinen komponentti ontologian laajentamiseksi käyttäjäkunnan ehdotusten mukaan.

- **Datapalvelut**. Linkitetyn datan palveluiden kautta voidaan julkaista ajantasaista linkittyä dataa ja ottaa sitä kustannustehokkaasti käyttöön rajapintojen avulla.

[3]FinnONTO-hankkeessa kehitettiin Luonnontieteellisen keskusmuseon erilaisista taksonomioista RDF-muotoisia ontologioita, kuten lintulajien AVIO-ontologia [50], nisäkkäiden MAMO-ontologia ja suomalaisten kasvien KASSU. Tätä työtä ja siihen liittyviä sovelluksia on esitelty sivulla http://seco.cs.aalto.fi/ontologies/biology/.

- **Infrastruktuurin ylläpitoprosessi**. Lisäksi tarvitaan systemaattinen ja koordinoitu organisaatio ja mekanismi, joka kantaa vastuun eri aloilla tarvittavien sanastojen ja datan yhteisöllisestä kehittämisestä.

17.2 Ontologiapalvelut

FinnONTO:ssa luotu kansallinen ONKI-ontologiapalvelupilotti julkaistiin verkossa 2008 ja sillä oli pian n. 14 000 ihmiskäyttäjää kuukaudessa. Palvelun rajapintoja rekisteröityi käyttämään yli 400 eri tahoa. Järjestelmän ytimessä ovat yleiskäsiteontologiat, joita muut ontologiat eri tavoin tarkentavat. Myös ONKI:sta vuonna 2014 Kansalliskirjastossa tuotettu Finto-palvelu keskittyy näihin ontologioihin. Esittelemme seuraavassa lyhyesti tuloksia tällä ontologiatyön osa-alueella.

Kehitystyön lähtökohdaksi otettiin maassamme jo käytössä olevat asiasanastot. Niiden käytöllä voidaan parantaa ja yhdenmukaistaa tiedon indeksointia, jolloin tiedon haussa päästään parempaa tarkkuuteen (precision) ja saantiin (recall). Perinteisiin asiasanoihin perustuvaan tekstihakuun liittyy kuitenkin vakavia haasteita, kuten olemme luvussa 3.2 nähneet. Esimerkiksi ravintoloita haettaessa ei löyty pizzerioita. Myös kielimuurit ovat perinteisen merkkijonohaun haasteena.

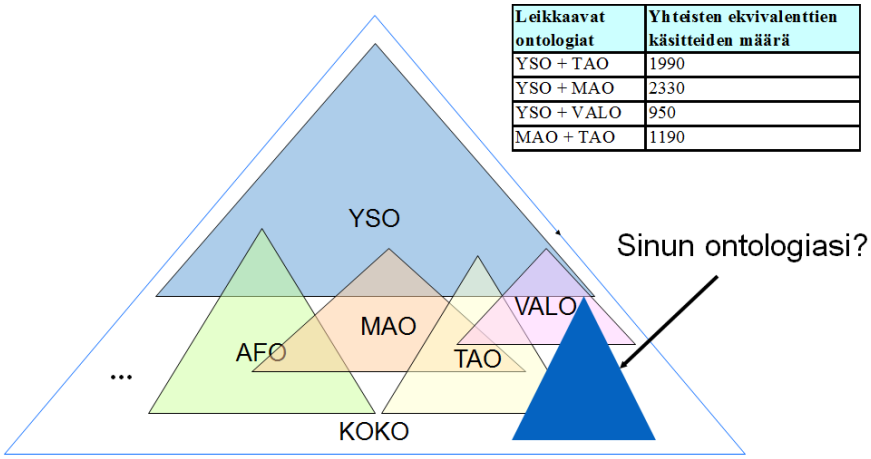
Ontologiaperustainen hakupalvelu sen sijaan ymmärtää paremmin käyttäjäänsä, koska haun apuna käytetään maailman käsitteitä kuvaavia ontologioita. Ne voivat kertoa, että esimerkiski ”Kallio” on osa Helsinkiä ja että ”Kallio” tässä yhteydessä ei tarkoita luonnonmuodostelmaa tai presidentti Kyösti Kalliota. Ontologisten viittausten kautta tieto yhdistyy myös luontevasti toisiin tietoihin, jolloin data rikastuu automaattisesti.

FinnONTO:n ontologisointityö alkoi maamme käytetyimmästä asiasanastosta, Kansalliskirjaston Yleisestä Suomalaisesta Asiasanastosta YSA, joka sisälsi FinnONTO-hankkeen alussa vuonna 2003 yli 20 000 yleiskäsitettä eri aloilta. Sanaston laajentaminen oli edelleen tarpeen, mutta kaikkien eri alojen asiantuntemuksen saaminen sanastotyöhön oli haasteellista, vaikkakin ylläpitoa varten oli olemassa monialainen työryhmä. Samaan aikaan maassamme oli käytössä ja kehitettiin lukuisia YSA:n kanssa erityisesti yleisempien käsitteiden osalta päällekkäisiä erityissanastoja, mikä tuntui monasti tarpeettomalta moninkertaiselta

työltä. Lisähaasteena oli, että samannimisillä käsitteillä saattoi olla eri sanastoissa erilainen merkitys, mikä johti sekaannuksiin tietoja yhdistettäessä.

FinnONTO tarttui näihin sisällöllisiin ja organisatorisiin haasteisiin ehdottamalla kansallisesti koordinoitua mallia, jossa sanastotyö voitaisiin yhteistuumin jakaa eri alojen sanastojen kehittäjäryhmien kesken ja yhdistää tulokset lopulta yhdeksi laajaksi, eri alat kattavaksi KOKO-ontologiaksi. Vision mukaan tämä edistäisi merkittävästi eri alojen tietosisältöjen semanttista yhteentoimivuutta ja säästäisi samalla kehityskustannuksia päällekkäisen työn vähentyessä.

Leikkaavat ontologiat	Yhteisten ekvivalenttien käsitteiden määrä
YSO + TAO	1990
YSO + MAO	2330
YSO + VALO	950
MAO + TAO	1190



Kuva 17.1: Yhteisöllinen kokonaisontologia KOKO koostuu yläontologiasta YSO ja sitä tarkentavista alaontologioista. Ontologioiden keskinäiset leikkaukset kuvassa ovat vain viitteellisiä ja tarkoitettu yleiskuvan havainnollistukseksi.

Työn tuloksena syntyi prototyyppi KOKO-ontologiasta, joka on eräänlainen eri alojen ontologioiden muodostama Linked Data -ontologiapilvi. KOKOa on havannollistettu kuvassa 17.1. Sen ytimenä on YSA:sta kehitetty Yleinen Suomalainen Ontologia YSO, joka muodostaa KOKO:n *yläontologian* (upper ontology) sisältäen merkitysalaltaan laajimmat käsitehierarkian käsitteet. Erikoisalojen tarkempi käsitteistö "ripustuu" sitten YSO:n eri haaroihin hierarkioita syventäen.

yläontologia

KOKO-ontologiapilveen kuului ensivaiheessa (v. 2013) taulukossa 17.1 luetellut 15 ontologiaa, yhteensä yli 71 000 käsitettä. Ontologiat linkitet-

Ontologia	**Käsitteitä**	**Asiasanasto**
Yleinen suomalainen ontologia (YSO)	n. 26 000	Yleinen suomalainen asiasanasto
Julkishallinnon ontologia (JUHO)	n. 6 350	Valtioneuvoston asiasanasto
Kaunokirjallisuuden ontologia (KAUNO)	n. 5 100	Kaunokki-asiasanasto
Kielitieteen ontologia (KTO)	n. 950	Uralistiikan tutkimuksen bibliografian asiasanaluettelo
Kirjallisuudentutkimuksen ontologia (KITO)	n. 850	Kirjallisuudentutkimuksen asiasanasto
Kulttuurien tutkimuksen ontologia (KULO)	n. 1 500	Kulttuurien tutkimuksen asiasanasto
Liiketoimintaontologia (LIITO)	n. 3 400	Yrityssuomi.fi-portaalin käsitteistö
Merenkulkualan ontologia (MERO)	n. 1 400	Merenkulkualan asiasanasto
Musiikin ontologia (MUSO)	n. 1 000	Musiikin asiasanasto
Puolustushallinnon ontologia (PUHO)	n. 2 000	Puolustushallinnon asiasanasto
Taideteollisuusalan ontologia (TAO) ja Museoalan ontologia (MAO)	n. 8 100	Muotoilun ja viestinnän asiasanasto ja Museoalan asiasanasto
Terveyden ja hyvinvoinnin ontologia (TERO)	n. 6 500	TESA, Stameta-asiasanasto, ja n. 2500 MeSH-käsitettä (Medical Subject Headings)
Valokuvausalan ontologia (VALO)	n. 2 000	Valokuvan asiasanasto
Viikin tiedekirjaston ontologia (AFO)	n. 6 000	Agriforest-asiasanasto

Taulukko 17.1: KOKO-ontologiapilven alkuperäiset osaontologiat

tiin toisiinsa ensin sellaisenaan, koska ne perustuivat eri aloilla jo käytössä oleviin asiasanastoihin. Seuraavaksi vuorossa on KOKO-ontologiapilven rakenteen muokkaus tarkoituksenmukaisempaan muotoon mm. tarpeettomia päällekkäisyyksiä poistamalla, pilven eri osien ylläpitoon liittyvästä työjaosta sopiminen, ja rakenteen laajentaminen ja linkittäminen muihin ontologioihin.[4] Työ on jatkunut Kansallikirjaston koordinoimana Finto-palveluun liittyen.

KOKO-pilven ontologiat perustuvat maassamme jo aiemmin käytettyihin vastaaviin asiasanastoihin. Alojen jako taulukon mukaisesti ei ole optimaalinen, vaan esimerkiksi monia pienempiä toisiinsa liittyviä ontologioita kannattaa jatkossa yhdistää laajemmiksi kokonaisuuksiksi. FinnONTO-hankkeen puitteissa tähän ei kuitenkaan haluttu ottaa kantaa eikä hankkeessa myöskään ryhdytty osaontologioiden päällekkäisyyksien poistamiseen: nämä tehtävät jätettiin sanastojen kehittäjätahojen asiaksi.

KOKO:n osaontologiat on muodostettu yhtenäisellä menetelmällä lukuun ottamatta kansainvälistä lääketieteen MeSH-sanastoa, jonka rakenne on alkuperäinen. Joissain tapauksissa ontologisoinnin yhteydessä alkuperäistä sanastoa laajennettiin ja YSA/YSO:n osalta hankkeessa kehitettiin sanaston ensimmäinen englanninnos, mikä mahdollistaa sen linkittämistä kansainvälisiin ontologioihin.

Haasteena asiasanaston ontologisoinnissa on sen semanttisen rakenteen täsmentäminen niin, että siinä toteutuvat koneellisessa päättelyssä, esimerkiksi kyselyn laajentamisessa, tarvittavat ominaisuudet. Keskeisimmät ontologiset muutokset ja tarkistukset, joihin työssä keskityttiin, olivat:

- Luokkahierarkian täydentäminen. Ontologisen luokkahierarkian muodostaminen ja täydentäminen niin, että kaikilla käsitteillä (kaikkein ylin pois lukien) on ainakin yksi yläkäsite. Tämä parantaa käsitteiden linkitettävyyttä ja mahdollistaa mm. päättelyä.
- Laajempi/suppeampi termi -suhteiden tarkentaminen. Asiasanastoissa käytettyjen laajempi termi/suppeampi termi (LT/ST) -suhteiden tarkentaminen yläluokkasuhteiksi (`rdfs:subClassOf`),

[4] Tätä FinnONTO-hankkeessa tehtyä laajaa työtä eri alojen asiasanastojen muuttamiseksi ontologioiksi on käsitelty tarkemmin raportissa [42].

osa-kokonaisuus-suhteiksi tai assosiatiivisiksi suhteiksi. Työssä keskityttiin erityisesti `rdfs:subClassOf` suhteiden kehittämiseen.

- Yksilö-luokka-suhteiden transitiivisuuden tarkistaminen. Keskeinen ontologioissa käytetty periaate luokkahierarkian muodostamisessa on, että kaikkien luokkien yksilöiden tulee olla samanaikaisesti kaikkien yläluokkiensa yksilöitä. Tämä ominaisuus on hyödyllinen koneellisessa päättelyssä, esimerkiksi ominaisuuksien periytymisessä ja kyselyn laajentamisessa.

- Monimerkityksisten asiasanojen merkitysten erottelu. Asiasanastojen monimerkityksisten käsitteiden merkitysten jakaminen eri käsitteiksi tarpeen mukaan niin, että ne voidaan sijoittaa luokkahierarkiaan.

Tarkastellaan esimerkkinä transitiivisuuden haasteesta asiasanoja taskupeilit ja peilit, jotka on määritelty asiasanastoissa seuraavien laajempien termien avulla (LT-suhde):

```
taskupeilit LT peilit
peilit LT huonekalut
```

Tämän mukaan taskupeilit on peilejä ja toisaalta peilit ovat huonekaluja, mikä kuulostaa järkevältä, mutta transitiivisuus molempien suhteiden ylitse ei toimi, sillä taskupeilit eivät ole huonekaluja. Jos LT-suhteet muutetaan mekaanisesti luokkasuhteiksi (`rdfs:subClassOf`) seurauksena on, että haettaessa huonekaluja hakutulokseen voi tulla virheellisesti mukaan taskupeilejä. Transitiivisuuden tarkistaminen on erityisen haastavaa KOKO-ontologiapilvessä, kun luokkahierarkiat kulkevat eri ontologioiden kautta. Taskupeilit löytyy Museoalan asiasanastosta MASA muttei YSA:sta. Esimerkiksi asiasanojen monimerkityksellisyyden haasteista sopii YSA:ssa oleva asiasana ”lapset”, joka voi tarkoittaa mm. ikäluokkaa tai perhesuhdetta. Haasteena on, että aikuisetkin ovat omien vanhempiensa lapsia. Siksi YSO-ontologiassa lapset-käsitteestä on erotettu merkitykset ”lapset (ikään liittyvä rooli)” ja ”lapset (perheenjäsenet)”. Mikäli erottelua ei tehtäisi, saattaisi tietojärjestelmä esimerkiksi suositella aikuisille tarpeettomasti lastenkirjoja luettavaksi, koska nämä ovat vanhempiensa lapsia.

FinnONTO-hankkeessa neuvoteltiin lupa julkaista kaikki KOKO-ontologiat avoimena datana. Tulosten hyödyntäminen myös kaupallisiin tarkoituksiin on vapaata, ainoastaan aineiston kehittäjätaho on mainittava (CC-BY-lisenssi). Edellytykset työn jatkamiselle ja tulosten hyödyntämiselle ovat tältä osalta paremmat, kuin hankkeen alkaessa.

17.3 Työn arviontia

Ontologiatyö, jo pelkästään insinöörien filosofiasta uusiokäyttöön ottama termi ontologia, on herättänyt myös vastustusta. Kritiikkiä on synnyttänyt ontologisoinnin yhteydessä syntyvä yhä tarkempi jaottelu merkityksiin, mikä johtaa yhä isompaan ja mutkikkaampaan käsitehierarkiaan. Tämä lisää paitsi ylläpitotyötä myös työtä indeksoinnissa ja saattaa rajata termeihin liittyviä vivahteita liikaa. Jos esimerkiksi kuvaillaan Albert Edelfeltin teos ”Pariisin Luxembourgin puistossa”, pitääkö siinä leikkivät lapset kuvata erikseen sekä ikäryhmänä, perhesuhteena että sosiaalisena ryhmänä, ja katetaanko näilläkään lapsiin liittyvä merkitysten ala?

Ontologiatyössä mennäänkin helposti liian hienojakoisiin ratkaisuihin, jos järjestelmän käyttötarkoitus pääsee unohtumaan. FinnONTO:ssa tähän vaaraan varauduttiin asettamalla tavoitteeksi minimaalinen ja mahdollisimman yleiskäyttöinen sanastojen ontologisointi, joka kuitenkin ratkaisisi eräitä keskeisimpiä asiasanastoihin ja asiasanastotyöhön liittyviä haasteita tietotekniikan sovellusten kannalta. Ontologian laajenemisen ongelma liittyy myös metatietomallien kehittämiseen, mikä kannattaa ottaa huomioon. Ontologioitahan käytetään metatiedon esittämiseen eikä yksinään. Esimerkiksi FinnONTO-hankkeen piirissä kehitetty julkishallinnon JHS suositus 183 *Julkisen hallinnon palvelujen tietomalli ja ryhmittely verkkopalveluissa*[5] päätyi luomaan yhden laajan palveluontologian sijasta yleisemmän metatietomallin, jonka metatietokentille voidaan valita arvoja eri ontologioista.

Ehkä keskeisin periaatteellinen kysymys ontologiatyössä on, missä määrin maailmaa voidaan ja on tarkoituksenmukaista kuvata loogisena struktuurina. Maailma ja siihen liittyvä tieto on esimerkiksi monin tavoin *epävarmaa* (uncertainty), *puutteellista* (incomplete) ja *sumeaa* (fuzzy). Eikö maailman loogisessa kuvaamisessa jo epäonnistuttu kerran 80-luvulla

[5] http://www.jhs-suositukset.fi/suomi/jhs183

tekoälytutkimuksen asiantuntijajärjestelmien yhteydessä? Semanttisen webin lähtökohdat ovat kuitenkin logiikan soveltamisen suhteen hyvin erilaiset verrattuna tekoälyn asiantuntijajärjestelmiin. Linked Data -ajattelussa korostetaan, että ontologioiden ei tarvitse olla virheettömiä ja loogisesti eheitä voidakseen olla silti hyödyllisiä, päinvastoin kuin vaikkapa sytostaattihoitoa annostelevan asiantuntijajärjestelmän. Eihän WWW ylipäätään ole virheetön, mutta silti hyödyllinen. On kuitenkin selvää, että täsmälliseen tietoon perustuva logiikka asettaa rajoitteita luonteeltaan monimuotoisen ja epätäsmällisen maailman kuvaamisella, ja että yhä automaattisemmin menetelmin ja yhteisöllisemmin tuotetun yhdistetyn datan laatu tulee olemaan keskeisiä haasteita semanttisessa webissä.

Haasteista huolimatta webin kehitys on kuitenkin astunut uudelle semanttiselle tasolle eikä paluuta entiseen ole. Web of Data luo WWW:n sisään uuden, W3C:n standardeihin perustuvan sisältökerroksen ja infrastruktuurin, joka on monin tavoin hyödyllinen. FinnONTO:n ontologioita ja ONKI-palvelua esimerkiksi on käytetty hyväksi monissa sovelluksissa, kuten Terveyden ja hyvinvoinnin laitoksen TerveSuomi-palvelussa, Yleisradion verkkosivuilla, kymmeniä kulttuurialan kokoelmia linkittävässä Kulttuurisampo-palvelussa, työ ja elin-keinoministeriön YritysSuomi-palvelussa, monissa maamme taide- ja kulttuurihistoriallisissa museoissa, yleisten kirjastojen Kirjasampo-portaalissa ja Sotasammossa.

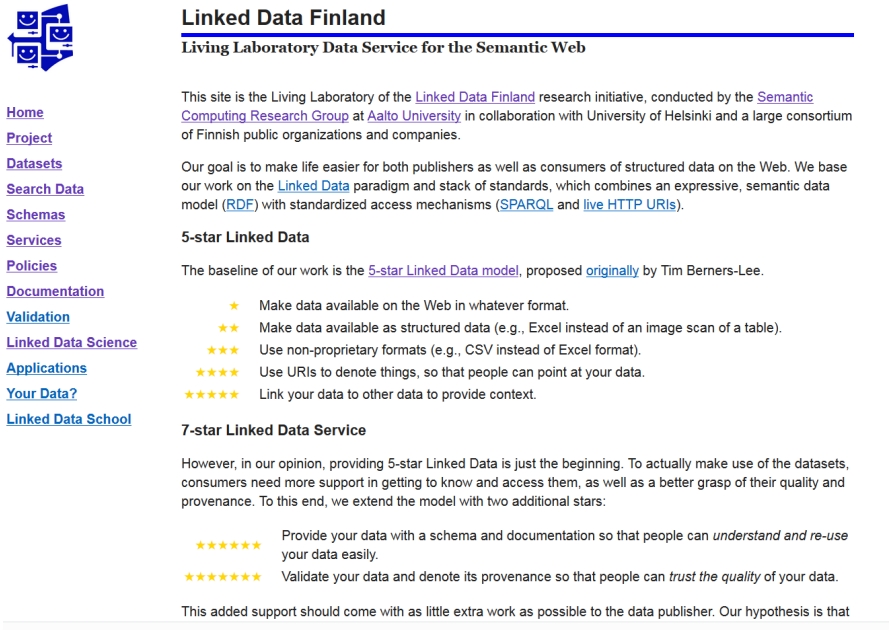
17.4 Linkitetyn datan palvelut

Käytännön sovellusten kannalta keskeinen palvelu semanttisessa webissä on linkitetyn datan palvelut, joiden rajapintojen varaan linkitetyn datan sovellukset kehitetään. Aiemmissa luvuissa on jo esitelty linkitetyn datan julkaisuperiaatteetteita, kuten URI-tunnisteiden ratkontaa sekä SPARQL-palvelupisteen käyttöä. Datapalvelu tarjoaa asiakkaalleen yleensä mahdollisuuden 1) saada datasta tarkempaa tietoa, 2) rajapinnat datan käyttämiseen ja 3) mahdollisuuden datan laataamiseen kopiona omaan käyttöön.

Verkossa olevia hyödyllisiä SPARQL-palvelupisteitä on listattu ja esitelty mm. W3C:n sivuilla[6]. Suosittuja palveluita ovat mm. Wikipedian dataa

[6] https://www.w3.org/wiki/SparqlEndpoints

tarjoava Wikidata ja DBpedia. Avoimet palvelut eivät kuitenkaan välttämättä aina ole toiminnassa ja niiden käyttöön liittyy haasteita. Palvelu kuormittuu helposti, jos käyttäjät tekevät sinne paljon laskentaa vaativia kyselyitä, joiden tuloksena syntyy isoja vastausjoukkoja. Käytön kontrollointi ei välttämättä ole helppoa täysin avoimessa ympäristössä.



Kuva 17.2: Linked Data Finland -palvelun etusivu. 7-tähden julkaisumalli kannustaa linkitetyn datan uusiokäyttöön. Datajulkaisut ja palvelut löytyvät vasemman sarakkeen linkkien kautta.

elävä laboratorio

Suomessa linkitetyn datan palveluita on pilotoitu erityisesti Aalto- ja Helsingin yliopiston kehittämässä Linked Data Finland -palvelussa LDF.fi (ks. kuva 17.2). Palvelun ideana on olla ONKI-ontologiapalvelun kaltainen *elävä laboratorio* (living laboratory), jossa uutta teknologiaa voidaan kehittää ja testata laajempaa käyttöönottoa varten. Tavoitteena on edistää avoimen linkitetyn datan tuotantoa, julkaisua ja uudelleen käyttöä sovelluksissa. Tätä kannustaakseen LDF.fi laajentaa Tim Berners-Leen viiden tähden mallin seitsemän tähden linkitetyn datan malliksi. Kun viiden tähden hotellien lisäksi on markkinoille ilmestynyt seitsemän tähdenkin hotelleja, voisi samaan pyrkiä datahotelleissakin:

★★★★★★Kuudennen tähden ideana on kannustaa datan julkaisijaa julkaisemaan datan ohella myös siinä käytetyt metadataskeemat, mikä helpottaa huomattavasti datan uudelleenkäyttöä ja validointia.

★★★★★★★Seitsemäs tähti tulee, jos data on validoitu siinä käytettyjen skeemojen suhteen, ts. että on olemassa edes teknisiä takeita datan laadusta.

LDF.fi-palveluun on liitetty lukuisia lisäpalveluita helpottamaan datan tuotantoa, julkaisua ja dataan tutustumista. Kun julkaistavaksi tarkoitetusta datasta luo standardin mukaisen metadatakuvauksen, LDF.fi palvelu generoi sen perusteella automaattisesti datajulkaisulle oman kotisivun, jossa on paitsi kuvaus itse datasta ja sen lisenssiehdoista, myös valmiita työkaluja linkitettynä datan käyttöä varten. Sivulta löytyy esimerkiksi linkki datasta automaattisesti tuotettuun dokumentaatioon, esimerkkejä, joiden avulla dataa voi selailla ja lomake SPARQL-kyselyiden tekemistä varten valmiilla esimerkkikyselyllä.

Palveluun on liitetty myös interaktiivinen koulutuspaketti *Linked Data School LinDa*[7] linkitetyn datan tuottamiseen, julkaisemiseen ja hyödyntämiseen liittyen.

LDF.fi-palvelun toteutukseen kuuluu Varnish edustapalvelin[8], joka hoitaa HTTP-UsRI-pyyntöjen käsittelyn. Taustalla olevat SPARQL-palvelupisteet on toteutettu Apache Jena Fuseki -alustan[9] avulla.

[7] http://linda.seco.cs.aalto.fi
[8] https://varnish-cache.org/
[9] https://jena.apache.org/documentation/fuseki2/

Luku 18

Sovellusesimerkki: Sotasampo

Tässä luvussa esitellään edellisessä luvuissa kuvattuun Sampo-portaalimalliin perustuva sovellusesimerkki Sotasampo.[1] Aluksi määritellään sovelluksen tavoitteet ja käyttötapaukset. Tämän jälkeen tarkastellaan portaalin tietomallia, sisällöntuotannon prosessia, sovellusportaalin toimintaa ja lopuksi sen perustana olevaa datapalvelua. Se sisältää sotahistorian sovellusalueelle luodun linkitettyyn avoimeen dataan perustuvan tietoinfrastruktuurin, jonka varaan voidaan kehittää uusia sovelluksia.

18.1 Tavoitteet ja käyttötapaukset

Sotasampo julkaisee linkitettynä avoimena datana talvi- ja jatkosotaan liittyviä keskeisiä kansallisia tietoaineistoja yhteistyössä Kansallisarkiston, puolustusvoimien, Maanmittauslaitoksen, Suomen Sotahistoriallisen Seuran, Yleisradion ja muun yhteistyöverkoston kanssa. Tavoitteena on Suomen historian suurimman tragedian ymmärtämisen edistäminen ja Digital Humanities -tutkimuksen tukeminen uusimman semanttisen webin ja kieliteknologian avulla. Datapalvelun hyödyntämistä pilotoidaan Sotasampo.fi-verkkopalvelussa, joka koostuu erilaisista näkymäsovelluksista aineistoihin, kuten sotien tapahtumat aikajanalla ja kartalla, henkilöt, joukko-osastot, historialliset paikat, veteraanien muistelmat, sotasurmat, valokuvat, sankarihautausmaat ja sotavansgit. Hankkeen tavoit-

[1] Sotasammon kotisivulta löytyy runsaasti aiheeseen liittyvää tietoa: https://seco.cs.aalto.fi/projects/sotasampo/.

teena on tarjota semanttisesti rikas kokonaisuus sekä avoimena linkitettynä datana sovelluksille että verkkosivustona ihmisille. Innovaationa on rikastaa eri datasiilojen aineistoja toistensa avulla linkittämällä dataa yhteisen tapahtumaperustaisen ontologiainfrastruktuurin kautta.

G. W. F. Hegelin mukaan *historiasta opimme sen, että emme opi historiasta mitään.* Toivottavasti näin ei tapahdu Suomessa toisen maailmansodan osalta. Itsenäisyytemme ajan sotahistoria, erityisesti talvi- ja jatkosota, ovat Suomen kansallisen identiteetin ja maanpuolustustahdon keskeinen tukipilari ja laajaa kiinnostusta herättävä historian tutkimuskohde niin tutkijoille kuin kansalaisillekin. Useimpien suomalaisten vanhemmat tai isovanhemmat osallistuivat tavalla tai toisella sodan eri tapahtumiin, pahimmassa tapauksessa surmansa saaden. Sotasampo-hankkeen tavoitteena on rakentaa uusimman Linked Open Data -teknologian avulla suomalaista identiteettiä lisäämällä ymmärrystämme viime sodistamme ja edistää näin rauhaa. Kehitettäviä uusia teknisiä ratkaisuja voidaan soveltaa muihinkin Digital Humanities -alueen aineistoihin.

Sotasampo koostuu kahdesta pääosasta:

1. **Linkitetyn avoimen datan palvelu**. Palvelu tarjoaa Suomen toiseen maailmansotaan liittyvää dataa rajapintojen kautta sitä tarvitseville sovellusten kehittäjille ja toisille verkkopalveluille.

2. **Semanttinen portaali**. Sotasampo.fi on Sotasampo-datapalvelun päälle kehitetty sovellusjoukko ihmiskäyttäjille, tutkijoille ja kansalaisille.

Datapalvelu ja semanttinen portaali julkistettiin 27.11.2015 Kansallisarkistossa pidetyssä tilaisuudessa[2].

18.2 Aineistot ja tiedon tuottajayhteisö

Vaikka verkossa on runsaasti tietoa sodistamme ja sotahistoriallista kirjallisuutta paljon saatavilla, ei tietoa ole ollut saatavilla koneluettavassa

[2] http://seco.cs.aalto.fi/events/2015/2015-11-27-sotasampo/

muodossa avoimena datana. Tämä on estänyt tiedonsaantia, datan analysointia ja käyttöä tutkimuksessa sekä sotahistoriallisten sovellusten kehittämistä. Maassamme on kuitenkin aiemmin julkaistu merkittäviä, laajan suosion saavuttaneita verkkopalveluita toiseen maailmansotaan liittyen, ja arkistojen kätköistä löytyy runsaasti lisää näihin liittyvää tietoa:

- **Sota-ajan valokuvat**. Sotamuseon SA-kuva-arkistopalvelu sisältää 160 000 Suomen talvi-, jatko- ja Lapin sodan ajan valokuvaa vuosilta 1939–1945 metatiedoilla varustettuna. Kuvat ovat katseltavissa verkossa. Sotamuseolla on myös arkistoissaan mm. n. 1300 piirrosta kattava jatkosodan ajan TK-piirroskokoelma, sotasaalislippukokoelma, ilmavoimien 60 000 kuvan 1918–1960 kokoelma ja yksityisten henkilöiden kuvakokoelmia. SA-kuva-arkiston verkkojulkaisulla oli ensimmäisen vuoden aikana n. miljoona vierailijaa.

- **Sotasurmatietokannat**. Kansallisarkisto on julkaissut verkkopalveluna useita sotahistoriallisia tietokantoja, kuten Suomen sotasurmat 1914–1922, Suomen sodissa 1939–1945 menehtyneet ja Suomi, sotavangit ja ihmisluovutukset 1939–1955 . Suomen sotien 1939–1945 sotasurma-aineistot ovat olleet ahkerassa käytössä verkossa: ensimmäisenä julkaisuvuotena 2013 aineistoihin tutustui n. 83 000 kävijää. Muita kiinnostavia aineistoja ovat n. 30 000 sotapäiväkirjaa, joiden avulla voi mm. tutustua yksittäisen sotilaan sotahistoriaan, kun hänen joukko-osastonsa tunnetaan. Sotilaiden kantakorteista selviää yksittäisten rintamamiesten sota-ajan tapahtumia, mutta niitä ei ole vielä digitoitu.

- **Kansa Taisteli – miehet kertovat -lehdet (1957–1986)**. Lehden artikkelit, yhteensä n. 3360 kpl, ovat paljolti rintamilla palvelleiden veteraanien itsensä kirjoittamia muistoja sota-ajan tapahtumista. Ne on kesällä 2014 julkaistu verkossa Suomen Sotahistoriallisen Seuran ja Bonnierin toimesta.

- **Sota-ajan filmi- ja ääniteaineistot**. Yleisradiolla ja Kansallinen audiovisuaalinen arkisto KAVA:lla on runsaasti arkistoissa sota-aikaan liittyvää kuva-, radio-, filmi- ym. materiaalia, josta osa löytyy jo esimerkiksi YLE:n Elävästä arkistosta.

- **Sotahistorialliset kokoelmat**. Sotamuseossa ja eräissä muissakin maamme museoissa on runsaasti sota-aikaan liittyvää esineistöä ja

dokumentteja, joita on osin julkaistu mm. Kulttuurisampo- (kulttuurisampo.fi) ja Finna-portaaleissa (finna.fi).

- **Kartat ja paikkatieto**. Maanmittauslaitos on avannut paikkatietodataansa. Saatavilla on digitoituna avoimena datana mm. luovutetun alueen vanhat Karjalan kartat. Jyrki Tiittasen talkootyönä on syntynyt n. 35 000 paikannimen geokoodattu paikannimirekisteri luovutetun Karjalan alueilta.

- **Sota-ajan muistot**. Suomalaisen Kirjallisuuden Seuran arkistoissa on sota-ajalla kerättyä tietoa ja muistoja sota-ajan elämästä, samoin Svenska Litteratur Sällskapetilla on aineistoja mm. Ruotsiin lähetettyihin sotalapsiin liittyen.

Sotasampo-hanke kokoaa edellä mainittuja datan julkaisijoita mukaan yhteisiin talkoisiin ajatuksena luoda eri organisaatioiden erillisten datasiilojen aineistoista suuri, avoin linkitetyn datan semanttinen verkko. Tutkimus- ja kehitystyötä on tehty osana Aalto-yliopiston ja Helsingin yliopiston HELDIG-keskuksen Semanttisen laskennan tutkimusryhmän Linked Open Data Science hanketta, joka kuuluu opetus- ja kulttuuriministeriön rahoittamaan Avoin tiede ja tutkimus -ohjelmaan. Sotasampo oli myös yksi Suomen itsenäisyyden satavuotisjuhlavuoden ohjelmahankkeista.

Sotasampo.fi:n ensimmäiseen demonstaattoriin muunnettiin linkitetyksi dataksi kuvan 18.1 taulukossa lueteltuja aineistoja.

Seuraavassa esitellään lyhyesti ensin Sotasammon tietomallia, sisällöntuotantoa, sitten itse portaalisovellus ja lopuksi palvelun perustana oleva datapalvelu ja infrastruktuuri.

18.3 Metadata ja ontologiat

Sotasammon ontologiseksi perustaksi valittiin CIDOC CRM (ks. luku 8.3), jonka tapahtumaluokille luotiin aliluokkina uusia sotahistoriaan liittyviä tapahtumaluokkia, kuten *taistelu* (`:Battle`) ja *poliittinen aktiviteetti* (`:PoliticalActivity`). Tietomallin ydin on esitetty kuvassa 18.2. Ideana on kuvata sotahistoriallinen tieto tapahtumina, joita karaterisoi erityisesti tapahtuman paikka (Missä?), aika (Milloin?) ja tapahtumaan

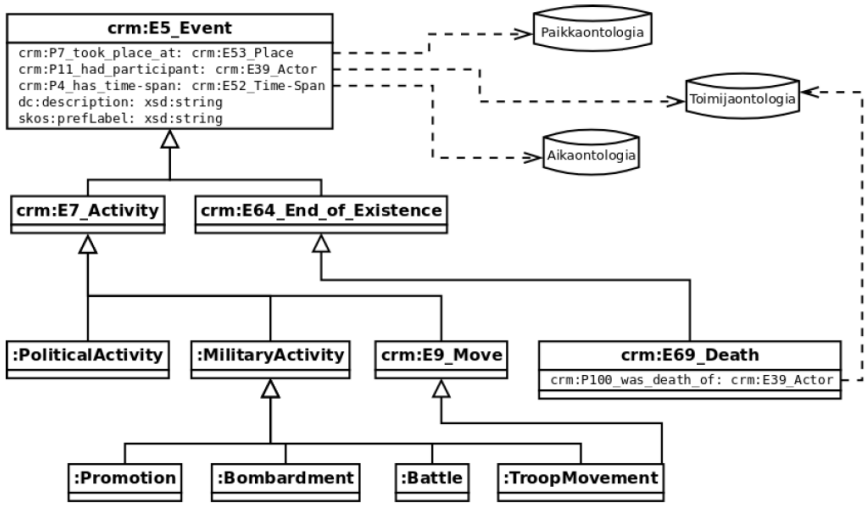
	Datan luovuttaja	Aineisto (25.11.2015)	Määrä	Datan lähteet ja tekijänoikeus	Lisenssi linkitettyyn dataan
1	**Puolustusvoimien SA-**	TK-kuvat ja videot	160 000	Puolustusvoimat & Aalto/SeCo	CC BY 4.0
2	**Kansallisarkisto**	Menehtyneet 39-45	95 000	Kansallisarkisto & Aalto/SeCo	CC BY 4.0
3	**Kansallisarkisto**	Organisaatiokortisto	500	Kansallisarkisto & Aalto/SeCo	CC BY 4.0
4	**Kansallisarkisto**	Sotapäiväkirjat	26 500	Kansallisarkisto & Aalto/SeCo	CC BY 4.0
5	**Kansallisarkisto**	Senaatin kartat	404	Kansallisarkisto & Aalto/SeCo	CC BY 4.0
6	**Suomen Sotahistoriallinen Seura Bonnier Publications Oy**	Kansa taisteli -lehtien artikkelit 1957-1986	3 360	Metadata: Timo Hakala & Aalto/SeCo. Artikkelit © Suomen Sotahistoriallinen Seura ja Bonnier, http://sshs.fi.	CC BY 4.0 (metadata)
7	**Maanmittauslaitos**	Paikannimirekisteri	800 000	Maanmittauslaitos (MML) & Aalto/SeCo	CC BY 4.0
8	**Maanmittauslaitos**	Karjalan topograf. kartat	47	MML & Aalto/SeCo	Ei-kaupallisiin tarkoituksiin
9	**Jyrki Tiittanen**	Karjalan paikat geokood.	35 000	Jyrki Tiittanen & Aalto/SeCo	CC BY 4.0
10	**Aalto-yliopisto**	Sotien tapahtumat	11 275	Aalto/SeCo. Lähteet: Talvi- ja jatkosodan pikkujättiläiset, organisaatiokortisto ym.	CC BY 4.0
	Aalto-yliopisto	Henkilöt (Menehtyneet 39-45 datan lisäksi)	1 050	Aalto/SeCo. Lähteet: Talvisodan historia, sotapäiväkirjat, organisaatiokortisto, Suomen Rintamamiehet 1939-1945, Maanpuolustuskorkeakoulu/Ohto Manninen, Wikipedia ym.	CC BY 4.0
		Biografiat	6 300	Suomalaisen Kirjallisuuden Seura / Kansallisbiografia & Aalto/Seco	CC BY 4.0
		Joukko-osastot	3 180	Aalto/SeCo. Talvisodan historia, Talvisodan sotapäiväkirjat, organsaatiokortisto	CC BY 4.0
	YHTEENSÄ		**1 142 616**		

Kuva 18.1: Sotasammon aineistoja, niiden lähteet ja datan määrä.

liittyvät toimijat (Kuka?). Näiden kuvaukset URI-tunnisteilla identifioituina löytyvät omista aiheontologioistaan. CIDOC CRM valittiin, koska sen tapahtumamalli tukee luontevasti tapahtumakeskeistä sotahistorian esittämistä ja koska malli on jo laajemmassa käytössä oleva kehittynyt standardi. Pyörää ei haluttu lähteä keksimään uudelleen.

Kuvan 18.1 aineistoista suurimmat, esimerkiksi SA-kuva-arkiston kuvat, Menehtyneet-tietokannan tiedot, sotapäiväkirjat ja Kansa Taisteli -lehdet oli jo aiemmin digitoitu ja julkaistukin verkossa ainakin osittain. Niistä oli saatavilla metadataa digitaalisessa muodossa CSV-muotoisina tietokantakopiona ja Excel-taulukkoina. Taulukkomuotoon digitoitiin lisäksi käsityönä monista lähdeteoksista tietoa mm. sota-ajan tapahtumista, Mannerheim-ristin ritareista, puolustusvoimien organisaatiokorteista ja paikannimilistoista. Lisäksi näistä tietoaineistoista tuotettiin uusia taulukoita datalähtöisesti tarvittavia ontologioita varten. Esimerkiksi Menehtyneet-tietokannassa mainituista henkilöistä saatiin dataa henkilöontologiaan ja joukko-osastoista tuhansittain joukko-osastojen nimiä.

Taulukkomuotoisen datan muuttaminen RDF-muotoon on teknisesti melko suoraviivasta: taulukon rivit vastaavat usein luokkien yksilöitä ja sarakkeiden arvot ominaisuuksien arvoja. Esimerkiksi valokuvataulukossa jokainen rivi vastaa valokuva-yksilöä, johon voi liittyä tieto kuvaajasta, kuvausajasta, kuvauspaikasta ja valokuvan sisällön kuvaus tekstinä. Tilanteesta ja mallinnuksesta riippuen taulukon rivin muunnos voi kui-



Kuva 18.2: Sotasammon tietomallin ytimenä on CIDOC CRM ontologia ja sen tapahtumaluokka `crm:E5_Event`. Kuvassa `crm` on CIDOC CRM -standardin nimiavaruus; oletusnimiavaruutena on Sotasammon oma nimiavaruus.

tenkin olla mutkikkaampikin. Esimerkiksi Menehtyneet-tietokannassa kerrotaan vainajasta mm. sekä syntymä- että kuolinaika ja -paikka ja lisäksi mahdollisesta haavoittumisesta. CIDOC CRM -mallin mukaan näistä pitää luoda erilliset tapahtumaluokan yksilöt, jotka sitten voivat rikastaa muuta henkilöön liittyvää tapahtumatietoa vaikkapa Wikipediasta tai organisaatiokorteista.

Taulukkomuotoisen datan muunnoksia ja putsausta varten on olemassa valmiita työkaluja kuten OpenRefine. Sotasampo hankkeessa jouduttiin kuitenkin kehittämään tähän työhön omia uusia välineitä ja muunnosohjelmia, jotka soveltuivat juuri Sotasammon datajoukkojen käsittelyyn, kuten Menehtyneet-tietokannan datan muokkaukseen.

Taulukkomuotoisen datan ohella aineistoihin kuului myös karttoja. Vanhat kartat saatiin valmiiksi digitoituina, mutta niistä ei ollut käytettävissä koordinaattitietoa, mikä oli tarpeen karttapohjaisissa visualisoinnessa. Karttojen asemointia varten projekti otti käyttöön New Yorkin yleisessä kirjastossa (New York Public Library) kehitetyn MapWarper-työkalun, jonka avulla luovutetun alueen vanhat kartat asemoitiin nykyisten kartto-

jen päälle nykyisin käytössä olevaan koordinaattijärjestelmään WGS 84. Metatieto kartoista talletettiin omaan RDF-graafiinsa, jota kautta niitä voidaan hakea ja visualisoida Google-karttojen päällä paikkatiedon perusteella. Tässä hyödynnettiin Aalto-yliopiston historiallisten paikkojen ja karttojen ontologiapalvelua Hipla.fi.

18.4 Entiteettien automaattinen linkitys

Sotasampo linkittää myös tekstidokumentteja, kuten Kansa Taisteli -lehtien kaikkien vuosikertojen 1957–1986 3360 artikkeleita, jotka on julkaistu Sotahistoriallisen seuran kotisivuilla[3]. Näiden linkitystä varten oli käytettävissä hakemistomuotoinen metadata Excel-taulukkona[4], jonka sen laatinut Timo Hakala luovutti Sotasampo-hankkeen käyttöön. Tämän lisäksi teksteistä tunnistettiin kieliteknologisin välinein metatietoa perustuen mm. mainintoihin henkilöistä, organisaatioista ja paikoista. Tämän avulla tekstejä voitiin yhdistää paremmin niihin liittyviin muihin aineistoihin. Tässä hyödynnettiin SeCo-ryhmässä kehitettyä automaattisen annotoinnin työkalua ARPA ja siihen liittyvää CORE-ohjelmistoa (Contextual Reader). CORE kykenee etsimään dokumentissa olevia viittauksia ontologioissa oleviin käsitteisiin ja tarjoamaan niihin liittyvää lisätietoa verkossa olevien datapalveluiden kuten DBpedia tai Sotasampo kautta.

Keskeinen haaste linkitetyn datan tuotannossa on tekstuaalisia ilmauksia vastaavien linkkien muodostaminen. Esimerkiksi henkilöiden nimille, joukko-osastojen nimille ja lyhenteille, paikannimille ja sisällönkuvailussa käytetyille asiasanoille pitää löytää vastaavat yksikäsitteiset käsitteet ontologioissa ja datassa. Jos data tulee tietokannasta tai on muuten määrämuotoista, tiedon rakenne auttaa linkityksessä. Esimerkiksi sana *Taipale* tietokentässä, jossa on tietokantaskeeman mukaan henkilön sukunimi, ei voi viitata Kannaksella olevaan saman nimiseen paikkaan *Taipale*, jolla on luonnollisesti eri URI-tunnus.

Haasteellisinta linkityksen muodostamisen kannalta on vapaan tekstin automaattinen annoitointi, jossa tekstistä muodostetaan tietokoneen toi-

[3] http://kansataisteli.sshs.fi/

[4] http://kansataisteli.sshs.fi/files/Kansa_Taisteli_hakemisto_57_86.pdf

mesta semanttinen kuvaus. Tähän ei riitä perinteinen nimettyjen entiteettien tunnistaninen (Named Entity Recogniition, NER), jossa tunnistetaan esimerkiksi henkilöihin, organisaatioihin ja paikkoihin viittaavat sanat ja ilmaisut. Lisäksi on pääteltävä, mihin yksilöön ontologiassa sana tai ilmaus viittaa. Menehtyneet-tietokannassa esimerkiksi on lueteltu 409 henkilöä, joiden sukunimi on *Korhonen*. Jos tekstissä kerrotaan sotamies *Korhosen* kuolemasta, ei riitä tietää, että *Korhonen* on henkilöön liittyvä erisnimi, vaan tarvitaan myös nimettyjen entiteettien linkittämistä (Named Entity Linking, NEL). Ilman lisätietoa ei ole mahdollista päätellä kenestä on kysymys, ja monessa tapauksessa lopullinen vastaus voi jäädä joka tapauksessa epävarmaksi kuten historiassa usein tapahtuu.

Lisähaastetta entiteettien linkitykselle antaa aineistoon mahdollisesti liittyvät OCR-prosessin virheet, mikä esimerkiksi oli tilanne Kansa Taisteli -lehtien osalta. Tähän ongelmaan voidaan tarttua tekstin korjaussäännöillä, joita yleensä muotoillaan säännöllisten lausekkeiden avulla. Pulmana on vielä tunnistaa ne lukuisat kielelliset tavat, joilla esimerkiksi henkilöihin voidaan viitata. Esimerkiksi Klaus Lennart Oeschiin viitataan Sotasammon aineistoissa ainakin seuraavilla muodoilla:

1. kenraali Oesch
2. kenraaliluutnantti Oesch
3. kenraaliluutnantti K.L. Oesch
4. kenr.luutn. Oesch
5. kenraali Svensson, Mäkinen ja Oesch
6. kenraalit Walden, Oesch, Lundqvist ja Talvela

Myös tällaisten muotojen tunnistamiseen tarjoavat säännölliset lausekkeet yhden lähestymistavan, mutta tuloksena voi olla hyvinkin mutkikkaita muotoiluja. Esimerkiksi viimeinen kenraalilistamuoto voidaan tunnistaa alla olevalla lausekkeella:

```
regex = '(?:[Kk]enraali(?:t)?(?:)?\W+)' + \
        '(?:([A-ZÄÖÅ]\w+)(?:,\W*))?' * 10 + \
        '(?:([A-ZÄÖÅ]\w+)?(?:\W+ja\W+)?([A-ZÄÖÅ]\w+)?)?'
```

Mutkikkaita sääntöjä on vaikea muotoilla ja ylläpitää, ja kun niiden ja aineiston määrä kasvaa, käy helposti niin, että joku sääntö laukeaa väärässä kontekstissa ja johtaa virheeseen.

Sotasammossa linkitettiin lehtiaineistojen yli 3300 artikkelia henkilöihin, organisaatioihin ja paikkoihin sekä Wikipedian käsitteisiin. Lisäksi analysointiin portaalin yli 1000 merkittävän historiallisen tapahtuman ja 160 000 valokuvan kuvailutekstit ja linkitettiin ne henkilöihin, organisaatioihin ja historiallisiin paikkoihin. Esimerkiksi talvisodan tapahtuman

6.12.1939 *Ylipäällikkö muodosti eversti P. Talvelan johtoon Tolvajärven ja Ilomantsin suunnissa taistelevista ja sinne keskitettävistä joukoista Osasto Talvelan.*

järjestelmä linkitti seuraavasti:

Teksti	**Entiteetti (IRI)**	**Ontologia**
Ylipäällikkö	C. G. Mannerheim	Henkilöt
eversti P. Talvelan	kenraali Paavo Talvela	Henkilöt
Tolvajärven	Tolvajärvi	Paikat
Ilomantsin	Ilomantsi	Paikat
Osasto Talvelan	Osasto Talvela	Joukko-osastot

Taulukko 18.1: Tapahtuman tekstikuvauksesta tunnistetut entiteetit ja niiden linkitys

Sissipataljoona 2. hiihtoammuntakilpailusta: Tässä ottaa kilpailun voittaja kersantti Leskinen komeasti mäen. Hän hiihti parhaan ajan ja ampui täydet pisteet. Eikä ihmekään sillä hän on siviilissä Pohjois-Savon Nuorten mestari.

7.2.1942

Kuva 18.3: Sotasammossa oleva SA-valokuva 7.2.1942 ja sen kuvateksti

Tarkastellaan toisena esimerkkinä kuvan 18.3 valokuvaa ja sen kuvatekstiä. Ongelmana on: viittaako tekstissä mainittu *kersantti Leskinen* jo-

honkin Sotasammon tuntemaan henkilöön ja jos, niin keneen? Ongelma on haastava ja sen ratkaiseminen vaatii sovellusaluekohtaista tietämystä, jota Sotasampoon on jossain määrin ohjelmoitu. Tässä tapauksessa Sotasammon annotointialgoritmi löytää semanttisesta verkosta kolme mahdollista kersantti Leskistä. Näistä yksi, Reino Leskinen palveli tekstissä mainitussa 2. Sissikomppaniassa, joten tämä tulkinta näyttäisi todennäköiseltä. Menehtyneet tietokannan mukaan Reino Leskinen kuitenkin kuoli juuri samana päivänä. Vaikka on mahdollista, että näin tosiaankin kävi heti kilpailun jälkeen, voi myös olla niin, että kuvassa on silti joku muu kersantti Leskinen, joka ei vaan satu olemaan mukana Sotasammon henkilöontologiassa. Henkilöontologiaan kuuluu tätä kirjoitettaessa menehtyneiden lisäksi vain joitain tuhansia muita elossa selvinneitä sotilaita ja poliitikkoja, joihin on viitattu eri aineistoissa.

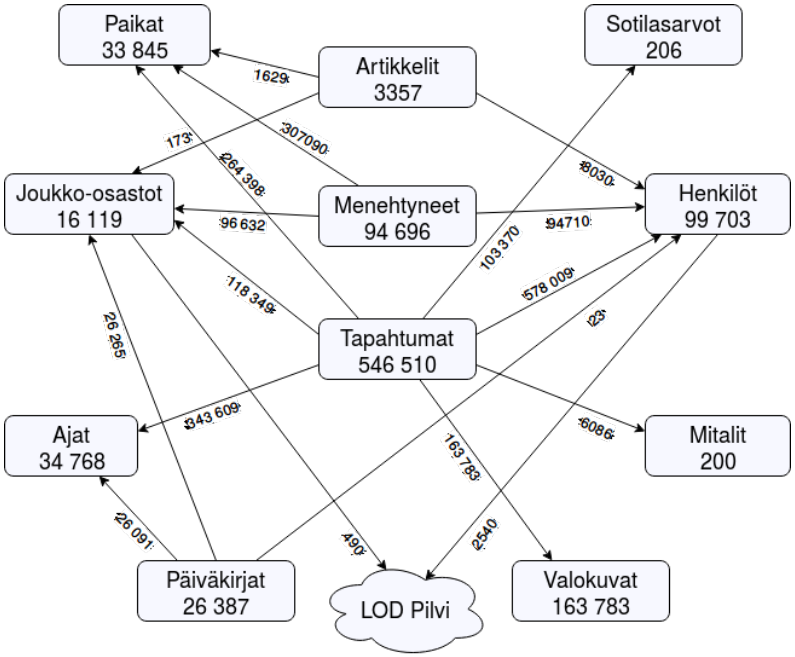
Automaattisesti tuotetun linkityksen laadun testaamiseksi linkityksiä on tarkastettu käsityönä pienillä 50–100 kohteen satunnaisotoksilla. Tarkkuus ja saanti näyttäisivät niiden perusteella olevan kohtuullista 70% luokkaa tehtävän haasteellisuudesta huolimatta. Virheitäkin siis kuitenkin esiintyy, ja koska tieto esimerkiksi eri henkilöistä ei ole täydellistä jää tulkintoihin tästä johtuen epävarmuutta. Lähdekriittisyys on paikallaan tällaisissa semanttisen webin sovelluksissa, joissa aineistot ovat laajoja ja joudutaan luottamaan koneen automaattisesti tekemiin päätöksiin ilman mahdollisuutta kattavaan tarkastukseen ja korjailuun käsityönä. Osittainkin oikea tulos voi kuitenkin olla parempi kuin ei lainkaan tulosta.

Kuva 18.4 esittää datan tuotannon ja linkityksen lopputuloksena syntynyttä Sotasammon linkitettyjen datajoukkojen pilveä. Suorakaiteen muotoiset solmut esittävät eri aineistoista luotuja RDF-graafeja ja kertovat niihin sisältyvien yksilöiden määrät. Datajoukkojen väliset kaaret kuvaavat datajoukkojen välisten linkkien lukumäärää ja suuntaa datajoukkojen yksilöistä toiseen. Data linkittyy myös Sotasammon ulkopuolelle Wikipediaan ja Kansallisbiografiaan.

18.5 Portaalisovellus Sotasampo.fi

Sotasampo-portaali[5] koostuu joukosta toisiinsa linkitettyjä *näkymäsovelluksia* (application perspective), jotka tarjoavat erilaisia näkökulmia so-

[5] http://sotasampo.fi

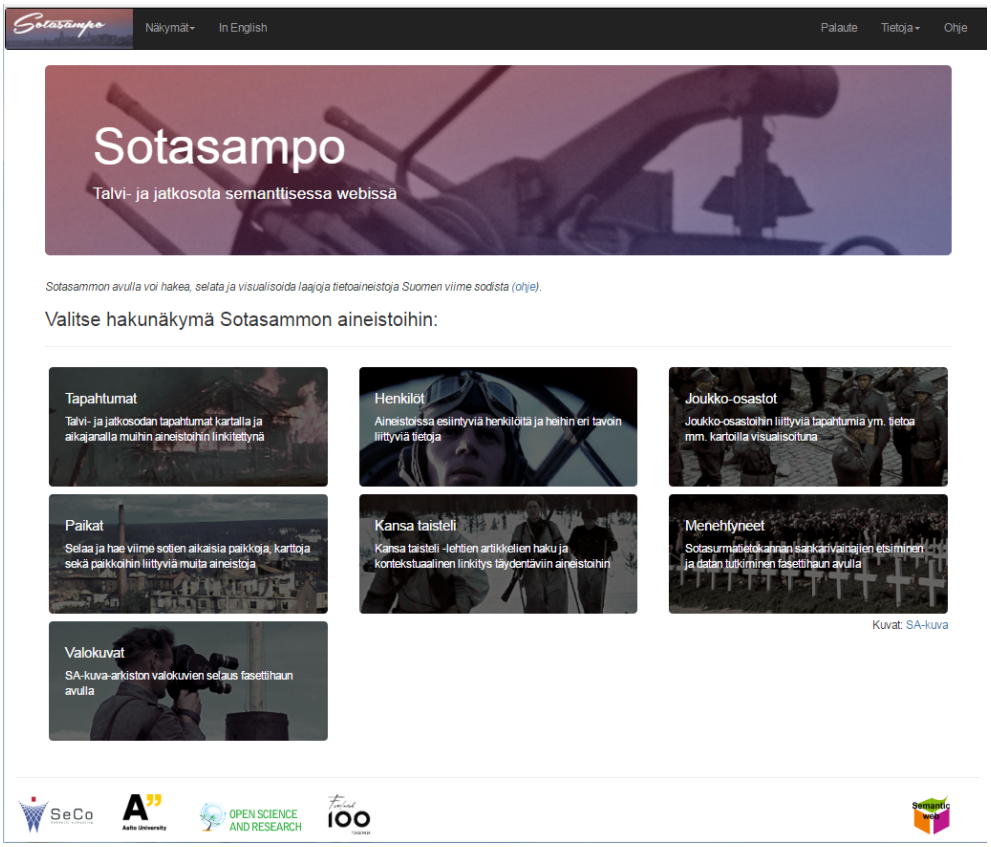


Kuva 18.4: Sotasammon datajoukot v. 2017 alussa, niiden luokkien yksilöiden lukumäärät, ja joukkojen yksilöiden välisten linkkien määrät ja suunta

dan aineistoihin. Kuvassa 18.5 näkyvät palvelun etusivulla olevat linkit eri näkymiin, jotka ovat Tapahtumat, Henkilöt, Joukko-osastot, Paikat, Kansa taisteli, Menehtyneet ja Valokuvat. Seuraavassa tarkastellaan lyhyesti näiden näkymien toimintaa.

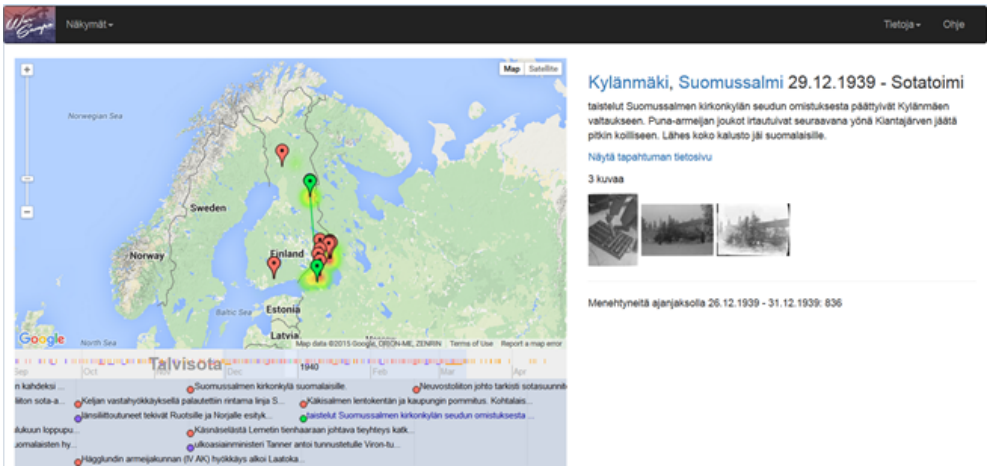
Tapahtumat-näkymän pääsivu on kuvassa 18.6. Näkymä esittää talvi- ja jatkosodan poliittisia ja sotilaallisia tapahtumia aikajanalla ja viiden päivän aikaikkunaan sisältyvät tapahtumat kartalla. Taustalla näkyvä lämpökartta lasketaan valittuna aikaikkunan aikana menehtyneistä sotilaista sotasurmadatan perusteella. Ikkunaa ajassa siirtämällä muuttuvat tapahtumat ja sotasurmien määrää ilmaisevan lämpökartan väritys (punainen-keltainen-vihreä), mikä antaa yleiskuvan tapahtumien kehittymisestä ajan ja paikan suhteen.

Käyttäjä voi klikkaamalla valita tapahtuman aikajanalta tai kartalta. Kuvassa on valittu tapahtuma ”taistelut Suomussalmen kirkonkylän seudun omistuksesta . . . ”, jolloin siihen liittyvä data ja semanttiset linkit muihin



Kuva 18.5: Sotasammon sovellusnäkymät, joista jokainen on oma datapalvelun SPARQL-rajapintaan perustuva sovelluksensa.

aineistoihin näytetään oikealla. Tapahtuman kuvauksessa on seuraavat toiminnot: 1) ”Kylänmäki” ja ”Suomussalmi” ovat linkkejä Sotasammon Paikat-näkymän paikkojen tietosivuille, joilla voi katsella mm. paikkoja historiallisilla kartoilla sekä paikkoihin eri tavoin linkittyvää tietoa. Sekä historialliset kartat että muu tieto tulevat jälleen toisista datajoukoista. 2) Linkki ”Näytä tapahtuman tietosivu” johtaa tapahtuman omalle ”kotisivulle”, jossa esitellään eri tavoin tapahtumiin loogisesti liittyviä muita tietoja, esimerkiksi kootusti eri aikoina samalla paikalla käytyjä taisteluita ja muita tapahtumia tai henkilöitä, jotka osallistuivat ko. tapahtumaan. Tällaisia semanttisia suositteluita voidaan luoda joustavasti SPARQL-rajapinnan kyselyiden avulla. 3) Lisäksi tapahtumatietoon on



Kuva 18.6: Sotasammon Tapahtumat-sovellusnäkymä.

valmiiksi linkitetty puolustusvoimien SA-kuva-arkiston kuvia, jotka on otettu samassa paikassa ja samaan aikaan kuin valitun tapahtuman tiedetään tapahtuneen.

Ideana on siis rikastaa eri datajoukkojen sisältöjä toistensa avulla, jolloin kaikkien julkaisijoiden dataa voidaan rikastaa yhteistyöllä ”ilmaiseksi” ja yhteisöllisen datajulkaisun luomiselle syntyy hyvä motivaatio. Esimerkiksi Henkilöt-näkymän kautta voidaan sotilaan joukko-osastotietoa yhdistää organisaatiokorteista saatavaan tietoon joukko-osaston tapahtumista sekä joukko-osaston sotapäiväkirjoihin, jolloin voidaan rakentaa ja visualisoida kartalla paitsi joukko-osastojen niin myös yksittäisten sotilaiden liikkeitä ja sotahistoriaa. Eri sovellusten aineistojen tietosivuihin voidaan viitata systemaattisesti yhteisesti sovittujen URI-tunnisteiden avulla muodossa:

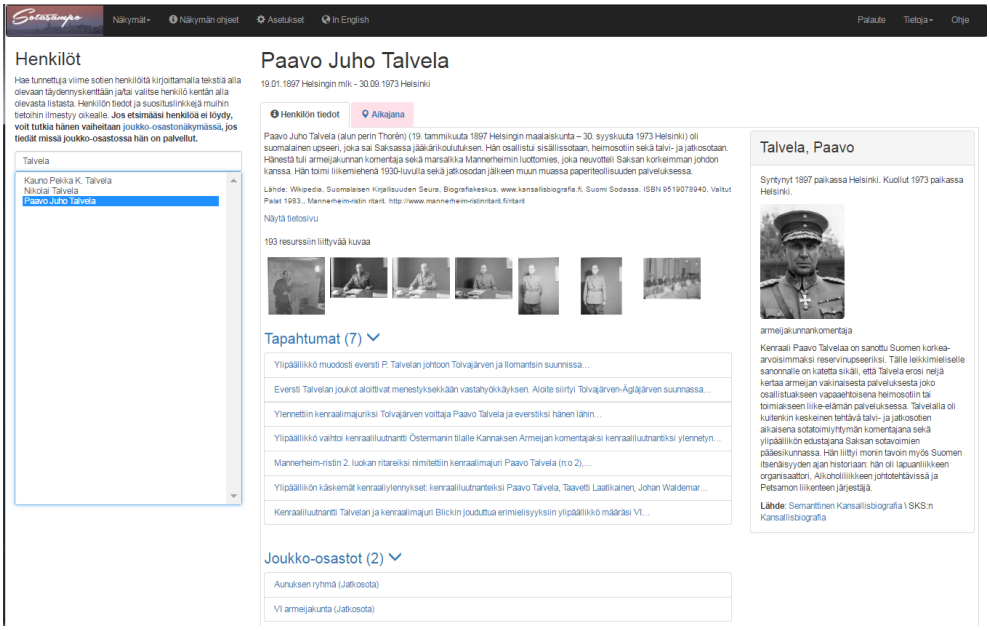
`http://www.sotasampo.fi/`*`SOVELLUS`*`/page?uri=URI`

Esimerkiksi Suomussalmen tietosivu (kotisivu) Paikat-sovelluksen (*SOVELLUS*=`places`) tuottamana palveluna on osoitteessa:

http://ww.sotasampo.fi/places/page?uri=http://ldf.fi/warsa/places/municipalities/m_place_87

Vastaavasti on systemaattisesti luotu kotisivut muillekin Sotasammon keskeisten aineistoluokkien yksilöille, kuten henkilöille ja joukko-osastoille. Tämä mekanismi mahdollistaa tietosivujen uudelleenkäytön

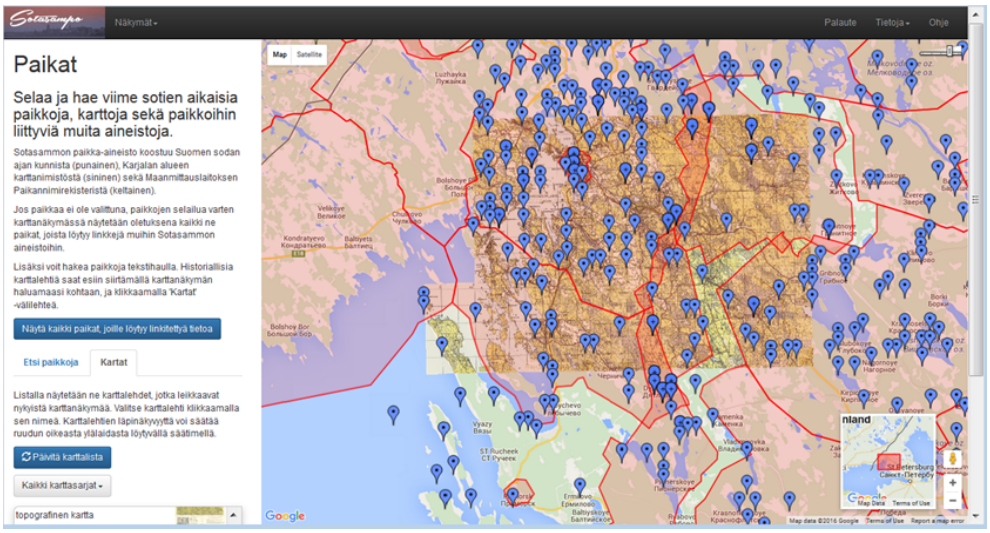
paitsi Sotasampo.fi:ssä niin myös missä tahansa toisissa sovelluksissa HTML-muodossa, joka voidaan näyttää selaimessa.



Kuva 18.7: Sotasammon Henkilöt-näkymän kautta löytynyt Paavo Talvelan kotisivu.

Henkilöt-näkymän kautta voi tutustua Sotasammon aineistoihin hakemalla henkilöiden kotisivuja ja heihin eri tavoin liittyviä aineistoja. Kuvassa 18.7 käyttäjä on kirjoittanut vasemmalla olevaan hakukenttään *Talvela* ja valinnut tuloslistasta *Paavo Juhani Talvelan*, jonka kotisivu näkyy oikealla *Henkilön tiedot* -välilehti avattua. Sivulle linkittyvät Paavo Talvelaan valokuvat (yhteensä 193), linkit historiallisiin tapahtumiin, joissa hän oli mukana ja oikealla elämäkerrallista tietoa SKS:n Kansallisbiografiasta. Kuvakaappauksen ulkopuolelle jäävät linkit hänen joukko-osastojensa kotisivuille, ylennystapahtumat, joista viimeinen on ylennys jalkaväen kenraaliksi 6.12.1966, tieto hänelle myönnetyistä kunniamerkeistä, linkkejä häneen liittyviin paikkoihin, Wikipediaan ja Kansa Taisteli -artikkeleihin, joissa hänet mainitaan. *Aikajana*-välilehdellä visualisoidaan Paavo Talvelaan liittyvät tapahtumat aikajanalla ja kartalla vastaavaan tapaan kuin historialliset tapahtumat Tapahtumat-näkymässä edellä.

Vastaavanlainen visualisointi on haettavissa myös *Joukko-osastot*-näkymän kautta Sotasammon tuhansille joukko-osastoille. Tätä kautta löytyvät linkitettynä myös joukko-osastoihin liittyvät autenttiset sotapäiväkirjat. Joukko-osastojen historian kautta voi selvittää siihen kuuluneiden henkilöiden liikkeitä sodassa, vaikka henkilöstä ei olisi muuta tietoa Sotasammossa. Suurimmalla osalla sodasta selvinneistä sotilaista ei ole omaa kotisivua, sillä Sotasammon tärkeimpänä tietolähteenä on sodissa menehtyneiden tietokanta, jota on täydennetty n. 5000 eri lähteissä mainitulla tunnetulla sotilaalla. Elossa selvinneistä rivimiehistä ei ainakaan vielä ole käytettävissä tarkempia tietoa kuin noissa tapauksissa.



Kuva 18.8: Sotasammon paikka-sovelluksen kuvakaappaus Viipurin alueen paikoista, joihin liittyy tapahtumia.

Paikat-näkymän kautta sodan tapahtumat ja aineistot löytää karttojen kautta (ks. kuva 18.8). Kartoilla näkyviä merkkejä klikkaamalla avautuu ikkuna paikkaan liittyviin tapahtumiin. Modernien Googlen karttojen ohella käytettävissä on näiden päälle läpinäkyvästi asemoidut historialliset Karjalan kartat (kuvassa keskellä näkyy näistä yksi), joilla vanhat sotien aikaiset paikannimet vielä löytyvät.

Kansa taisteli -näkymä tarjoaa semanttiseen fasettihakuun perustuvan hakukäyttöliittymän Kansa taisteli -lehtien artikkeleihin. Kun kiinnostava artikkeli löytyy, voi siihen tutustua paitsi lukemalla myös lisälinkkien

kautta, joita luodaan automaattisesti tekstissä esiintyvistä henkilönimistä, paikannimistä ja muista käsitteistä. Linkit johtavat Sotasammon vastaaville kotisivuille ja Wikipediaan.

Menehtyneet-näkymän kautta voi semanttisen fasettihaun avulla selata sodissa menehtyneiden henkilöiden tietokantaa. Muutamalla fasettivalinnalla voi löytää esimerkiksi Helsingissä syntyneet kaatuneet lesket ja nähdä, kuinka paljon heillä oli orvoksi jääneitä lapsia Lasten lukumäärä -fasetin kautta. *Valokuvat*-näkymässä voi hakea fasettivalinnoillaa vastaavaan tapaan sodanaikaisia valokuvia esimerkiksi henkilöiden ja paikkojen avulla aineistoa suodattaen.

18.6 Datapalvelu Sotasampo

Kaikki Sotasammon kuvan 18.4 datajoukot ovat käytettävissä avoimella Linked Data Finland -palvelualustalla (LDF.fi), jonka SPARQL-rajapinnan käyttöön myös kaikki Sotasampo.fi-portaalin eri näkymäsovellukset suoraan perustuvat. Lisää dataa julkaistaan LDF.fi-palvelussa avoimen datan lisensseillä sitä mukaa, kun niiden toimivuus testataan ja data dokumentoidaan.

Sotasammon datajoukkojen julkaisu alkoi Kansallisarkiston kolmen sotasurmatietokannan avaamisella, joista Sotasampo käyttää talvi- ja jatkosodassa menehtyneiden sotilaiden tietokannasta muodostettua datajoukkoa ja avointa datapalvelua. Sen kotisivu on LDF.fi:ssä osoitteessa:

`http://www.ldf.fi/dataset/narc-menehtyneet1939-45`

LDF.fi-palvelussa eri datajoukkojen SPARQL-rajapintapalvelu löytyy LDF.fi-palvelussa aina osoitteessa, jonka muoto on:

`http://ldf.fi/`*`SERVICE`*`/sparql`

Tässä *SERVICE* on datajoukon nimi. Kirjoittamalla rajapinnan osoite selaimeen, kuten

`http://ldf.fi/narc-menehtyneet1939-45/sparql`

pääsee käyttämään rajapintaa interaktiivisesti LDF.fi-palveluun asennetun SPARQL-editori YASGUIn avulla.

Sotasammon datajulkaisut on koottu omaan uuteen palveluun nimeltä *warsa* (WarSampo), jossa eri datajoukot ovat mukana omina RDF-graafeinaan:

```
http://ldf.fi/warsa/sparql
```

Palvelun oletuskysely YASGUI-editorissa hakee eri graafit, laskee niiden koot kolmikkoina sekä näyttää esimerkkinä niistä yhden kolmikon. Sen resurssilinkkien kautta voi tutustua dataan selailemalla.

18.7 Uutuusarvo ja jatkokehitys

Vaikka toiseen maailmansotaan liittyviä aineistoja on julkaistu verkossa runsaasti, oli Sotasampo tiettävästi ensimmäinen datapalvelu maailmassa, jossa aineistojen data on saatavilla myös linkitettynä avoimena datana. Sotasampo.fi-sovelluksen innovatiivisuus liittyy sen arkkitehtuuriin, jossa eri datasiilojen tietoa tarjotaan käyttäjille erilaisten toisiaan semanttisesti rikastavien näkymien kautta. Sisällöt on harmonisoitu yhteentoimiviksi yhteisten tapahtuma- ja muiden ontologioiden avulla. Linkitetty data perustuu joukkoon W3C:n standardoimia semanttisia teknologioita ja käytäntöjä (best practices), joiden avulla voidaan julkaista laajoja tietosisältöjä siten, että ne yhdistyvät toisiinsa käsitteellisellä tasolla. Näin muodostuva webin datapilvi, englanniksi ”Web of Data”, on kustannustehokkaasti hyödynnettävissä sovelluksissa sekä hyvin määritellyn standardoidun RDF-perustaisen (Resource Description Framework) rakenteensa että rajapintojensa (SPARQL, REST ym.) kautta. Linked Data -teknologia on keskeisessä asemassa monissa Digital Humanities -alueen hankkeissa, jollainen Sotasampokin on.

Sotasampo-pilotin pidemmän aikavälin tavoitteena on käynnistää prosessi sotahistoriallisen tiedon pysyvästä tuotannosta ja ylläpidosta linkitettynä datana osana jonkun muistiorganisaation työtä. Datan ylläpito on kuitenkin haasteellisempaa uudessa semanttisen webin toimintaympäristössä, jossa ei riitä vain yhden organisaation oman tietokannan ylläpito, vaan huomioon on otettava laajempi ympäristö ja muut verkoston jäsenet. Tällä hetkellä esimerkiksi Menehtyneet-tietokantaa ylläpidetään Kansallisarkistossa erillisessä tietokannassa. Kun dataa päivitetään esimerkiksi asiakaspalautteen perusteella, eivät muutokset siirry automaattisesti Sotasampoon, vaan välissä tarvitaan muunnoksia ja mahdollisesti käsityötä. Lisäksi jotkin muutokset, kuten vaikka uuden historiallisen paikan tai henkilön lisääminen, voi olla kriittistä myös muiden organisaatioiden tietokantojen ylläpidon kannalta, ja tieto muutoksesta ja uusista

tunnisteista pitäisi saada reaaliajassa kaikkien niitä tarvitsevien tietoon. Ratkaisumallina tähän olisi Sotasammon datapilven keskitetty ylläpito ja julkaiseminen suoraan RDF-muodossa, mutta tämä vaatisi jatkossa muistiorganisaatioiden resursointia ja yhteistyön nykyistä tiiviimpää koordinointia.

18.8 Kirjallisuutta

Sotasampo toteutettiin Aalto-yliopistossa joukkona toisiinsa liittyviä opinnäytetöitä, joihin tämä lukukin perustuu. Työt käsittelevät järjestelmän perustana olevaa tapahtumaperustaista mallia ja aineistojen linkitystä [18], sotilashenkilöiden ja joukko-osastojen ontologista mallintamista, hakua ja suosittelua [33], Kansa Taisteli -lehtien automaattisista linkitystä ja hakua [48] sekä historiallisten paikkojen ontologioita ja karttapalvelua [29]. Tuloksista on raportoitu kansainvälisissä tieteellisissä konferensseissa [22, 31, 19, 49, 34]. Kesällä 2017 Sotasampo sai arvostetun LODLAM Challenge Open Data Prize -palkinnon Venetsiassa.

Liite A

Esimerkki OWL-ontologiasta

Alla W3C:n suosituksessa[1] esitetty laajempi esimerkki ontologia-dokumentista, jossa on määritelty erilaisia sukulaisuussuhteita.

```
Prefix(:=<http://example.com/owl/families/>)
Prefix(otherOnt:=<http://example.org/otherOntologies/families/>)
Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
Ontology(<http://example.com/owl/families>
  Import( <http://example.org/otherOntologies/families.owl> )

  Declaration( NamedIndividual( :John ) )
  Declaration( NamedIndividual( :Mary ) )
  Declaration( NamedIndividual( :Jim ) )
  Declaration( NamedIndividual( :James ) )
  Declaration( NamedIndividual( :Jack ) )
  Declaration( NamedIndividual( :Bill ) )
  Declaration( NamedIndividual( :Susan ) )
  Declaration( Class( :Person ) )
  AnnotationAssertion( rdfs:comment :Person "Represents the set of all people." )
  Declaration( Class( :Woman ) )
  Declaration( Class( :Parent ) )
  Declaration( Class( :Father ) )
  Declaration( Class( :Mother ) )
  Declaration( Class( :SocialRole ) )
  Declaration( Class( :Man ) )
  Declaration( Class( :Teenager ) )
  Declaration( Class( :ChildlessPerson ) )
  Declaration( Class( :Human ) )
  Declaration( Class( :Female ) )
  Declaration( Class( :HappyPerson ) )
  Declaration( Class( :JohnsChildren ) )
  Declaration( Class( :NarcisticPerson ) )
  Declaration( Class( :MyBirthdayGuests ) )
  Declaration( Class( :Dead ) )
  Declaration( Class( :Orphan ) )
```

[1] http://www.w3.org/TR/2012/REC-owl2-primer-20121211/

```
Declaration( Class( :Adult ) )
Declaration( Class( :YoungChild ) )
Declaration( ObjectProperty( :hasWife ) )
Declaration( ObjectProperty( :hasChild ) )
Declaration( ObjectProperty( :hasDaughter ) )
Declaration( ObjectProperty( :loves ) )
Declaration( ObjectProperty( :hasSpouse ) )
Declaration( ObjectProperty( :hasGrandparent ) )
Declaration( ObjectProperty( :hasParent ) )
Declaration( ObjectProperty( :hasBrother ) )
Declaration( ObjectProperty( :hasUncle ) )
Declaration( ObjectProperty( :hasSon ) )
Declaration( ObjectProperty( :hasAncestor ) )
Declaration( ObjectProperty( :hasHusband ) )
Declaration( DataProperty( :hasAge ) )
Declaration( DataProperty( :hasSSN ) )
Declaration( Datatype( :personAge ) )
Declaration( Datatype( :minorAge ) )
Declaration( Datatype( :majorAge ) )
Declaration( Datatype( :toddlerAge ) )

SubObjectPropertyOf( :hasWife :hasSpouse )
SubObjectPropertyOf(
  ObjectPropertyChain( :hasParent :hasParent )
  :hasGrandparent
)
SubObjectPropertyOf(
  ObjectPropertyChain( :hasFather :hasBrother )
  :hasUncle
)
SubObjectPropertyOf(
  :hasFather
  :hasParent
)

EquivalentObjectProperties( :hasChild otherOnt:child )
InverseObjectProperties( :hasParent :hasChild )
EquivalentDataProperties( :hasAge otherOnt:age )
DisjointObjectProperties( :hasSon :hasDaughter )
ObjectPropertyDomain( :hasWife :Man )
ObjectPropertyRange( :hasWife :Woman )
DataPropertyDomain( :hasAge :Person )
DataPropertyRange( :hasAge xsd:nonNegativeInteger )

SymmetricObjectProperty( :hasSpouse )
AsymmetricObjectProperty( :hasChild )
DisjointObjectProperties( :hasParent :hasSpouse )
ReflexiveObjectProperty( :hasRelative )
IrreflexiveObjectProperty( :parentOf )
FunctionalObjectProperty( :hasHusband )
InverseFunctionalObjectProperty( :hasHusband )
TransitiveObjectProperty( :hasAncestor )
FunctionalDataProperty( :hasAge )

SubClassOf( :Woman :Person )
SubClassOf( :Mother :Woman )
SubClassOf(
  :Grandfather
  ObjectIntersectionOf( :Man :Parent )
)
SubClassOf(
```

```
  :Teenager
  DataSomeValuesFrom( :hasAge
    DatatypeRestriction( xsd:integer
      xsd:minExclusive "12"^^xsd:integer
      xsd:maxInclusive "19"^^xsd:integer
    )
  )
)
SubClassOf(
  Annotation( rdfs:comment "States that every man is a person." )
  :Man
  :Person
)
SubClassOf(
  :Father
  ObjectIntersectionOf( :Man :Parent )
)
SubClassOf(
  :ChildlessPerson
  ObjectIntersectionOf(
    :Person
    ObjectComplementOf(
      ObjectSomeValuesFrom(
        ObjectInverseOf( :hasParent )
        owl:Thing
      )
    )
  )
)
SubClassOf(
  ObjectIntersectionOf(
    ObjectOneOf( :Mary :Bill :Meg )
    :Female
  )
  ObjectIntersectionOf(
    :Parent
    ObjectMaxCardinality( 1 :hasChild )
    ObjectAllValuesFrom( :hasChild :Female )
  )
)

EquivalentClasses( :Person :Human )
EquivalentClasses(
  :Mother
  ObjectIntersectionOf( :Woman :Parent )
)
EquivalentClasses(
  :Parent
  ObjectUnionOf( :Mother :Father )
)
EquivalentClasses(
  :ChildlessPerson
  ObjectIntersectionOf(
    :Person
    ObjectComplementOf( :Parent )
  )
)
EquivalentClasses(
  :Parent
  ObjectSomeValuesFrom( :hasChild :Person )
)
```

```
EquivalentClasses(
  :HappyPerson
  ObjectIntersectionOf(
     ObjectAllValuesFrom( :hasChild :HappyPerson )
     ObjectSomeValuesFrom( :hasChild :HappyPerson )
  )
)
EquivalentClasses(
  :JohnsChildren
  ObjectHasValue( :hasParent :John )
)
EquivalentClasses(
  :NarcisticPerson
  ObjectHasSelf( :loves )
)
EquivalentClasses(
  :MyBirthdayGuests
  ObjectOneOf( :Bill :John :Mary)
)
EquivalentClasses(
  :Orphan
  ObjectAllValuesFrom(
    ObjectInverseOf( :hasChild )
    :Dead
  )
)
EquivalentClasses( :Adult otherOnt:Grownup )
EquivalentClasses(
  :Parent
  ObjectSomeValuesFrom(
    :hasChild
    :Person
  )
)

DisjointClasses( :Woman :Man )
DisjointClasses(
  :Mother
  :Father
  :YoungChild
)
HasKey( :Person () ( :hasSSN ) )

DatatypeDefinition(
  :personAge
  DatatypeRestriction( xsd:integer
    xsd:minInclusive "0"^^xsd:integer
    xsd:maxInclusive "150"^^xsd:integer
  )
)
DatatypeDefinition(
  :minorAge
  DatatypeRestriction( xsd:integer
    xsd:minInclusive "0"^^xsd:integer
    xsd:maxInclusive "18"^^xsd:integer
  )
)
DatatypeDefinition(
  :majorAge
  DataIntersectionOf(
    :personAge
```

```
      DataComplementOf( :minorAge )
    )
  )
  DatatypeDefinition(
    :toddlerAge
    DataOneOf( "1"^^xsd:integer "2"^^xsd:integer )
  )

  ClassAssertion( :Person :Mary )
  ClassAssertion( :Woman :Mary )
  ClassAssertion(
    ObjectIntersectionOf(
      :Person
      ObjectComplementOf( :Parent )
    )
    :Jack
  )
  ClassAssertion(
    ObjectMaxCardinality( 4 :hasChild :Parent )
    :John
  )
  ClassAssertion(
    ObjectMinCardinality( 2 :hasChild :Parent )
    :John
  )
  ClassAssertion(
    ObjectExactCardinality( 3 :hasChild :Parent )
    :John
  )
  ClassAssertion(
    ObjectExactCardinality( 5 :hasChild )
    :John
  )
  ClassAssertion( :Father :John )
  ClassAssertion( :SocialRole :Father )

  ObjectPropertyAssertion( :hasWife :John :Mary )
  NegativeObjectPropertyAssertion( :hasWife :Bill :Mary )
  NegativeObjectPropertyAssertion(
    :hasDaughter
    :Bill
    :Susan
  )
  DataPropertyAssertion( :hasAge :John "51"^^xsd:integer )
  NegativeDataPropertyAssertion( :hasAge :Jack "53"^^xsd:integer )

  SameIndividual( :James :Jim )
  SameIndividual( :John otherOnt:JohnBrown )
  SameIndividual( :Mary otherOnt:MaryBrown )
  DifferentIndividuals( :John :Bill )
)
```

Kirjallisuutta

[1] David Allemang and Jim Hendler. *Semantic Web for the working Ontologiest*. Morgan Kaufmann, San Francisco, 2008.

[2] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services: Concepts, Architectures and Applications*. Springer–Verlag, 2003.

[3] Paul Anderson. *Web 2.0 and Beyond: Principles and Technologies*. Chapman and Hall/CR, USA, 2012.

[4] Grigoris Antoniu, Frank van Harmelen, and Rinke Hoekstra. *Semantic Web Primer (3rd Edition)*. The MIT Press, Cambridge, Massachusetts, 2012.

[5] Murtha Baca, editor. *Introduction to metadata*. Getty Publications, Los Angeles, 2008.

[6] T. Berners-Lee, M. Fischetti, and M. Dertouzos. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*. Harper Business, 2000.

[7] Ronald Brachman and Hector Levesque. *Knowledge representation and reasoning*. Morgan Kaufmann, San Francisco, 2004.

[8] Ivan Bratko. *Prolog programming for Artificial Intelligence (4th edition)*. Pearson, Harlow, U.K., 2012.

[9] Megan R. Brett. Topic modeling: A basic introduction. *Journal of Digital Humanities*, 2(1), 2012.

[10] Nick Crofts, Martin Doerr, Tony Gill, Stephen Stead, and Matthew Stiff (Eds.), editors. *Definition of the CIDOC Conceptual Reference Model, Version 5.0.4.* ICOM/CIDOC Documentation Standards Group (CIDOC CRM Special Interest Group), 2011. http://www.cidoc-crm.org/docs/cidoc_crm_version_5.0.4.pdf.

[11] Martin Doerr. The CIDOC CRM—an ontological approach to semantic interoperability of metadata. *AI Magazine*, 24(3):75–92, 2003.

[12] Martin Doerr. Ontologies for cultural heritage. In Staab and Studer [44], pages 463–486.

[13] J. Euzenat and P. Shvaiko. *Ontology Matching.* Springer–Verlag, 2007.

[14] Andy Field, Jeremy Miles, and Zoe Field. *Discovering Statistics Using R.* SAGE Publications Inc., USA, 2015.

[15] A. J. Gilliland. Setting the stage. In Baca [5], pages 1–19.

[16] Nicola Guarino, Daniel Oberle, and Steffen Staab. What is an ontology? In Staab and Studer [44], pages 1–17.

[17] D. Hardman and L. Edwards. Lost in hyperspace: Cognitive mapping and navigation in a hypertext environment. *Hypertext: Theory into practice*, pages 105–145, 1989.

[18] Erkki Heino. Sotahistorian kuvaaminen ja rikastaminen linkitettynä datana. Master's thesis, University of Helsinki, Dept. of Computer Science, Finland, 2017.

[19] Erkki Heino, Minna Tamper, Eetu Mäkelä, Petri Leskinen, Esko Ikkala, Jouni Tuominen, Mikko Koho, and Eero Hyvönen. Named entity linking in a complex domain: Case second world war history. In *Proceedings, Language, Technology and Knowledge (LDK 2017)*, pages 120–133. Springer–Verlag, 2017.

[20] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web technologies.* Springer–Verlag, 2010.

[21] Eero Hyvönen. *Publishing and Using Cultural Heritage Linked Data on the Semantic Web*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool, Palo Alto, USA, 2012.

[22] Eero Hyvönen, Erkki Heino, Petri Leskinen, Esko Ikkala, Mikko Koho, Minna Tamper, Jouni Tuominen, and Eetu Mäkelä. WarSampo Data Service and Semantic Portal for Publishing Linked Open Data about the Second World War History. In *The Semantic Web – Latest Advances and New Domains (ESWC 2016)*, pages 758–773. Springer–Verlag, 2016.

[23] Eero Hyvönen, Ilkka Karanta, and Markku Syrjänen, editors. *Tekoälyn ensyklopedia*. Gaudeamus, 1993.

[24] Eero Hyvönen, Petri Leskinen, Minna Tamper, Jouni Tuominen, and Kirsi Keravuori. Semantic National Biography of Finland. In *Digital Humanities in Nordic Countries, 3rd Conference*, 2017. Submitted, available at http://seco.cs.aalto.fi/publications/.

[25] E. Hyvönen, E. Mäkela, M. Salminen, A. Valo, K. Viljanen, S. Saarela, M. Junnila, and S. Kettula. MuseumFinland—Finnish museums on the semantic web. *Journal of Web Semantics*, 3(2):224–241, 2005.

[26] E. Hyvönen, K. Viljanen, E. Mäkelä, T. Kauppinen, T. Ruotsalo, O. Valkeapää, K. Seppälä, O. Suominen, O. Alm, R. Lindroos, T. Känsälä, R. Henriksson, M. Frosterus, J. Tuominen, R. Sinkkilä, and J. Kurki. Elements of a national semantic web infrastructure—case study Finland on the semantic web (invited paper). In *Proceedings of the First International Semantic Computing Conference (IEEE ICSC 2007), Irvine, California*, pages 216–223. IEEE Press, Sept 2007.

[27] Eero Hyvönen, editor. *Semantic Web Kick-Off in Finland*, number 2002-001 in HIIT Publications. Helsinki Institute for Information Technology (HIIT), Helsinki, Finland, May 2002. http://seco.cs.aalto.fi/publications/2002/hyvonen-semantic-web-kick-off-2002.pdf.

[28] Eero Hyvönen, Eetu Mäkelä, Tomi Kauppinen, Olli Alm, Jussi Kurki, Tuukka Ruotsalo, Katri Seppälä, Joeli Takala, Kimmo Puputti, Heini Kuittinen, Kim Viljanen, Jouni Tuominen, Tuomas Palo-

nen, Matias Frosterus, Reetta Sinkkilä, Panu Paakkarinen, Joonas Laitio, and Katariina Nyberg. CultureSampo – Finnish culture on the Semantic Web 2.0. Thematic perspectives for the end-user. In *Museums and the Web 2009, Proceedings*. Archives and Museum Informatics, Toronto, 2009.

[29] Esko Ikkala. Suomalainen historiallisten paikkojen ja karttojen ontologiapalvelu. Master's thesis, Aalto University, Dept. of Computer Science, Finland, 2016.

[30] Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia. *Learning Spark: Lightning-Fast Big Data Analysis*. O'Reilly, USA, 2015.

[31] Mikko Koho, Eero Hyvönen, Erkki Heino, Jouni Tuominen, Petri Leskinen, and Eetu Mäkelä. Linked Death - Representing, Publishing, and Using Second World War Death Records as Linked Open Data. In *The Semantic Web: ESWC 2017 Satellite Events*. Springer–Verlag, 2017.

[32] Douglas Laney. 3D data management: Controlling data volume, velocity and variety, 2001.

[33] Petri Leskinen. Sotilashenkilöiden ja joukko-osastojen mallintaminen ja käyttö toimijaontologiana. Master's thesis, Aalto University, Dept. of Computer Science, Finland, 2016.

[34] Petri Leskinen, Mikko Koho, Erkki Heino, Minna Tamper, Esko Ikkala, Jouni Tuominen, Eetu Mäkelä, and Eero Hyvönen. Modeling and using an actor ontology of second world war military units and personnel. In *Proceedings of the 16th International Semantic Web Conference (ISWC 2017)*, pages 280–296. Springer–Verlag, 2017.

[35] John Lloyd. *Foundations of Logic Programming (2nd edition)*. Springer–Verlag, 1987.

[36] Eetu Mäkelä, Kaisa Hypén, and Eero Hyvönen. BookSampo—lessons learned in creating a semantic portal for fiction literature. In *Proceedings of ISWC-2011, Bonn, Germany*. Springer–Verlag, 2011.

[37] Eetu Mäkelä, Tuukka Ruotsalo, and Hyvönen. How to deal with massively heterogeneous cultural heritage data—lessons learned in CultureSampo. *Semantic Web – Interoperability, Usability, Applicability*, 3(1), 2012.

[38] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

[39] Eetu Mäkelä. Combining a rest lexical analysis web service with sparql for mashup semantic annotation from text. In *Proceedings of the ESWC 2014 demonstration track*. Springer–Verlag, 2014.

[40] Michael Piotrowski. *Natural Language Processing for Historical Texts*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool, Palo Alto, USA, 2012.

[41] S. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach*. Prentice-Hall, Upper Saddle River, New Jersey, 2010.

[42] Katri Seppälä and Eero Hyvönen. Asiasanaston muuttaminen ontologiaksi. Yleinen suomalainen ontologia esimerkkinä FinnONTO-hankkeen mallista. Technical report, National Library of Finland, Plans, Reports, Guides, 2014. https://www.doria.fi/handle/10024/96825.

[43] J. Sowa. *Knowledge Representation. Logical, Philosophical, and Computational Foundations*. Brooks/Cole, 2000.

[44] S. Staab and R. Studer, editors. *Handbook on ontologies (2nd Edition)*. Springer–Verlag, 2009.

[45] Brian Steele, John Chandler, and Swarna Reddy. *Algorithms for Data Science*. Springer–Verlag, 2016.

[46] Osma Suominen, Eero Hyvönen, Kim Viljanen, and Eija Hukka. HealthFinland – a national semantic publishing network and portal for health information. *Journal of Web Semantics*, 7(4):287–297, Dec 2009.

[47] Osma Suominen, Kim Viljanen, and Eero Hyvönen. TerveSuomi-portaalin metatietomäärittely. Technical report, Semantic Computing Research Group, SW 2.0 project, 2009.

[48] Minna Tamper. Extraction of entities and concepts from Finnish texts. Master's thesis, Aalto University, Dept. of Computer Science, Finland, 2016.

[49] Minna Tamper, Petri Leskinen, Esko Ikkala, Arttu Oksanen, Eetu Mäkelä, Erkki Heino, Jouni Tuominen, Mikko Koho, and Eero Hyvönen. AATOS: a configurable tool for automatic annotation. In *Language, Technology and Knowledge*. Springer–Verlag, 2017.

[50] Jouni Tuominen, Nina Laurenne, Mikko Koho, and Eero Hyvönen. The birds of the world ontology AVIO. In *The Semantic Web: ESWC 2013 Satellite Events*, pages 300–301. Springer–Verlag, 2013.

Hakemisto

Semanttinen web – linkitetyn avoimen datan käsikirja on ensimmäinen suomenkielinen perusteos yhdestä WWW:n keskeisimmistä kehityssuunnista, semanttisesta webistä (Semantic Web), jossa verkon sisällöt muuttuvat vähitellen tietokoneiden ymmärtämäksi linkitetyksi dataksi, Web of Data. Tämä mahdollistaa verkon sisältöjen rikastamisen uudella tavalla tietoja toisiinsa linkittämällä ja tekoälyyn perustuvien älykkäiden järjestelmien kehittämisen. Teos on tarkoitettu kaikille webin mahdollisuuksista, web-teknologioista ja niiden käytännön sovelluksista kiinnostuneille.

Teos esittelee linkitetyn datan (Linked Data) idean, sen perustana olevan semanttisen teknologian sekä avoimen datan julkaisuperiaatteet ja -käytännöt WWW:n kehitystä koordinoivan W3C-järjestön standardien ja suositusten mukaan. Teknologian soveltamista ja datan hyödyntämistä käsitellään käytännön esimerkkien ja kokemusten avulla, joita on saatu mm. laajoissa kansallisissa FinnONTO- ja Linked Data Finland -tutkimushankkeissa. Työssä tarvitaan ontologioihin perustuvan verkkomuotoisen datan sisällöntuotantoa ja yhteisen verkossa jaetun tietoinfrastruktuurin hyödyntämistä.

Eero Hyvönen on Helsingin yliopiston digitaalisten ihmistieteiden keskuksen HELDIG johtaja ja semanttisen mediatekniikan professori Aalto-yliopiston tietotekniikan laitoksella. Hänen keskeisenä tutkimusaiheenaan vuodesta 2001 lähtien on ollut semanttinen web ja siihen liittyvät kansalliset tietoinfrastruktuurit ja sovellukset, joita hän on kehittänyt yhteistyössä tutkimusryhmänsä kanssa laajoissa monialaisissa hankkeissa Suomessa ja kansainvälisesti. Eero Hyvönen on julkaissut yli 400 artikkelia ja kirjaa saanut työstään lukuisia kansainvälisiä ja kotimaisia palkintoja, kuten alan tutkijayhteisön Semantic Web Challenge Award (kahdesti), LODLAM Challenge Open Data Prize, Unesco/UNIDAn World Summit Award, ja kotimaassa mm. Apps4Finland Award, Open Finland Challenge Award, Vuoden Norssi ja Tiedonjulkistamisen valtionpalkinto.