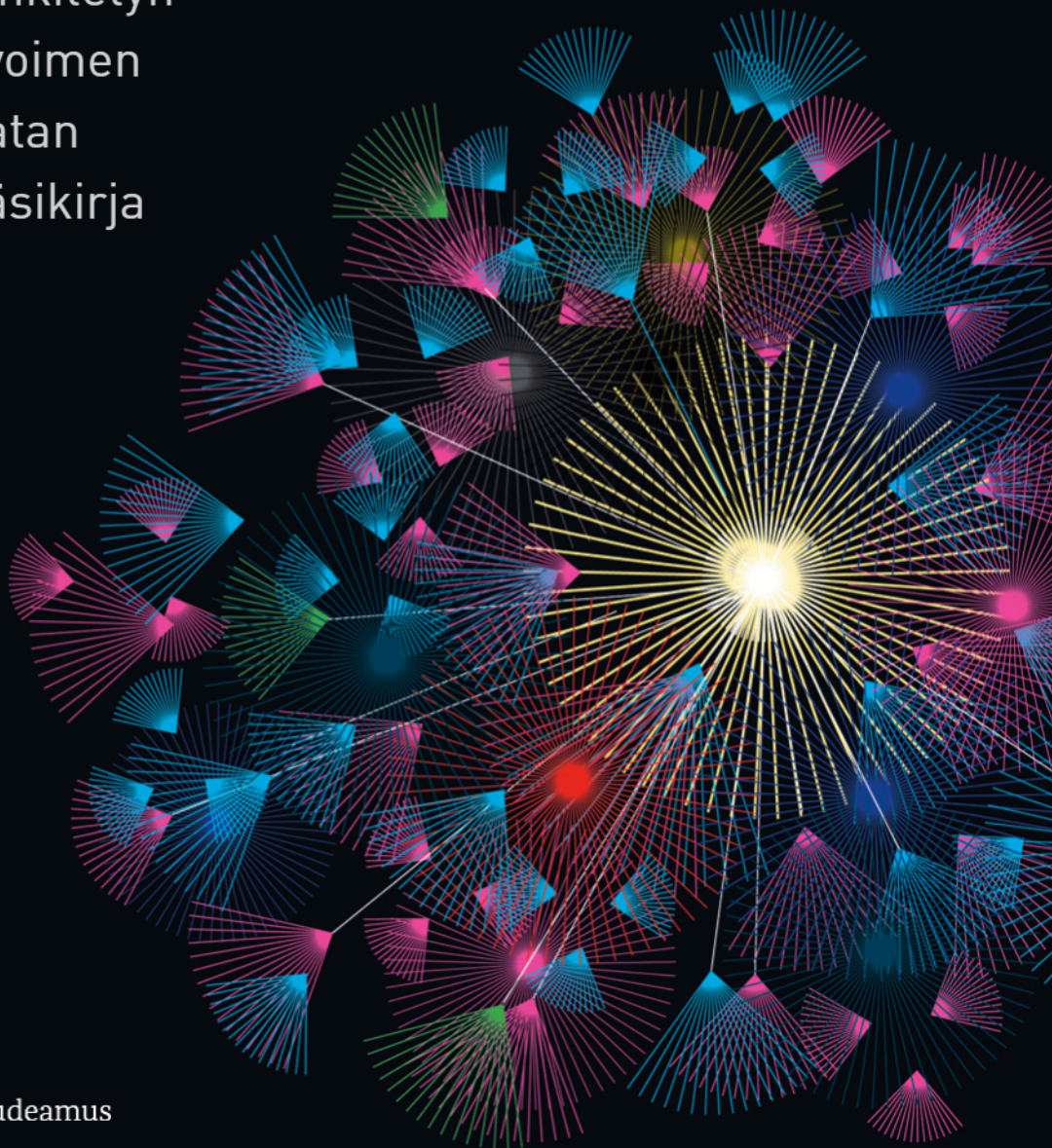


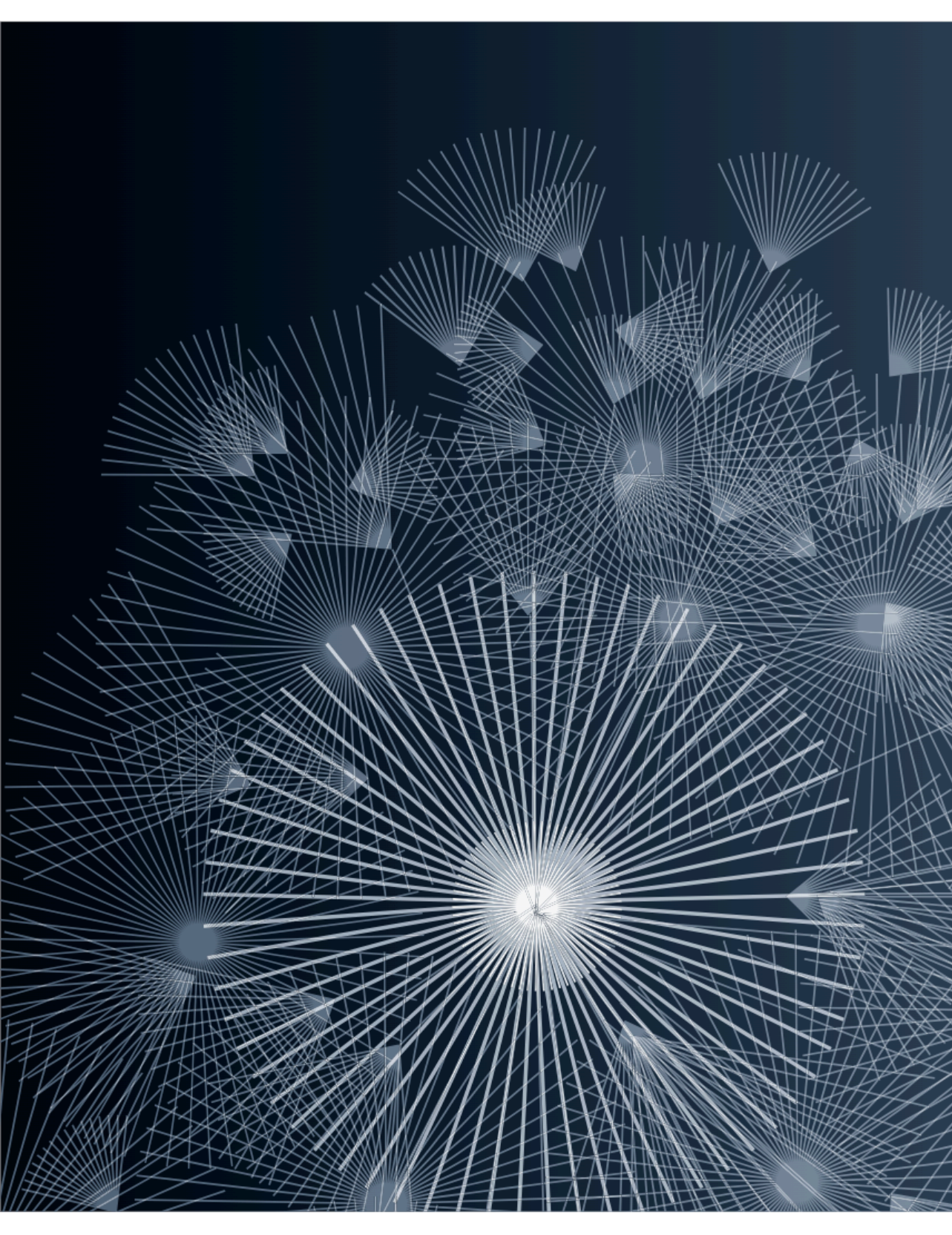
Eero Hyvönen

Semanttinen web

Linkitetyn
avoimen
datan
käsikirja



Gaudeamus



Eero Hyvönen

Semanttinen web

Linkitetyn
avoimen
datan
käsikirja





WSOY:n kirjallisuussäätiö on tukenut teoksen kirjoittamista

Copyright © 2018 Eero Hyvönen & Gaudeamus

Gaudeamus Oy

www.gaudeamus.fi

Kansi: Pekka Krankka

KL: 61.72

UDK: 004

ISBN 978-952-345-524-5 (pdf)

Esipuhe

Vaarana ei ole se, että kone kehittyy ihmisen kaltaiseksi, vaan se, että ihmisestä tulee konemainen.

Sydney J. Harris (1917–1986)

Lähtölaukaus tämän teoksen kirjoittamiselle paukahti helmikuussa 2001, kun webin infrastruktuurin kehitystyötä maailmanlaajuisesti koordinoiva W3C-järjestö¹ käynnisti laajamittaisen semanttisen webin kehitysohjelmansa *Semantic Web Activity*. Sen visiona oli luoda WWW:n perustaksi tietoinfrastruktuuri, joka yhdistäisi verkossa olevan tiedon maailmanlaajuisesti semanttiseksi, tietokoneiden ”ymmärtämäksi” verkoksi (Web of Data). Idea ruokki kaltaiseni tekoälytutkijan mielikuvitusta siinä määrin, että heti samana syksynä järjestin Helsingin yliopistolla yhteistyökumppaneiden ja W3C:n väen kanssa tilaisuuden *Semantic Web Kick-off in Finland*, johon saapui yllättäen yli 200 kuulijaa.² Aihe selvästi kiinnosti monia eri tahoja, ja sen piiriin syntyikin nopeasti kansainvälinen tutkimustraditio 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) -tutkimuskes-

¹ <https://www.w3.org/>

² Tilaisuuden esitelmät julkaisiin HIIT-tutkimuskeskuksen julkaisusarjan ensimmäisenä niteenä (Hyvönen, 2002).

³ <http://seco.cs.aalto.fi/>

kukseen.¹

Uuden tutkimus- ja kehitysalueen yhtenä kotimaisena haasteena oli opetuksen käynnistäminen uusista semanttisista web-teknologioista ja suomen kielen terminologian luominen alalle. Alussa asiaa hoidettiin satunnaisilla projektikursseilla ja tutkimusseminaareilla, mutta alan teknologioiden ja W3C:n standardien vakiintuessa kasvoi tarve alan peruskurssin kehittämiseksi. Tähän päästiin vasta keväällä 2005 Teknillisen korkeakoulun viestintätekniikan laboratoriossa, jonne oli perustettu juuri tähän aihepiiriin keskittynyt semanttisen mediatekniikan professuuri. Minut nimitettiin virkaan, ja siirryimme tutkimusryhmäni kanssa Otniemeen. Yhteistyö Helsingin yliopiston kanssa on kuitenkin jatkunut katkeamattomana.

Tarve suomenkieliselle oppikirjalle tuli vastaan heti ensimmäisellä luentokerralla: voiko ala olla uskottava, jos siitä ei ole saatavilla suomenkielistä kirjallisuutta? Kesti kuitenkin peräti kymmenen vuotta tarttua tehtävään. Ratkaiseva taitekohta oli apurahahakemuksen jättäminen WSOY:n kirjallisuussäätiölle, jonka myönteisen päätöksen jälkeen pakotie oli tukossa, sekä Aalto-yliopiston myöntämä ”sapattivapaa”, jonka tavoitteisiin kirjattiin kirjan kirjoittaminen vuosina 2015–2016. Kiitos tuesta! Kymmenen vuoden aikana yliopisto-opetus oli tosin kansainvälistynyt sen verran, että teoksen pohjana olevaa Semanttinen web -kurssia Aalto-yliopistossa alettiin luennoida englanniksi. Toisaalta digitalisaation edetessä yhä uusille aloille ohjelmointia on alettu opettaa yhä enemmän eri sovellusaloilla ja yhä nuoremmille, kuten lukioissa. Lisäksi digitaalisten ihmistieteiden ja Helsingin yliopiston uuden aihepiiriä edistävän HELDIG-keskuksen² piirissä on syntynyt uutta tarvetta suomenkieliselle oppimateriaalille.

Teos on tarkoitettu kaikille web-teknologiasta ja tietotekniikasta kiinnostuneille. Erityinen painopiste on kulttuurialan sovelluksissa, varsinkin niin sanotuissa semanttisissa portaaleissa, joissa eri organisaatioiden dataa linkitetään ja rikastetaan sisällöllisesti toistensa avulla verkossa. Oman kohderyhmänsä teokselle muodostavat tutkijat ja kehittäjät, joiden nuoruudessa webiä ei ollut tai sen teknologioita opetettu, mutta jotka jälkepäin omaehtoisesti haluavat täydentää osaamistaan web-ilmioista. Web ja internet ovat ennennäkemättömällä tavalla muuttaneet

¹ <http://www.hiit.fi/>

² <http://heldig.fi/>

maailmaamme vain parinkymmenen vuoden kuluessa, ja ne voidaan lukea jo nyt ihmiskunnan suurimpien innovaatioiden joukkoon.¹

Tämän teoksen tavoitteena on luoda konkreettinen ja käytännöllinen kuva kehityksestä kohti tekoälyyn perustuvaa, ”älykästä” semanttista webiä. Tarkoitus on, että teoksen luenut 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 laskennallisista menetelmistä ja ohjelmoinnista on erittäin hyödyllistä.

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

Kirjan kolmannen osan aiheena ovat RDF-mallin varaan kehitetyt tietämyksen esittämisen mallit ja standardit, joita käytetään eri sovellusalueiden tietosisältöjen kuvaamiseen metadatana ja niin sanottuina ontologioina. Semanttisen webin tietämyksen esitysmuodot perustuvat loogiikkaan, jonka avulla tietokone pystyy tekemään automaattisesti päätelmiä. Päätelyn 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 osuus perustuu 15 vuoden aikana saatuihin omakohtaisiin kokemuksiin kansallisen semanttisen webin tietoinfrastruktuurin ja sovellusten kehittämisestä eri aloilla.

Teosta tukemaan ja täydentämään olemme tutkimusryhmäni kans-

¹ Ks. esim.

<https://www.famousscientists.org/5-most-important-inventions-of-all-time/>.

sa kehittämässä itseopiskeluun soveltuvaa, interaktiivista verkkosivustoa Linked Data School LinDa:

<http://linda.seco.cs.aalto.fi/>

Sivuston ideana on esitellä vaiheittain prosessi, jolla tietoaaineisto voidaan muuntaa semanttiseksi verkoksi, rikastaa sitä muilla aineistoilla ja julkaista tulos verkossa linkitetyn datan palveluna. Sen varaan voidaan joustavasti kehittää erilaisia sovelluksia, kuten tämän teoksen lopussa esiteltävä Sotasampo.

Nyt valmistunut teos perustuu pitkäaikaiseen yhteistyöhön opiskelijoiden ja tutkijoiden kanssa Aalto-yliopiston ja Helsingin yliopiston Semanttisen laskennan tutkimusryhmässä SeCo, jossa toimivat tätä kirjoitettaessa Kasper Apajalahti, Erkki Heino, Esko Ikkala, Mikko Koho, Petri Leskinen, Eetu Mäkelä, Arttu Oksanen, Minna Tamper ja Jouni Tuominen. Monet teoksessa esiteltävät ajatukset ja sovellukset perustuvat yhteistyöhön SeCo-ryhmän 65 alumnin ja vierailevan tutkijan kanssa, joista mainittakoon muita väheksymättä Miika Alonen, Matias Frosterus, Tomi Kauppinen, Jussi Kurki, Thea Lindquist, Tuukka Ruotsalo, Samppa Saarela, Osma Suominen ja Kim Viljanen.

Tutkimusryhmämme työtä on vuodesta 2002 rahoittanut ja ohjannut yli viisikymmentä organisaatiota ja yritystä edustajineen, jotka ovat uskoneet kanssamme semanttisen webin sekä avoimen datan ja tieteen vision. Lämmin kiitos luottamuksesta ja yhteistyöstä kuuluu teille kaikille mukana matkanneille. Tässä teoksessa esiteltävien järjestelmien ja sovellusten osalta haluaisin erityisesti kiittää Kansalliskirjastoa (Juha Hakala) ja Sanastokeskus TSK:ta (Katri Seppälä), jotka olivat mukana FinnONTO:n ontologioiden ja ONKI/Finto-ontologiapalveluiden kehittämisessä. Museoalan MuseoSuomi- ja Kulttuurisampo-hankkeissa muun muassa Museoviraston (Sirkka Valanto ja Vesa Hongisto), Espoon kaupunginmuseon (Suvi Kettula), Suomen valokuvataiteen museon (Asko Mäkelä ja Elina Heikka) ja Valtion taidemuseon (Perttu Rastas) aineistot, asiantuntemus ja kannustus olivat tärkeitä. Kirjasampo taottiin yhdessä yleisten kirjastojen kanssa (Erkki Lounasvuori, Matti Sarmela ja Kaisa Hypén). Semanttisen Kalevalan ja Semanttisen Kansallisbiografian kehittäminen perustui yhteistyöhön Suomalaisen Kirjallisuuden Seuran (Lauri Harvilahti ja Kirsi Keravuori) kanssa. Sotasammon kehittämisessä Kansallisarkiston (Tomi Ahoranta ja Päivi Happonen) aineistot,

yhteistyö ja tuki olivat ratkaisevassa asemassa. TerveSuomi-järjestelmä suunniteltiin Terveystieteiden ja hyvinvoinnin laitoksen (Eija Hukka) hankkeisiin liittyen. Paikkatietoon liittyvää yhteistyötä tehtiin erityisesti Maanmittauslaitoksen (Reino Ruotsalainen ja Esa Tiainen) ja Geologian tutkimuskeskuksen (Jari Väättäinen) kanssa, ja Luonnontieteellinen museo oli mukana biologian ontologioiden ja havaintoaineistojen kehittämisessä (Hannu Saarenmaa ja Nina Laurenne). Oikeusministeriöstä (Aki Hieta-nen) saimme kumppanin Semanttisen Finlexin kehittämiseen, ja valtiovarainministeriö (Olli-Pekka Rissanen ja Anne Kauhanen-Simanainen), liikenne- ja viestintäministeriö (Taru Rastas), opetus- ja kulttuuriministeriö (Juha Haataja) ja CSC – Tieteen tietotekniikan keskus (Pirjo-Leena Forsström) ovat jakaneet uskomme avoimeen dataan ja tieteeseen.

Lopuksi vielä erityinen kiitos Jouni Tuomiselle arvokkaista kommentista käsikirjoitukseen liittyen, Gaudeamuksen Siiri Raiskilalle avusta kielenhuollossa sekä Teemu Ojanteelle ja Leena Kaakiselle hyvästä yhteistyöstä teoksen kustantamisessa.

Espoossa Suomen 100. itsenäisyyspäivänä 6.12.2017

Eero Hyvönen

Sisällys

1	Johdanto	15
I	KOHTI SEMANTTISTA WEBIÄ	17
2	Webin kehityssuuntia	19
2.1	Web julkaisukanavana	19
2.2	Tiedon avoimuus – Open Data	21
2.3	Tiedon yhteisöllisyys – Web 2.0	23
2.4	Tiedon määrä – Big Data	24
2.5	Tieto palveluina – Web Services	25
2.6	Tiedon verkko – Web of Data	26
3	Tiedonhaun haasteita	32
3.1	Perinteinen tekstihaku	33
3.2	Haun haasteita	34
3.3	Selailun haasteita	38
3.4	Ratkaisuna tiedon semantiikka	39
II	LINKITETTY DATA	49
4	Linkitetyn datan esittäminen	51
4.1	Tiedon esitys semanttisena verkkona	52
4.1.1	Verkko kaaviona	52
4.1.2	Verkon esittäminen kolmikkoina	52

4.1.3	Verkon looginen tulkinta	53
4.1.4	Verkko tekstinä	54
4.2	Semanttisen webin tekninen perusta	55
4.2.1	HTML-kieli	56
4.2.2	HTTP-protokolla	56
4.2.3	Yhtenäiset osoitteet ja tunnisteet: URI	57
4.2.4	Resurssien ja literaalien esittäminen	62
4.2.5	Datan linkitys tietojoukkojen välillä	64
5	RDF-verkon esittämiskielet	67
5.1	Data kolmikkoina: N-Triples	67
5.2	Yksinkertaisempaa koodia: Turtle	69
5.2.1	Nimiavaruudet	69
5.2.2	Monta samaa ominaisuutta	72
5.2.3	Eri ominaisuudet samalla kertaa	72
5.2.4	Ominaisuuden arvo omilla ominaisuuksilla	72
5.2.5	Tietotyypit	74
5.3	Verkon esittäminen RDF/XML-kielillä	76
5.4	Esimerkki tietojen yhdistämisestä	76
5.5	Muita RDF:n ominaisuuksia	81
5.5.1	Säiliöt	81
5.5.2	Kokoelmat	82
5.5.3	Reifikaatio	83
5.6	JSON-LD	84
6	Tiedon haku ja ylläpito: SPARQL	89
6.1	SPARQL-datapalvelun perustaminen	90
6.2	Peruskyselyt	92
6.2.1	Tiedon haku: SELECT	93
6.2.2	Tiedon testaaminen: ASK	98
6.2.3	Resurssin kuvaus: DESCRIBE	99
6.2.4	Uuden RDF-verkon luominen: CONSTRUCT	99
6.3	Laajennuksia peruskyselyihin	100
6.3.1	Ominaisuuspolut	100
6.3.2	Arvosijoitus	101

6.3.3	Aggregaattikyselyt	102
6.3.4	Alikyselyt	106
6.3.5	Federoitu kysely	106
6.4	Graafien hallinta ja päivitys	107
6.5	Graafin datan päivittäminen	108
7	Linkitetyn datan julkaiseminen	110
7.1	Linkitetyn datan neljä periaatetta	110
7.2	HTTP-kutsujen dereferointi	111
7.3	URI-tunnisteiden sisältöneuvottelu	113
7.4	Datajulkaisujen tähtiluokitus	115
7.5	Datan julkaiseminen WWW-sivuilla	116
III	TIETÄMYKSEN ESITTÄMINEN	121
8	Metadata ja ontologiat	123
8.1	Metadatan käsite ja muodot	124
8.2	Dublin Core -malli	126
8.3	CIDOC CRM: datan harmonisointi	129
8.4	Ontologian käsite	133
9	RDF Schema	135
9.1	Luokat ja yksilöt	135
9.2	Luokkien hierarkia	136
9.3	Ominaisuuksien hierarkia	136
9.4	Alue- ja arvoraioitteet	137
10	SKOS – Simple Knowledge Organization System	139
10.1	Käsittemalli	140
10.2	Käsitteen nimikkeet	140
10.3	Notaatiot	143
10.4	Semanttiset relaatiot	144
10.5	Ryhmittelevät termit	146
10.6	SKOS-sanastojen siltaaminen	147
10.7	Dokumentointiominaisuudet	148

10.8	Päätelysäännöt ja integriteettiehtoja	148
11	Web Ontology Language OWL	150
11.1	Johdatus logiikkaan	151
11.1.1	Logiikan idea	151
11.1.2	Lauselogiikka	152
11.1.3	Predikaattilogiikka	154
11.2	OWL-kielen syntaksit	159
11.3	Ontologiadokumentti	161
11.4	Luokkien määrittely	162
11.4.1	Samuus ja eriyys	163
11.4.2	Luokkien määrittely joukko-opillisesti	164
11.4.3	Luokkien määrittely joukko-opilla	167
11.5	Ominaisuuksien määrittely	167
11.5.1	Objektiominaisuudet	168
11.5.2	Tietotyyppiominaisuudet	170
11.5.3	Annotointiominaisuudet	171
11.6	OWL-profililit	171
11.6.1	OWL 2 EL	172
11.6.2	OWL 2 QL	172
11.6.3	OWL 2 RL	173
12	Päätelysäännöt	174
12.1	Semanttisen webin looginen tulkinta	174
12.2	Sääntöjen käyttö	177
12.2.1	Hornin logiikan suhde kuvailulogiikoihin	180
12.2.2	Suljetun maailman oletus	183
12.2.3	Yksikäsitteisten nimien oletus	184
12.2.4	Yhteenveto	185
12.3	Käyttötapauksia säännöille	185
IV	SOVELLUKSET JA INFRASTRUKTUURI	187
13	Sovellusten kehittäminen	189
13.1	Yleiset ja alakohtaiset sovellukset	190

13.2 Semanttisen portaalin osat	192
14 Portaalimalli yhteisölliseen julkaisemiseen	195
14.1 Kulttuuriaineistot verkossa	195
14.2 Hajautettu haku ja tiedon aggregointi	197
14.3 Edut käyttäjille	203
14.4 Edut julkaisijoille	203
14.5 Uusia haasteita	204
15 Sisällöntuotanto	205
15.1 Tekstin tunnistaminen (OCR)	206
15.2 Datamuunnokset ja linkitys	207
15.3 Merkitysten erottelu	209
15.4 Datan semanttinen validointi	210
16 Webin tietoinfrastruktuurit	212
16.1 Tietoinfrastruktuurin osat	213
16.2 Ontologiapalvelut	216
16.3 Työn arviointia	222
16.4 Linkitetyn datan palvelut	223
17 Sovellusesimerkki: Sotasampo	226
17.1 Tavoitteet ja käyttötapaukset	227
17.2 Aineistot ja tiedon tuottajayhteisö	228
17.3 Metadata ja ontologiat	229
17.4 Entiteettien automaattinen linkitys	232
17.5 Portaalisovellus Sotasampo.fi	236
17.6 Datapalvelu Sotasampo	242
17.7 Uutuusarvo ja jatkokehitys	242
Liite: Esimerkki OWL-ontologiasta	245
Kirjallisuutta	251
Kirjoittaja	271

1

Johdanto

Aloita, ja puolet työstä on tehty. Aloita uudelleen, ja työ on valmis.

Decimus Magnus Ausonius
(n. 310–395)

World Wide Webin (WWW) idean koko maailman kattavasta hypertextijärjestelmästä esitti Tim Berners-Lee vuonna 1989. Ajatus syntyi tarpeesta tukea CERN-tutkimuskeskuksen fyysikkojen työtä yhteisöllisen, hajautetun julkaisukanavan kautta internetissä. Internetiä oli kehitetty jo aiemmin 1960-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, televisioasemien tai kirjankustantajien. Toki vieläkin tarvitaan palveluntarjoaja kotisivuille ja blogeille tai erityinen yhteisöllinen tiedonjakamisen alusta, kuten Facebook tai Twitter, mutta näitä webissä riittää. Julkaisemisen hinta enää ole (yleensä) merkittävä este ainakaan pienimuotoisemmille julkaisuille.

URL-osoite

WWW:n uusi tekninen oivallus oli maailmanlaajuinen, *URL-osoitteisiin* (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. Osoitteina käytettävät tunnisteet ovat keskeisessä asemassa myös semanttisessa 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 ihmiskäyttäjää varten.

syvä web

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 rikkoutui 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.

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

¹ Webin historiaa on muisteltu ja käsitelty mm. teoksessa (Berners-Lee ym., 2000).

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

I

KOHTI SEMANTTISTA WEBIÄ

2

Webin kehityssuuntia

Maailma on luotu uudelleen
luotavaksi.

Georges Duhamel (1884–1966)

Tässä luvussa luodaan katsaus webin kehitykseen tiedon julkaisemisen ja käytön monista näkökulmista. Tämän jälkeen näkökulma tarkennetaan teoksen aiheena olevaan semanttiseen webiin.

2.1 Web julkaisukanavana

Web tarjoaa kanavan tiedonhankintaa ja sosiaalista kanssakäymistä varten. Fyysistä kokemusta näyttöpääte tai mobiililaite ei korvaa, mutta se mahdollistaa uusia tapoja tiedon saamiseen ja yhteydenpitoon. Tässä teoksessa keskitytään erityisesti webin tarjoamiin mahdollisuuksiin tietojen yhdistämisen, julkaisemisen ja haun kannalta. Näitä ovat muiden muassa:

1. *Ajasta ja paikasta riippumaton tiedonvälitys.* Tietosisällöt saadaan joustavasti tutkijoiden ja suuren yleisön käyttöön. Tietoon tutustuminen ei edellytä fyysistä läsnäoloa, esimerkiksi kirjastoon tai museoon tulemistä tiettyyn aukioloaikaan.
2. *Tietosisällön määrän lisääminen.* Laajat kokoelmat, joiden fyysinen näyttäminen ja tiedon saataville asettaminen olisi mahdotonta, voidaan avata yleisölle WWW:n välityksellä.

vuorovaikutteisuus
sopeutuvuus

3. *Tietosisällön laadun parantaminen.* Tietotekniikka mahdollistaa tietojen haun, yhdistämisen ja esittämisen tavoilla, jotka eivät ole teknisesti tai taloudellisesti mahdollisia fyysisissä tiloissa.
4. *Vuorovaikutteisuuden hyödyntäminen.* Tietojärjestelmien *vuorovaikutteisuus* (interactivity) ja *sopeutuvuus* (adaptivity) 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. Internet-sivustoja on myös määrätietoisesti ryhdytty tallentamaan uudenaikaisina historiallisina dokumentteina.¹

Web-sovellusten kehittämisessä on runsaasti käytännöllisiä, teknisiä haasteita. Suurimmat hankaluudet liittyvät kuitenkin tietosisältöjen esittämiseen eivätkä tekniikkaan. Avainkysymys on, miten web-sivustoista saadaan kävijöiden kannalta aidosti mielenkiintoisia, hyödyllisiä ja helpokäyttöisiä. 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 muotiterminsä sekä poliittisia ja teknologisia kehityspolkuja:

Open Data

1. *Tiedon avoimuus (Open Data).* Tieto on tietoyhteiskunnan toiminnan tärkeimpiä tukipilareita. Mutta missä määrin sen pitäisi olla avointa (open data) ja suljettua? Erilaisia näkökohtia ja argumentteja voidaan esittää muun muassa 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?

Big Data

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

¹ Ks. esim. Internet Archive <https://archive.org/> ja Kansalliskirjaston Suomalaisen verkkoarkisto <http://verkkoarkisto.kansalliskirjasto.fi/va/>.

4. *Tieto palveluina (Web Services)*. Millaisina toiminnallisina palveluina tieto kannattaa julkaista toisaalta ihmiskäyttäjille, toisaalta toisten verkkopalveluiden käytettäväksi? 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 tietoa ja kehittää älykkäitä sovelluksia (tekoäly). Web of Data

Seuraavaksi tarkastellaan lyhyesti näitä webin tietoon liittyviä kehityssuuntia.

2.2 Tiedon avoimuus – Open Data

Tietoyhteiskunnan tärkein 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 voisimme olla tiedon miljonäärejä. Sitä ennen on kuitenkin ratkaistava joitain perustavaa laatua olevia kysymyksiä. Kuka kustantaisi tietotuotteen ensimmäisen kopion valmistamisen, jos kaikki muut saavat sen ilmaiseksi käyttöönsä? Hyödyn korjaa tällöin jokin muu taho kuin tiedon tuottaja. Yritysten kannalta avoin ilmainen data 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 koskee tiedon arvoa. Eikö tiedon arvo vähene sitä levitettäessä, 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, YouTube'n 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 palvelut ovat osoittaneet.

Julkisen tiedon avoimeen, maksuttomaan avaamiseen sitä tarvitseville tahoille on useita perusteita.

- *Demokratian edistäminen.* Kansalaisilla on oltava demokraattinen oikeus saada ja käyttää julkisin varoin tuotettua tietoa maksutta, koska he ovat sen viime kädessä maksaneetkin esimerkiksi veroina. Tämä on jo pidempään ollut yleisperiaatteena muun muassa Yhdysvalloissa.
- *Tiedon saatavuuden parantaminen.* Saatavuutta rajoittava, yhden tahon eduntavoittelu saattaa estää tiedon hyödyntämisen laajemmassa yhteydessä. Esimerkiksi kartta- ja paikkatiedon osalta helposti saatavilla olevat aineistot, kuten Google Maps ja alun perin sotilaskäyttöön kehitetty, satelliittien kautta saatava GPS-paikkatieto, ovat käynnistäneet todellisen sovellusten ja innovaatioiden aallon. Suomessa julkisia organisaatioita on kuitenkin suoraan edellytetty muun muassa maksuperustella 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 yksittäiset Wikipedian artikkelit erillisinä eivät ole kovin arvokkaita, vaan arvo syntyy niiden muodostamasta kokonaisuudesta.
- *Yhteentoimivuuden edistäminen.* Yhteiset avoimet pelisäännöt tiedon julkaisemisessa, yhdistämisessä ja jakamisessa edistävät tieto-

järjestelmien ja palveluiden *yhteentoimivuutta* (interoperability), jonka puute on keskeinen este muun muassa julkisen sektorin eri toimijoiden yhteistyössä.

yhteentoimivuus

- *Tuottavuuden tehostaminen.* Tietoja ja ratkaisuja jakamalla voidaan tehostaa työnjakoa, eliminoida tarpeetonta päällekkäistä työtä ja lisätä näin työn tuottavuutta. Esimerkiksi Suomen kuntien verkkopalveluilla on paljon samoja toiminnallisuuksia, mikä mahdollistaa yhteistyön 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 tieto vain 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 International¹ -yhteisö, jolla on aktiivinen alajärjestö Suomessakin, on toimittanut verkossa olevan monikielisen Open Data Handbook -sivuston.² Sen 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, toisin sanoen web-sivustojen luominen ja omien aineistojen julkaiseminen, ei kuitenkaan ollut kovin yksinkertaista, vaan vaati teknistä osaamista. Web 2.0 -termin lanseerasivat vuonna 2004 erityisesti Tim O'Reilly ja Dale Dougherty. Sillä viitattiin uudentyyppisiin, helppokäyttöisiin ja yhteentoimiviin web-järjestelmiin, jotka perustuvat käyttäjien itse tuottamaan si-

¹ <https://okfn.org/>

² <http://opendatahandbook.org/>

sältöön.¹ Monet eniten käytetyistä nykyistä web-palveluista ovat tämän tyyppisiä sovelluksia, esimerkiksi YouTube ja Facebook.

2.4 Tiedon määrä – Big Data

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

skaalautuvuus

Dough Laney'n klassisen, vuonna 2001 esittämän niin sanotun kolmen V:n mallin mukaan² tiedon määrällisen kasvun haasteita ovat:

1. *Volume (määrä)*. Perinteinen tietokantatekniikka ei riitä hyvin suurien datajoukkojen tallentamiseen, siirtoon ja käsittelemiseen. Ratkaisumallina on datan jakaminen osiin.
2. *Velocity (nopeus)*. Nopeus viittaa tässä yhteydessä laskennan nopeuteen. Riittävän tehokkuuden aikaansaamiseksi laskenta tyypillisesti hajautetaan kymmenille tai jopa tuhansille rinnakkain samanaikaisesti toimiville laitteistoille, joiden tulokset kootaan lopuksi yhteen.
3. *Variety (monimuotoisuus)*. Tieto tai sen osat voidaan esittää eri tavoin, jolloin niiden käsitteleminen kokonaisuutena hankaloituu. Käytössä voi olla esimerkiksi vaihtoehtoisia tai epäyhtenäisiä tapoja nimien, osoitteiden, numeeristen arvojen ynnä muiden esittämiseksi tai näiden välisten yhteyksien kuvaamiseksi.

Mallia on sittemmin täydennetty uusilla V-ulottuvuuksilla, kuten:

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

Suurdatan käsittelyssä tarvitaan tehokkaita tietojenkäsittelyn menetel-

¹ Lisätietoa Web 2.0 -ilmiöstä ja tiedon yhteisöllisyydestä löytyy esimerkiksi teoksesta (Anderson, 2012).

² Ks. (Laney, 2001).

miä ja laitteistoja, joissa tyypillisesti hyödynnetään rinnakkaislaskentaa.¹ Toisaalta suurdatan tulkinnessa ja hyödyntämisessä tarvitaan *data-analyysin* ja *datatieteen* menetelmiä² sekä visualisointeja.³

data-analyysi
datatiede

2.5 Tieto palveluina – Web Services

Passiivisen tiedon julkaisemisen ohella toinen webin kehittymisen päähaaroista on ollut toiminnallisuuksien julkaiseminen *verkkopalveluina* (Web Service).⁴ 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 monipuolinen verkkopalvelu, jonka avulla toiset verkkosovellukset saavat vaivattomasti käyttöönsä paikkatietoa, karttoja ja satelliittikuvia.

verkkopalvelu

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 ainakin toistaiseksi

¹ Uusia menetelmiä ja työkaluja suurdatan käsittelyyn ovat mm. Hadoop MapReduce (Steele ym., 2016) ja Apache Spark (Karau ym., 2015).

² Datatieteen menetelmiä on esitelty esim. teoksessa (Steele ym., 2016).

³ Suosituksi työkaluksi tässä on noussut mm. tilastollisia menetelmiä ja esitysgrafiikoita hyödyntävä R (Field ym., 2015) ja monet Python-ohjelmointikielen kirjastot.

⁴ Verkkopalvelutekniikoita on esitelty esim. teoksessa (Alonso ym., 2003).

vähäiseksi.

WWW:n keskeisin protokolla on HTTP, jonka avulla esimerkiksi selaimet lukevat eri palvelimilla olevia verkkosivuja. Yhä tärkeämmäksi verkkopalveluarkkitehtuuriksi on muodostunut HTTP-protokollaa käyttävä REST (Representational State Transfer). Sen ideana on tarjota yksinkertainen tapa toteuttaa verkkopalveluita, joissa *asiakasjärjestelmä* (client) antaa HTTP-protokollan mukaisen palvelupyynnön URL-osoitteena *palvelimelle* (server) ja saa siihen vastauksen. Kommunikointi tapahtuu usein JSON-muotoisen datan (JavaScript Object Notation) avulla. Tätä kompaktia tiedon esitysmuotoa on yksinkertaista käyttää verkkoselainten JavaScript-ohjelmissa.

Yhä useammat sovellukset perustuvat nykyään monipuoliseen selaimen ohjelmointiin (Rich Internet Application, RIA), jossa palvelimia käytetään lähinnä datan varastoina ja sovelluksen toiminnallisuus on toteutettu selaimessa. Perinteisten palvelinperustaisten sovellusten yksi keskeinen haaste interaktiivisten käyttöliittymien teossa on, ettei käyttöliittymäsivua voida päivittää nopeasti osittain vaan ainoastaan ladata malla aina koko sivu uudelleen palvelimelta, mikä hidastaa merkittävästi interaktiivisuutta. Selainta ohjelmoimalla käyttöliittymää voidaan manipuloida suoraan ja saada aikaan verkossa vastaavanlainen nopea toiminnallisuus 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-teknologiaa käyttämällä syntyy yhteen interaktiiviseen verkkosivuun perustuvia sovelluksia, joista käytetään nimitystä *yhden sivun sovellus* (Single Page Application, SPA).

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 kysy-

mys 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. Ajatuksena on esittää verkon datat ja niiden kuvailuissa (metadata) käytetyt sanastot yhteismittalaisesti niin sanottuna *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.¹ 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. Semanttisesta webistä käytetään toisinaan nimitystä Web 3.0, mikä korostaa verkon luonteessa ja teknologisessa perustassa tapahtunutta syvällistä muutosta alkuperäisen webin (Web 1.0) ja yhteisöllisen webin (Web 2.0) jälkeen. Semanttisen webin tunnetuimpana edustajana on toiminut W3C-järjestön vetäjä ja ”webin isä” Sir Tim Berners-Lee.

semanttinen verkko

semanttinen web

Web 3.0

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

RDF

SPARQL

SKOS

OWL

RIF

¹ Semanttisen webin englanninkielisiä oppikirjoja ovat esim. (Antoniou ym., 2012) ja tiedon kuvaamiseen tarkemmin paneutuva (Allemang & Hendler, 2008). W3C:n sivulta <https://www.w3.org/2001/sw/wiki/Books> löytyy kokoelma eri vuosina ilmestyneitä alan oppikirjoja.

Maailmanlaajuisen yhteisöllisen työn seurauksena verkossa on nyt vapaasti saatavilla valtava 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.

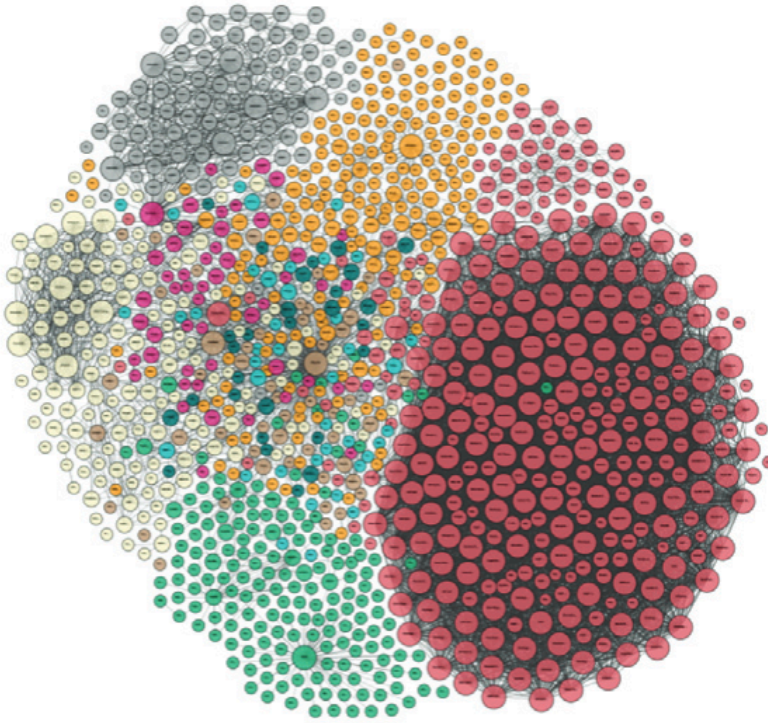
GGG (GGG). Googlen lanseeraama nimitys dataverkosta on Knowledge Graph. GGG:n ytimessä on joukko keskeisiä, toisiinsa yhdistettyjä avoimia *datajoukkoja* (dataset), jotka muodostavat eräänlaisen ”virallisen” linkitetyn avoimen datapilven, *LOD-pilven* (Linked Open Data Cloud, LOD Cloud). Kuvassa 2.1 on esitetty LOD-pilven rakenne keväällä 2017.¹ Pilvi koostuu 1139 solmusta, jotka ovat datajoukkoja jaoteltui-
 na sisällöllisesti muutamaaan eri kategoriaan. Datajoukkoja yhdistävät
 kaaret kertovat, mitkä datajoukot on sillattu toisiinsa. *Siltaus* (map-
 ping) tarkoittaa kuvausta, joka kertoo, mitkä käsitteet yhdistetyissä da-
 tajoukoissa vastaavat toisiaan. Maailmassa on esimerkiksi useita London-
 nimisiä paikkoja, joita löytyy LOD-pilven kuuluvasta kansainvälisestä
 Geonames-paikkatietorekisteristä, mutta näistä vain yksi vastaa Wikipe-
 diassa (DBpediassa) olevaa Iso-Britannian pääkaupungin käsitettä. Tä-
 mä 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 linkitetuin aineisto. LOD-pilven datajoukot ja siltaukset muodostavat jättiläismäisen, miljardeista soluista ja kaarista koostuvan semanttisen RDF-verkon, joka on maailmanlaajuisen semanttisen webin ydin. Haastajaksi DBpedialle on muodostumassa Wikipedia-järjestön itsensä koordinoima Wikidata-ohjelma, jossa luodaan semanttisen webin tekniikoilla kaikkien erikielisten Wikipedioiden yhteisestä datasta koneille vastaava linkitetyn datan palvelu (GGG) kuin mitä Wikipediat ovat ihmisikäyttäjille (WWW).² Wikidataa kehi-

¹ Tietoa pilvestä ylläpidetään osoitteessa <http://lod-cloud.net/>, jossa kerrotaan myös pilven ylläpidosta kansainvälisen yhteisön voimin. LOD-diagrammin ovat laatineet Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch ja Richard Cyganiak.

² Ohjelman kotisivu on <https://www.wikidata.org/>, ja kerätty data on avoimesti käytettävissä rajapinnan kautta osoitteessa <https://query.wikidata.org/>.

tetään tätä kirjoitettaessa yhteisöllisesti yli 17 000 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.



Kuva 2.1. Linked Open Data -pilven ydin koostui vuoden 2017 keväällä 1139 datajoukosta, jotka näkyvät kuvassa erivärisinä tai -sävyisinä palloina sen mukaan, minkä aihealueen datasta on kysymys. Datajoukkojen väliset kaaret kuvaavat siltauksia. Pilven keskellä on erikielisiä Wikipedioista louhittu semanttinen verkko, DBpedia. (<http://lod-cloud.net/>)

Avointen tietoaaineistojen ja niiden yhdistämisen perustalle on alkanut nopeasti syntyä uusia innovatiivisia toimintamalleja ja käytännön sovelluksia, joita ollaan ottamassa käyttöön tietoyhteiskunnissa ja julkishallinnoissa eri puolilla maailmaa. Monissa maissa, kuten Iso-Britanniassa, on avattu julkisen sektorin avoimen datan portaaleja, joiden kautta löytyy sekä avoimia tietovarantoja että näihin perustuvia sovelluksia, vaik-

kapa lähellä olevien koulutusmahdollisuuksien tai hoitokotien löytämiseen tai liikennetietojen seuraamiseen.¹ Eräs tunnettu sovellus on BBC:n kotisivujen verkkopalvelu, jossa organisaation eri tahojen mediasisältöjen yhdistämiseen käytetään Wikipedian semanttisen webin muunnosta DBpediaa.²

FinnONTO Suomessa semanttista webiä on kehitetty laajimmin kansallisessa semanttisen webin FinnONTO-hankesarjassa³ (2003–2012), jossa kehitettiin kansallisen semanttisen webin sisältöinfrastruktuurin prototyyppi ONKI.fi (2008) ontologiapalveluineen⁴ ja useita hajautettuun sisällöntuotantoon perustuvia semanttisia portaaleja, kuten Museo-Suomi⁵ (2004), TerveSuomi⁶ (2008), Kulttuurisampo⁷ (2008) ja Kirjasampo⁸ (2011). Aalto-yliopiston ja Helsingin yliopiston kehittämä ONKI.fi-prototyyppi otettiin koekäyttöön muun muassa monissa maamme museoissa yhdistämällä palvelu osaksi kokoelmanhallinnan järjestelmiä. Valtiovarain- ja opetus- ja kulttuuriministeriön rahoituksella ONKI:n keskeisiä osia tuotteistettiin ja otettiin käyttöön vuonna 2014 Kansalliskirjaston ylläpitämänä Finto.fi-palveluna⁹ (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 -datajulkaisualustan¹⁰ kehittämisellä ja siihen perustuviin sovelluksiin liittyen. Näitä ovat muiden muassa talvi- ja jatkosodan aineistoja julkaiseva Sotasampo.fi¹¹ (2015) ja oikeusministeriön Finlex-lakiaineistoihin perustuva Semanttinen Finlex¹² (2016), joka julkaisee keskeisimmät osat lainsäädännöstämme ja oikeustapauksia avoi-

¹ Osoitteita näihin löytyy sivustolta <https://data.gov.uk/>.

² <http://www.bbc.co.uk/academy/technology/software-engineering/semantic-web>

³ <http://seco.cs.aalto.fi/projects/finnonto/>

⁴ <http://onki.fi>

⁵ <http://www.museosuomi.fi>

⁶ <https://seco.cs.aalto.fi/applications/tervesuomi/>

⁷ <http://www.kulttuurisampo.fi>

⁸ <http://www.kirjasampo.fi>

⁹ <http://finto.fi>

¹⁰ <http://ldf.fi>

¹¹ <http://sotasampo.fi>

¹² <http://data.finlex.fi>

mena linkitettyinä 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 tietoa tuhansista sodanajan tapahtumista ja kymmenistä tuhansista luovutetun alueen paikoista historiallisilla kartoilla. Järjestelmään on yhdistetty Kansallisarkiston tuottamat tiedot kaikista viime sodissamme menehtyneistä noin 95 000 sotilaasta ja tuotu eri lähteistä tietoja tuhansista muista sodasta selvinneistä tunnetuista sotilaista. Puolustusvoimien SA-Kuva-arkistosta on saatu noin 160 000 autenttisen sota-ajan valokuvan kokoelma. Aineistoja on sillattu toisiinsa ja ulkoisiin aineistoihin, kuten Suomen Sotahistoriallisen Seuran verkossa julkaisemien *Kansa Taisteli* -lehtien (1957–1986) tuhansiin muisteluartikkeleihin.

Eräs kiinnostava kehityssuunta jatkossa on kieliteknologian yhdistäminen semanttisen webin menetelmiin ajatuksena kutoa webin semanttista verkkoa automaattisesti tekstiaineistosta. Yhtenä esimerkkinä tästä on elämäkertoihin liittyvä sovellus Vanhat norssit semanttisessa webissä (2017)¹ ja Semanttinen Kansallisbiografia, jossa Suomalaisen Kirjallisuuden Seuran toimittamat yli 13 000 elämäkertaa rakenteistetaan ja rikastetaan semanttisesti verkon data-aineistojen avulla.²

¹ <http://www.norssit.fi/semweb/>

² Ks. artikkeli (Hyvönen ym., 2018).

3

Tiedonhaun haasteita

Todellista tietämistä on tietää, että tietää, mitä tietää, ja tietää, ettei tiedä sitä, mitä ei tiedä.

Kungfutse (551–479 eaa.)

Ihmiskäyttäjän näkökulmasta webillä on kaksi pääasiallista käyttötapaa tiedon etsimisessä:

- | | |
|------------|---|
| haku | 1. <i>Haussa</i> (search) käyttäjä muotoilee tietotarpeensa <i>kyselyksi</i> (query) interaktiivisen hakujärjestelmän haluamaan muotoon ja saa kyselyn vastauksena joukon tietosisältöjä, kuten linkkejä web-sivuille. |
| kysely | |
| selailu | 2. <i>Selailussa</i> (browsing) webin sisältöihin tutustutaan assosiatiivisesti sivuja yhdistäviä linkkejä seuraten. |
| tiedonhaku | <i>Tiedonhaku</i> 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. ¹ 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 |

¹ Tiedonhaun tutkimusalaa ja menetelmiä on esitelty tarkemmin esim. teoksessa (Manning ym., 2008), josta on saatavilla oppimateriaalia myös vapaasti verkossa: <https://nlp.stanford.edu/IR-book/>.

sisältöjä, joita se käsittelee.

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* (relevance) eli parhaat osumat tulevat ensin.

olennaisuus

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

tekstihaku

käänteinen hakemisto

Tarkastellaan esimerkkinä kolme tekstidokumenttia:

Dokumentti	Teksti
D1	Kissalla on kala.
D2	Kissa syö katolla kalaa.
D3	<i>Kissa kuumalla katolla on elokuva.</i>

Taulukko 3.1. Haun kohteena oleva dokumenttijoukko.

Välimerkit poistamalla ja sanat perusmuotoistamalla näistä syntyy taulukon 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 käänteisindeksi.

Useamman hakutermien kyselyn tulos voidaan laskea termejä vastaavien

dokumenttijoukkojen leikkauksen avulla. Haettaessa esimerkiksi dokumentteja, joiden aiheena on sekä *kissa* että *katto*, saadaan tulokseksi dokumentit D2 ja D3:

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

Tuloksen dokumenttien järjestämiseen jonkin paremmuuskriteerin mukaan on kehitetty lukuisia menetelmiä. Esimerkiksi paljon käytetyllä TF-IDF-menetelmällä voidaan arvioida, miten tärkeitä eri sanat ovat kullekin dokumentille niiden joukossa. Ajatuksena on, että tärkeämmät sanat esiintyvät useaan kertaan dokumentissa mutta ovat harvinaisia koko dokumenttijoukossa. Tätä voidaan hyödyntää tiedonhaussa tulosten järjestämiseen kyselyn hakusanoja vastaan. Googlen lanseeraama PageRank-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.

tarkkuus
saanti

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

$$Tarkkuus = |(Tulos \cap Oikein)|/|Tulos|$$

Saanti taas mittaa, kuinka suuri osuus kaikista oikeista vastauksesta jäi haaviin:

$$Saanti = |(Tulos \cap Oikein)|/|Oikein|$$

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 sadan prosentin saanti olisi helppoa toteuttaa hakemalla tulokseen aina 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. Seuraavaksi luetaan joitain perustavaa laatua

olevia haasteita, joita hakujärjestelmissä joko ratkaistaan 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.
- *Homonyymit*. *Homonymialla* (homonymy) tarkoitetaan termin monimerkityksisyyttä: samalla ilmauksella voi olla samanaikaisesti useita eri merkityksiä. Esimerkiksi ”varkaudella” voidaan viitata rikokseen tai kaupunkiin, ”nokiolla” taas yritykseen, kaupunkiin, nokinäätään eli soopeliin tai henkilöön F. E. Sillanpään romaanissa *Ihmiset suviyössä* (1934). Nimet, esimerkiksi paikkojen ja henkilöiden nimet, ovat usein homonyymisia. Maanmittauslaitoksen Paikannimirekisterin mukaan Suomessa on esimerkiksi 49 eri Pyhäjärveä, ja Pyhäjärvi on myös sukunimi. Homonyymista termiä, jonka merkitykset liittyvät toisiinsa, kutsutaan *polysemiseksi* (polysemy). Esimerkiksi ”päällä” voidaan viitata nuolen tai ihmisen päähän. Homonymia huonontaa haun tarkkuutta.
- *Monikielisyys*. Erikielisissä dokumenteissa samoihin sisältöihin viitataan erikielisin termein. Lisäksi nimet kirjoitetaan eri kielissä käyttäen hyväksi erilaisia translitterointisääntöjä ja merkistöjä. Esimerkiksi venäläisen säveltäjän 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 ja niin edelleen. Lisäksi tulevat vielä päätteet (esimerkiksi hypyttääköhän). Kelvollinen tekstihaku edellyttää sekä dokumenteissa että kyselyissä esiintyvien sanojen palauttamista perusmuotoon eli *lemmausta* (lemmatization) tai vähintään taipuneiden päätteiden poistamista

homonymia

polysemia

lemmaus

stemmaus	eli <i>stemmausta</i> (stemming).
yhteyshaku	• <i>Käsitteiden väliset suhteet.</i> Hakusanojen välillä on semanttisia suhteita, joita perinteinen tekstihaku ei huomioi. Töölössä oleva pizzeria ei löydy haettaessa helsinkiläisiä ravintoloita, vaikka tiedetään Töölön olevan Helsingin osa ja pizzerian eräänlainen ravintola. Sanalla ”savu” ei löydy tulesta kertovia sivuja (ellei sivulla esiinny myös sana ”tuli”), vaikka tiedetään, ettei ole savua ilman tulta. • <i>Erilaiset hakukohteet.</i> 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 <i>yhteyshausta</i> (relational search). Tällöin halutaan esimerkiksi tietää, miten Jean Sibelius liittyy Hämeenlinnaan¹ tai Akseli Gallen-Kallela Carl Gustaf Emil Mannerheimiin.² • <i>Ongelmanratkenta.</i> Viime kädessä tiedon haussa on yleensä kysymys ongelmanratkonnasta, johon dokumenttien haku on vain yksi epäsuora väline. Halutaan esimerkiksi tietää, paljonko kilo höyheniä painaa kuussa, paljonko Togossa on asukkaita tai 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	• <i>Personointi ja käyttökonteksti.</i> 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 toiminnastaan hakukoneen personointiin ja kohdennettuun markkinointiin. Samoin käyttöpaikka, kuten maa, ja sen kieli voidaan huomioida haussa ja hakutuloksia järjestettäessä.

¹ Sibelius syntyi Hämeenlinnassa vuonna 1865, ja siellä on monia häneen liittyviä kulttuurikohteita ja tapahtumia.

² Mannerheim esimerkiksi nimitti Gallen-Kallelan pääesikunnan kirjapainon esimieheksi, ja Gallen-Kallela toimi Mannerheimin adjutantina.

M23:H26069:467

museo	Museo23;
objekti	kuppi ja lautanen;
materiaali	posliini;
valmistuspaikka	Saksa;
valmistaja	Meissen.



Kuva 3.1. Museoesineen tietoja kokoelmassa. (Kuva: Kansallismuseo.)

Tarkastellaan esimerkkinä hakua museokokoelmasta, johon on talletettu muun muassa kuvan 3.1 mukainen esine. Kokoelmaan tehdään 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 tekstikenttensä hakua varten. Kysely (2) esimerkiksi tehdään niin, että objekti-kenttään kirjoitetaan ”astia” ja materiaali-kenttään ”keramiikka”. Perinteistä tekstihakua käyttämällä yhteenkään kyselyistä 1–3 ei kuitenkaan saada vastauksena kuvan esinettä, vaikka ihmishakija niin kuvittelisikin jokaisessa tapauksessa: Kyselyssä (1) kone ei ymmärrä kohteen olevan esine, vaikka siihen kuuluvat 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 eräs usein toimiva yksinkertainen ratkaisu, kuten esimerkiksi Googlen menestyksekkäs käyttöliittymä on osoittanut. Moniin käyttötapauksiin on kuitenkin kehitetty parempiakin ratkaisuja, kuten hakujen muotoilu hierarkkisista hakufaseteista valintoja tehden niin sanotussa *fasettihaussa* (faceted search) ja erilaiset graafiset hakukäyttöliittymät, kuten kartta-

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. Esimerkiksi karttoja voidaan käyttää paikkatietoa haettaessa, aikajanoja ajasta riippuvaisen tiedon esittämiseen, tuloksia ryhmitellä aineistotyyppin 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.¹
- *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, eivätkä vanhat sivut 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 sivusto² ei sisältäne yhtä luotettavaa tietoa kuin vaikkapa *Nature*-lehden tiedesivut. Esimerkiksi yhteisöllisesti tuotetun Wikipedian ja toimituksellisten tietosanakirjojen, kuten Encyclopedia Britannican, luotettavuudesta on keskusteltu paljon ja vertailtu niitä toisiinsa.

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 verkos-

¹ Aihetta on tutkittu mm. hypertextijärjestelmien yhteydessä, ks. esim. (Edwards & Hardman, 1989).

² <http://www.theflatearthsociety.org/cms/>

sa olevan museon kokoelmätietokannan sisällöstä tai täsmällistä tiedonhaun tavoitetta, järkevien hakusanojen keksiminen Googlen kaltaiselle hakukoneelle voi olla vaikeaa. Silloin olisi voitava löytää kokoelmia selailmalla 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 tulkitsemaan niitä. ”Semanttinen web” tarkoittaa siis koneellisesti tulkittavissa olevaa webiä tai webiin upotettua datakerrosta. Tässä yhteydessä puhutaan usein *metatiedosta* eli *metadatasta* (metadata), joka on tietoa tiedosta. Rajanveto datan ja metadatan välillä ei kuitenkaan aina ole selkeää, sillä metadatalakin viitataan web-ympäristössä koneellisesti tulkittavaan dataan.

metadata

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

semantiikka
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

Ontologioiden käsitteitä voi hakea ja tutkia näppärästi verkossa olevien ontologia- ja sanastopalveluiden avulla.¹ Esimerkiksi kuvankaappauksessa 3.2 käyttäjä on hakenut Kansalliskirjaston ylläpitämästä

¹ Ontologiapalveluihin palataan tarkemmin teoksen viimeisessä osassa semanttisen webin infrastruktuurien esittelyn yhteydessä.

Finto-palvelusta hakusanalla ”kissa” KOKO-nimisen ontogian käsitteitä. KOKO on kansallisessa FinnONTO-hankkeessa kehitetty, kymmeniä tuhansia käsitteitä sisältävä ontologia, joka perustuu Suomessa käytössä oleviin eri alojen asiasanastoihin. Ensimmäisen hakutuloksen (”kissa”) kuvaus kertoo kissa-käsitteen yläkäsitteen (”kissaeläimet”) ja alakäsitteet (kuten ”burmankissa”). Tämän jälkeen luetellaan käsitteen aiheeseen läheisesti liittyviä käsitteitä (esim. ”kulkukissat”), käsitteeseen ohjaavia termejä (kuten ”kesykissa”) sekä käsitteen ”kissa” englannin-, latinan-, ja ruotsinkieliset nimikkeet.

The screenshot shows the Finto KOKO-ontology web application. The header includes the Finto logo and navigation links for Sanastot, Tietoja, Palaute, and Ohje. The main content area is titled 'KOKO-ontologia' and features a search bar with the query 'kissa'. On the left, there are filters for 'Rajaus' (Osasanaston mukaan, Tyyppin mukaan, Yläkäsitteen mukaan) and 'Aseta rajoitukset'. The search results are displayed on the right, showing 9 results for the query 'kissa'. The results are organized into three sections: 'kissa', 'kissaeläimet', and 'kissamaki'. Each section lists related terms and their translations in English, Latin, and Swedish, along with a URL to the specific ontology entry.

Kuva 3.2. Kissa-käsitteen tietoja KOKO-ontologiassa Finto-ontologiapalvelussa. (<http://finto.fi>.)

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

Keskeinen ontologioiden etu on, että kyselyä voidaan *laajentaa* (query expansion) käsitteeseen liittyvien synonyymien ja semanttisten suhteiden kautta. Esimerkiksi hakusanalla ”kesykissa” voidaan hakea samalla dokumentteja, joissa esiintyy termi ”kissa” tai ”kotikissa”, tai englanninkielisissä teksteissä ”cat”. Homonymien osalta menettely mahdollistaa monimerkityksisten termien tunnistamisen sellaisina käsitteinä, joilla on

kyselyn laajentaminen

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 toisistaan eli *disambiguoida* (disambiguate). 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-sanana disambiguointi olisi lähtökohdaisesti vaikeampaa, koska se edellyttäisi käyttökontekstin analysointia.

disambiguointi

Käsittehaiku: Hae ☐ tarkenna hakua

Hakusana: *nokia* (poista)

Valmistaja > ... > [Nokia](#) (14),
 Valmistuspaikka > ... > [Nokia](#) (32),
 Käyttöpaikka > ... > [Nokia](#) (4),
 Valmistaja > ... > [Oy Nokia Ab](#) (1),
 Valmistaja > ... > [Nokia-Mobira](#) (1),
 Valmistaja > ... > [Nokian Jalkinetehtas Oy](#) (7),
 Valmistaja > ... > [Nokian Kutomo ja Värjäys Oy](#) (2)

Kuva 3.3. MuseoSuomi disambigui hakutermiä ”Nokia” merkitykset: Nokia valmistajana (yrityksenä), valmistus- tai käyttöpaikkana (kaupunkina) sekä eri valmistajina, joiden nimen osana on merkkijono ”Nokia”. (<http://www.museosuomi.fi>)

Disambiguointia tarvitaan toisaalta dokumenttien indeksoinnissa, toisaalta kyselytermien merkityksen selvittämisessä. Jos jompikumpi tai molemmat jää tekemättä, hakutuloksen tarkkuus heikkenee, sillä mukaan tulee helposti vääriä osumia. Esimerkiksi MuseoSuomi-järjestelmässä¹ museokokoelmien kohteiden merkitykset on disambiguitu ja tieto on indeksoitu merkityksen perusteella. Kun käyttäjä kirjoittaa tekstihakukenttään monimerkityksisen termin, kuten ”Nokia”, ei kone voi tietää, mihin merkitykseen käyttäjä sanalla viittaa, mutta koska merkitykset on erotettu niiden nimikkeistä, systeemi tunnistaa mahdolliset eri merkitykset ja voi pyytää käyttäjältä apua disambiguointiin. Esimerkiksi kuvassa 3.3 käyttäjä on kirjoittanut ”Nokia” käsittehaiku-kenttään, jonka alle kone on tulostanut kyselyn mahdolliset tulkinnot ja valintoja vastaavien tulosjoukkojen koot suluissa. Viimeinen vaihtoehto tarkoittaa

¹ <http://www.museosuomi.fi>

tulkintaa, jossa ”Nokia” viittaa Nokian Kutomo ja Värjäys -nimiseen osakeyhtiöön esineen valmistajana. Linkkiä painamalla tulokseksi tulee tarkasti vain kaksi tekstiiliä (ks. kuva 3.4), pusero ja yöpuku, eikä esimerkiksi mobiililaitteita tai kumisaappaita, joita löytyy muiden vaihtoehtojen kautta.¹

Hakuehdot

Kategoria: Valmistaja > yritykset > *Nokian Kutomo ja Värjäys Oy* (ryhmittele kohteet) (poista)

Kohteet ryhmiteltyinä kategorian *Nokian Kutomo ja Värjäys Oy* mukaisesti

(näytä ilman ryhmittelyä)

Nokian Kutomo ja Värjäys Oy, kohteet 1-2/2



Pusero,
lapsen:neulepusero,
tytön (LKM LHM
LHM ES 97070 331)



Yöpuku,
naisen:trikooyöpaita
(LKM LHM LHM ES
97070 354)

Kuva 3.4. Hakutulos valinnalla Nokia → Nokian Kutomo ja Värjäys Oy. (<http://www.museosuomi.fi>)

MuseoSuomen tapauksessa kohteiden kuvailussa käytetyt merkitykset disambiguoitiin puoliautomaattisesti aineiston tuotantoprosessin aikana niin, että kone tunnisti monimerkityksiset (tai ontologiaan kuulumat) ilmaukset ja antoi ihmisluettelijan tehdä korjaukset niiltä osin, kun disambiguointia ei voitu tehdä automaattisesti. Yleensä termien semanttinen disambiguointi vapaata tekstiä käsiteltäessä on haasteellista, ja asiaa tutkitaan aktiivisesti.

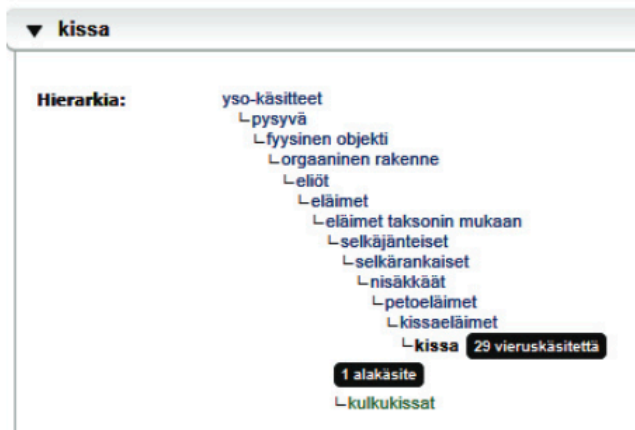
Käsitteiden määrittely ja niiden välisten suhteiden esittäminen tehdään semanttisessa webissä niin sanottujen ontologioiden avulla.² Tällöin käsitteet esitetään tyypillisesti *luokkina*, jotka edustavat *yksilöiden-*

¹ MuseoSuomen ideaa ja ominaisuuksia on selostettu tarkemmin artikkelissa (Hyvönen ym., 2005). Sovellus sai vuonna 2004 Japanissa kansainvälisen Semantic Web Challenge -teknologiapalkinnon.

² Käsitteitä määrittelevät ontologiat ovat semanttisen webin keskeinen elementti, joita ”ymmärtävät” niin ihmiset kuin koneetkin. Ontologioita esitellään tarkemmin myöhemmissä luvuissa.

sä 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 hierarkkinen suhde on alaluokka-yläluokkasuhde, jolla voidaan esimerkiksi kertoa, että kupit ja lautaset (alaluokkia) ovat astioita (yläluokka), jotka taas ovat esineitä (*astia*-luokan yläluokka). Kuvassa 3.5 on esimerkkinä esitetty *kissa*-käsitteen yläluokat ja alaluokka KOKO-ontologiassa ONKI-ontologiapalvelimen näyttämänä. Käsitteellä on lisäksi 29 vieruskäsitettä, kuten *leijona*.

yksilö

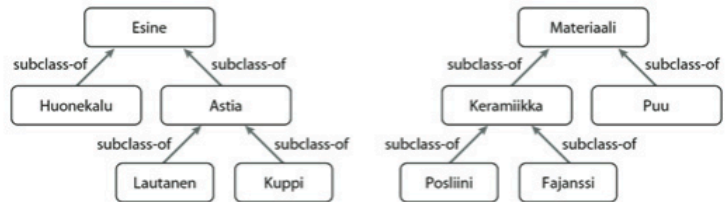


Kuva 3.5. Kissa-käsitteen yläkäsitteitä ja alakäsite KOKO-ontologiassa. (<http://onki.fi>.)

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

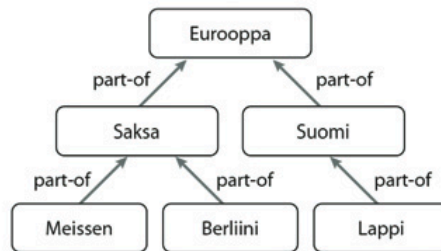
kyselyn laajentaminen

Tarkastellaan esimerkkinä kyselyn laajentamisesta kuvan 3.1 tapaus-
ta, jossa tekstihaku ei löytänyt esinettä museon kokoelmatiedoista. Ole-
tetaan, että käytettävissä ovat esineet ja materiaalit hierarkkisesti ku-
vaavat ontologiat, joissa kullekin käsitteelle on kerrottu sen yläluokka
subclass-of-suhteella kuvan 3.6 mukaisesti. Esimerkiksi *lautasen* yläluok-
ka on *astia*. Jos kyselyt laajennetaan alaluokkiin päin, kupit ja lautaset
voidaan löytää esineitä haettaessa ja tunnistaa posliini eräänlaiseksi ke-
ramiikaksi (esimerkit 1 ja 2).



Kuva 3.6. Esine- ja materiaalityyppien luokkahierarkiat.

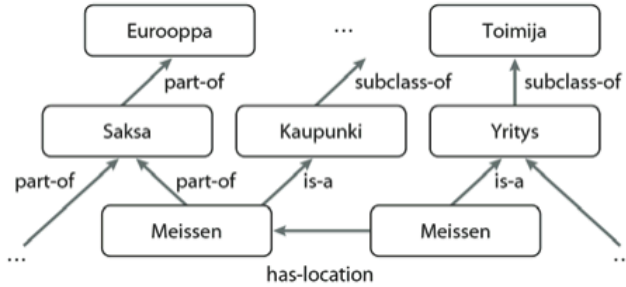
Toinen sovelluksissa paljon käytetty semanttinen suhde on osa-
kokonaisuus-suhde (part-of), jolla voidaan esimerkiksi kertoa, että Meis-
senin kaupunki on osa Saksaa, joka on osa Eurooppaa (kuva 3.7). Näin
löytyy Saksa Euroopan osana, ja kuvan 3.1 yhteydessä esitetyn hakuesi-
merkin 3 haaste löytää Saksa Eurooppa-hakusanan kautta ratkeaa.



Kuva 3.7. Paikkaontologia, jossa kuvataan alueellista kuuluvuutta.

Ontologioissa on usein tärkeää erottaa yksilökäsitteen käsiteluokkaan
kuulumista ilmaiseva suhde (is-a) yläluokkasuhteesta (subclass-of). Esi-
merkiksi toimijaontologiassa olevat yksittäiset yritykset tai henkilöt ei-
vät ole *luokkia* (class, type) vaan yritys- ja henkilöluokan *yksilöitä* (indi-

vidual, instance). Kuvan 3.8 ontologiassa solmu Meissenin oikealla puolella on Yritys-luokan yksilökäsite ja vasemmalla Kaupunki-luokan; ontologia voi näin erottaa samannimisen yrityksen kaupungista.



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

Merkitykset erottamalla on mahdollista kertoa tietokoneelle, että Meissen-yrityksen valmistamat esineet on valmistettu Meissenin kaupungissa, jossa yritys sijaitsee. Näin voidaan löytää ilman sekaannusta eri paikoissa valmistetut ja eri toimijoiden valmistamat esineet.

Rakenteinen tieto mahdollistaa myös haun kohdistamisen dokumenttien ohella niiden sisältämän datan osiin. *Entiteettihauksa* (entity search) esimerkiksi haetaan yksilökohteista eli entiteeteistä kertovien dokumenttien sijasta itse yksilöiden kuvauksia, kuten paikkoja, henkilöitä, yhtyeitä, teoksia ja niin edelleen. 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” tietolaatikkossa eikä vain listausta dokumenteista, joissa esiintyvät sanat ”who”, ”is” ja ”Kekkonen”. Hakukysymykseen ”Who is Kekkonen’s spouse” tulee vastaukseksi entiteetti ”Sylvia Salome Uino”.

entiteettihaku

Objektien ohella voidaan hakea myös niiden välisiä suhteita. Kuvassa 3.9 on esimerkki Kulttuurisampo-järjestelmän *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 yhteyshakun järjestelmän toimijaontologian kautta, jossa kuvataan muun muassa hen-

yhteyshaku

kilöiden välisiä sosiaalisia suhteita.¹ Tässä tapauksessa datana hyödynnettiin yhdysvaltalaisen Getty-säätiön ULAN-rekisteristä (Union List of Artist Names) muodostettua ontologiaa.²

Henkilö 1	Gallen-Kallela, Akseli (suomalainen taiteilija, 1865-1931)
Henkilö 2	Napolean I, Emperor of the French (ranskalainen suosija, 1769-18)
	Gallen-Kallela, Akseli (suomalainen taiteilija, 1865-1931) <i>jonka opettaja on</i>
	Becker, Adolf von (suomalainen taiteilija, 1831-1909) <i>jonka oppilas on</i>
	Berndtson, Gunnar (suomalainen taiteilija, 1854-1895) <i>jonka opettaja on</i>
	Gérôme, Jean-Léon (ranskalainen taiteilija, 1824-1904) <i>jonka oppilas on</i>
	Burnand, Eugène (sveitsiläinen taiteilija, 1850-1921) <i>jonka opettaja on</i>
	Menn, Barthélemy (sveitsiläinen taiteilija, 1815-1893) <i>jonka opettaja on</i>
	Ingres, Jean-Auguste-Dominique (ranskalainen taiteilija, 1780-1867) <i>jonka suosija oli</i>
	Napolean I, Emperor of the French (ranskalainen suosija, 1769-1821)

Kuva 3.9. Kulttuurisammon yhteyshaku hakee henkilöiden välisiä yhteyksiä. (<http://www.kulttuurisampo.fi>.)

Tiedon haussa on viime kädessä 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 jolta saadaan tuloksena vastaus. Esimerkiksi: ”Kuka voitti keihäänheiton Helsingin olympialaisissa?” Askel tähän suuntaan on esimerkiksi WolframAlpha-hakukone, jolta voi kysyä vaikka ”Who was the president of the US in 1856?” tai ”How much does one kilogram weight on the Moon?” Täl-

¹ Vuonna 2007 julkistettu Kulttuurisampo (<http://www.kulttuurisampo.fi>) yhdistää erilaisia heterogeenisiä suomalaisia kulttuurisisältöjä semanttisessa webissä (Hyvönen ym., 2009; Mäkelä ym., 2012) ja oli FinnONTO-ohjelman laaja-alaisin sovellus.

² ULAN-rekisteri sisältää yksityiskohtaista tietoa n. 120 000 historiallisesta henkilöstä. Aineisto on nykyisin saatavilla linkitettyinä avoimena datana ja on käytettävissä semanttisen webin standardien mukaisen SPARQL-datapalvelun kautta: <http://www.getty.edu/research/tools/vocabularies/ulan/>.

laisten järjestelmien kehittämisessä ei ole enää kysymys dokumenttien hausta vaan niiden sisältävän rakenteisen semanttisen informaation käsittelystä.

Rakenteista semanttista dataa hyödynnetään myös IBM:n Jeopardy-peliä pelaavassa, parhaat ihmisasiantuntijat voittaneessa tekoälyjärjestelmässä Watson. Pelissä on tehtävänä löytää kysymyksiä annettuihin vastauksiin. Watsonin 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 personoidun käyttötilanteen, esimerkiksi paikan ja ajan, mukaan.

II

LINKITETTY DATA

4

Linkitetyn datan esittäminen

Ei ole olemassa vastauksia, on
vain viittauksia.

Norbert Wiener (1894–1964)

Tässä luvussa kerrotaan ensin, miten tietoa voidaan esittää semanttisena verkkona. Sitten esitellään 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

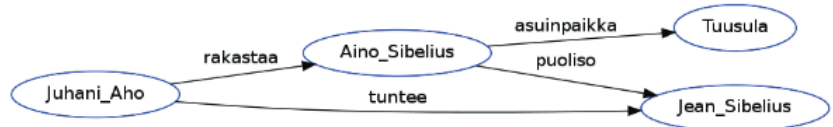
Semanttisen webin keskeisenä ideana on verkossa olevan *tietämyksen esittäminen* (knowledge representation) *linkitettyinä datana* (linked data) sellaisessa muodossa, että tietokoneet voivat sitä tulkita. Tiedon esittämisen perustana on yksinkertainen tietomalli, *semanttinen verkko* (semantic net).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

4.1.1 Verkko kaaviona

solmu

kaari

Verkkoa kannattaa yleensä ajatella ja esittää havainnollisesti kaaviona, jossa on nimettyjä *solmuja* (node) ja näitä yhdistäviä, suunnattuja nimettyjä *kaaria* (arc). Esimerkiksi kuvassa 4.1 on esitetty yksinkertainen verkko, joka kuvaa Tuusulanjärven alueella 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>}

```

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

subjekti
predikaatti
objekti

$\langle \text{subjekti}, \text{predikaatti}, \text{objekti} \rangle$

Tällaisten kolmikoiden voidaan myös tulkita liittävän solmuihin nimettyjen ominaisuuksien arvoja:

$\langle \text{solmu}, \text{ominaisuus}, \text{ominaisuuden arvo} \rangle$

Luettelemalla verkon kaikkien solmujen ominaisuudet tällä tavalla voidaan mielivaltainen verkko esittää yksinkertaisesti joukkona kolmikoita. Kolmikkojoukkojen käsittely on erittäin yksinkertaista ohjelmallisesti, ja kolmikkomuotoisen tiedon käsittelyyn tarkoitettut ohjelmakirjastot ja työkalut kuten esimerkiksi Jena¹ perustuvat tähän. Kolmikoiden käsitte-
lyyn 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 tahojen julkaisemia verkkotietoja 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 *lauseiksi* eli *väittämiksi* (statement), joissa kaari kertoo alku- ja päätesolmun välisen yhteyden pitävän
paikkansa. Esimerkiksi väittämä *Juhani Aho rakastaa Aino Sibeliusta* voidaan kirjoittaa logiikan predikaattina muodossa:

väittäjä
lause

`rakastaa(Juhani_Aho, Aino_Sibelius)`

Näin semanttisille verkoille voidaan luoda täsmällisesti määritelty tulkinta. Logiikka mahdollistaa muun muassa uuden tiedon muodostami-

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

sen 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

sarjallistaminen

Jotta tietokone voisi lukea muistiinsa verkon tiedostosta ja vastaavasti tallentaa verkon tiedostoon myöhempää käyttöä varten, pitää verkko voida esittää tekstimuodossa merkkijonona eli *sarjallistaa* (serialize).

Argumentti 1	Predikaatti	Argumentti 2
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.

Verkot sarjallistetaan semanttisessa webissä tätä varten kehitettyjen kielten avulla, joita ovat muun muassa XML-perustainen RDF/XML, N-Triples ja erityisesti Turtle, joihin palataan tarkemmin seuraavassa luvussa. Esimerkin verkko voitaisiin esittää yksinkertaisesti kirjoittamalla verkon solmujen tunnisteet kulmasulkujen sisään ja luettelemalla kolmiot pisteellä eroteltuina:

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

CSV-muoto

Verkko voidaan esittää luontevasti myös taulukkona, jonka rivit kuvaavat kolmikoita (taulukko 4.1). Taulukkolaskimessa tällainen tieto sarjallistetaan *CSV-muodossa* (Comma Separated Values). Siinä jokainen laskimen taulukon rivi on esitetty niin, että sarakkeiden arvot on eroteltu toisistaan pilkuilla (comma).¹ Esimerkiksi kuvan 4.1 ja taulukon 4.1 data voidaan sarjallistaa neljällä eri rivillä näin:

¹ Mikäli sarakkeiden arvoissa on pilkkuja, nämä voidaan huomioida omalla lainausmekanismilla. Sarakkeiden *erottimena* (delimiter) voidaan myös sopia käytettävän jokin muuta merkkiä kuin pilkku.

```
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 Framework*, lyhyesti *RDF*.¹ Olennaista on huomata, että semant-

tisessa webissä on kyse tiedon esittämisestä verkkoperustaisella RDF-tietomallilla, joka voidaan kuvata käyttämällä erilaisia *syntaktisia kieliiä*. 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 on (meta)kieli erilaisten merkkauksien määrittämiseksi ja käsittelemiseksi, mutta RDF on tietomalli, joka voidaan esittää erilaisilla syntaktisilla kielillä.

RDF

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

¹ RDF-suositukset on julkaistu osoitteessa <https://www.w3.org/RDF/>, jonka kautta löytyy runsaasti muitakin aiheeseen liittyvää tietoa ja työkaluja.

4.2.1 HTML-kieli

HTML *HTML-kieli* (Hypertext Markup Language) tarjoaa välineet web-sivujen ja niiden välisten linkkien esittämiseen, jotka sitten näytetään käyttäjälle selaimen (browser) avulla. HTML on niin sanottu *merkkauškieli* (markup language), jossa tietosisällön sekaan on upotettu tiedon rakennetta ja ilmiä suua kuvaavia sisäkkäisiä *merkkauksia* (tag). Merkkaus alkaa kulmasulkuihin kirjoitetulla alkumerkkauksella `<merkkkaus>` ja päättyy loppumerkkaukseen `</merkkkaus>`. Esimerkiksi ylimmän tason 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 kappaletekstin pienemmällä. Olennaista on, että ilmiä sun muodosta päätetään vasta päätelaitteessa: esimerkiksi älypuhelimien näytöllä 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 merkkauškieliä. Kansainvälinen ISO-järjestö standardoi SGML-kielen jo ennen webin tuloa vuonna 1986.¹ Nykyisin HTML-kielestä on käytössä alkuperäistä monipuolisempi versio HTML5.² Se perustuu W3C-järjestön vuonna 1998 standardoimaan XML-metakieleen ja standardiperheeseen.³ XML on yksinkertaistettu versio SGML-metakielestä.

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 *verkkotunnuksia* eli *aluenimiä* (domain name). Esimerkiksi Aalto-yliopisto on rekisteröinyt käyttöön-

¹ ISO 8879:1986

² <https://www.w3.org/TR/html5/>

³ HTML5-dokumentin ei kuitenkaan tarvitse olla validia XML:ää, ellei dokumentissa kerrota sen olevan XHTML5-kieltä – esimerkiksi loppumerkkauksia voi jättää pois.

sä aluenimen `aalto.fi`. Siksi kaikki HTTP-URL osoitteet, jotka alkavat tuolla nimellä, kuten

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

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.

Nimen haltijalla on mahdollisuus ottaa käyttöönsä *edustapalvelin* (reverse proxy), joka tarvittaessa ohjaa alueelle tulevan HTTP-kutsun uudelleen (redirect) toiseen osoitteeseen. Ohjauksessa voidaan ottaa huomioon muun muassa 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 muun muassa tallettaa usein kysyttyjä sivuja paikallisesti ja näin nopeuttaa tiedonhakua. *Turvapalvelin* (security server) taas on verkon tietoturvaa hoitava erillinen palvelin.

alialue

edustapalvelin
uudelleenohjaus

välipalvelin

turvapalvelin

4.2.3 Yhtenäiset osoitteet ja tunnisteet: URI

HTML-sivujen sijainti esitetään internetissä systemaattisesti *URL-osoitteina* (Uniform Resource Locator). Kun osoite kirjoitetaan esimerkiksi selaimen 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, määrittävät niin sanottujen *URI-skeemojen*¹ (URI schema) avulla. Näitä on otettu käyttöön kymmeniä erilaisia, kuten URN-tunnisteet, OID-tunnisteet, GIT-tunnisteet ja BITCOIN-tunnisteet. Käytetyin tunnistemuoto ovat

URI-tunniste
URI-skeema

¹ <https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml>

kuitenkin web-osoitteina toimivat HTTP-tunnisteet (URL-tunnisteet) (ja tietoturvasemmät HTTPS-tunnisteet). Semanttisessa webissä pyritään käyttämään erityisesti niitä tunnisteina. Tähän on kaksi painavaa syytä:

1. HTTP-osoitteiden aluenimiin (verkkotunnuksiin) perustuva kansainvälinen webin nimipalvelujärjestelmä takaa automaattisesti sen, ettei kukaan muu kuin alueen, esimerkiksi `dbpedia.org`, rekisteröinyt käyttäjä voi ottaa käyttöön toista samanlaista tunnistetta. Toisin sanoen 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.
2. HTTP-tunnisteeseen sisältyy paitsi itse tunniste myös verkko-osoite, johon on mahdollista tallentaa lisätietoa tunnisteesta. Tiedon saa siten näkyville ja käyttöön pelkän tunnisteiden 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.

Seuraavassa on esitetty yhteenvetona URI-skeemojen yleinen rakenne kaavana:

skeema:hierarkiaosa[?kysely][#fragmentti]

Kaavassa kaksoispiste erottaa skeeman nimen hierarkiaosasta ja hakasulut rajaavat valinnaisia osia *kysely* ja *fragmentti*, jotka tunnistaa alkumerkeistä "?" ja "#". Skeemoja ovat muun muassa `http`, `urn` ja `mailto`. Hierarkiaosa koostuu tyypillisesti aluenimestä (domain), esimerkiksi `aalto.fi` tai `www.helsinki.fi`, ja tätä seuraavasta kauttavivalla alkavasta hakemistopolusta palvelimella, kuten esimerkiksi `http://aalto.fi/fi/research/`. Aluenimen jälkeen voidaan kaksoispisteellä erottaen vielä määritellä numeerinen portti, johon HTTP-kutsu lähetetään.¹ Esimerkiksi:

¹ Oletusporttina HTTP-protokollassa on 80 ja HTTPS-protokollassa 443.

<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*-parina, jotka on erotettu toistaan &-merkillä. Tyypillinen käyttötapa 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 -palvelussa¹ käytetään EU:ssa kehitettyjä ELI-² ja ECLI-tunnisteita³. 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),⁴ 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

¹ Palvelu on käytettävissä osoitteessa <http://data.finlex.fi>, ja sen ominaisuuksista ja toteutuksesta löytyy lisätietoa sivulta <http://seco.cs.aalto.fi/projects/lawlod>.

² http://en.wikipedia.org/wiki/European_Legislation_Identifier, European Legislation Identifier.

³ http://en.wikipedia.org/wiki/European_Case_Law_Identifier, European Case Law Identifier.

⁴ Ankkuri on HTML-kielen merkkauksena, jonka avulla URL-osoitteissa voidaan viitata eri kohtiin WWW-sivuilla.

URI-tunnisteiden parametreina välitetään usein tekstidataa palvelimelle, esimerkiksi hakutermejä hakukoneelle tai toinen URI-tunniste uudelleenohjausta varten. Tällaisessa datassa ei välttämättä voi käyttää URI-tunnisteissa muuhun käyttöön varattuja tai muita erikoismerkkejä, kuten esimerkiksi välilyönti, &-merkki, kaksoispiste ja kauttaviiva. 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

`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 muunnospalveluiden¹ sivuilla.

URN-skeema

Kukin URI-skeema määrittelee omanlaisensa tunnisteiden kirjoitusasun (rakenteen). URL-muotoisen tunnisteiden 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 tapaan kuin henkilötunnukset. Esimerkiksi julkaisujen ISBN-numeroille on Suomessa käytössä muun muassa seuraava URN-tunniste:

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

Seuraava taas 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 selaimen lisätiedon toivossa, koska se ei sisällä osoitetietoa. URN-tunnisteen *tulkittamiseksi* (resolve) tarvittaisiin

¹ Esim. <http://meyerweb.com/eric/tools/dencoder/>.

erillinen maailmanlaajuinen URN-tulkintajärjestelmä, joka olisi paljolti 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 käyttöön 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 turvallisesti mielin luoda hajautetusti ja itsenäisesti tunnisteita hallinnassaan olevilla nimialueilla (esimerkiksi `helsinki.fi` tai `nokia.com`) ilman pelkoa tunnisteiden sekoittumisesta.

Tällöin eri tahot voivat kuitenkin paikallisesti luoda useita eri tunnuksia samalle asialle, vaikkapa Sibeliuksen viidennelle sinfonialle tai Espoon Otaniemelle. Tämä aiheuttaa haasteita tunnisteiden samastamisessa, jos eri toimijoiden tietoja myöhemmin yhdistetään. Esimerkiksi samalle kirjailijalle on yleensä käytössä erilaisia tunnisteita eri valtioiden kirjastojen, arkistojen ja museoiden rekistereissä (niin sanotuissa 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 hiraganatavukirjoitusta.¹ Perinteisissä URI-tunnisteissa tyydytään käyttämään vain yksinkertaista ASCII-merkistöä, josta puuttuvat muun muassa ääkköset.² Haasteena erikoismerkkejä sisältävien IRI-tunnisteiden käytölle ovat eri maissa käytettävät erilaiset näppäimistöt. Esimerkiksi amerikkalaisissa näppäimistöissä ei ääkkösiä näy, jolloin niitä sisältävän HTTP-

IRI-tunniste

¹ Unicode-merkistöön kuuluu nykyisin yli 128 000 merkkiä 135 kirjoitusjärjestelmäästä.

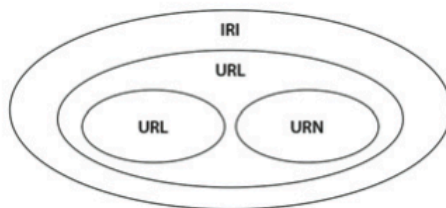
² ASCII-merkistöön kuuluu 128 (tai laajennettuna 256) merkkiä.

osoitteen syöttäminen vaikkapa selaimeen vaatii erityistoimenpiteitä. Samoin arabiankielisen 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 samantekevää, 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 ja ASCII-järjestelmää ilman erikoismerkkejä. Tämän vuoksi tässä teoksessa puhutaan URI-tunnisteista. IRI-tunnisteisiin viitataan vain silloin, jos on tarvetta korostaa erikoismerkkien käyttöä.

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



Kuva 4.2. Webin eri tunnistetyyppiä joukkoina ja niiden sisältyvyys toisiinsa.

4.2.4 Resurssien ja literaalien esittäminen

Resurssi (resource) on semanttisessa RDF-verkossa oleva solmu, joka voidaan esittää graafisesti soikiona. Resurssin ominaisuudet ilmaistaan solmusta lähtevien kaarien avulla. Tarkastellaan esimerkkinä Wikipediasta louhittua RDF-verkkoa DBpedia, joka on julkaistu ja saatavilla verkossa

avoimena linkitettyinä datana.¹ Esimerkiksi elokuva *Tuntematon sotilas* (alkuperäinen vuoden 1955 versio) on esitetty DBpediassa resurssilla, jonka tunniste on:

[http://dbpedia.org/resource/The_Unknown_Soldier_\(1955_film\)](http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film))

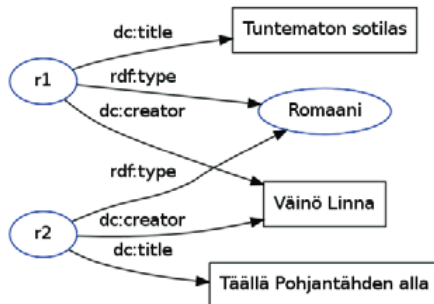
Tuntemattoman sotilaan resurssista, toisin sanoen verkon solmusta, lähtee runsaasti kaaria eli resurssin *ominaisuuksia* (property), jotka kuvaavat metatietoa elokuvasta, kuten että elokuvan nimi on *The Unknown Soldier (1955 film)*, sen pituus on 10 140 sekuntia ja sen ohjasi *Edvin Laine* ja tuotti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

ominaisuus

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

literaali

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.



Kuva 4.3. Semanttinen verkko.

Literaalisolmut esitetään usein graafisesti suorakaiteen muotoisina solmuina, jotta ne erottuisivat selkeästi soikeista resurssisolmuista. Esimerkiksi kuvassa 4.3 on esitetty tietoa romaaneista *Tuntematon sotilas* (1954) ja *Täällä Pohjantähden alla* (1959–1962). Mukana on pyöreiden resurssisolmujen ohella myös literaaleja, jotka on esitetty suo-

¹ <http://dbpedia.org>

rakaiteina. Kuvan solmu *r1* edustaa *Tuntematon sotilas* -romaanina, ja sen ominaisuudet kertovat, että kyse on romaanista (`rdf:type`), että teoksen kirjoitti (`dc:creator`) Väinö Linna ja että teoksen nimi (`dc:title`) on merkkijono ”Tuntematon sotilas”. Tässä esimerkiksi romaani-käsitteeseen *Romaani* ei liity muuta tietoa kuin 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.

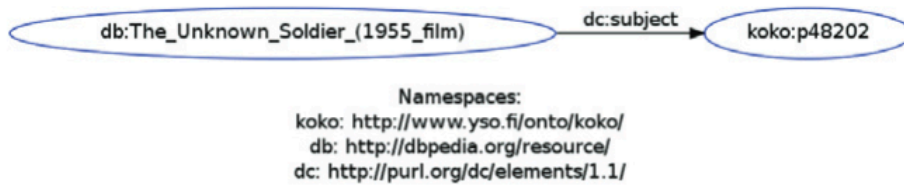
Literaallilla tarkoitetaan merkkijonoa, lukua, päiväystä tai muuta perustietoa, jota käytetään ominaisuuden arvona. Alla ovat esimerkkeinä Helsingin nimeä, piin likiarvoa ja päiväystä kuvaavat literaalit:

"Helsinki"
3.14
1917-12-06

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 vaikkapa, että Väinö Linna syntyi vuonna 1920. Tiedot pitäisi liittää Väinö Linnan identifioivaan resurssisolmuun. Literaalidatan kautta datat eivät 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 olisi kirjoittanut teoksen *Tuntematon sotilas* ja toinen *Täällä Pohjantähden alla*.

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



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

englanninkielisen 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*). Solmujen tunnisteiden alku-osat *db:*, *dc:* ja *koko:* viittavat kuvan alaosassa ilmaistuihin *nimiavaruuksiin* (namespace) eli web-osoitteisiin, joissa resurssit on tarkemmin määritelty. Nimiavaruuksiin palataan tarkemmin tuonnempana.

Sota-käsitteen kuvaus RDF-muodossa löytyy tunnisteiden mukaisesta osoitteesta ja kertoo muun muassa 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* ja niin edelleen. 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 ne voidaan yhdistää toisiinsa. Tällaisilla datajoukkojen välisillä linkityksillä voidaan esimerkiksi mukana yhdistää englanninkielisen DBpedian aineistoja kotimaiseen ontologiainfrastruktuuriin, jolloin dataa voidaan rikastaa semanttisesti toistensa avulla.

Olennaista on huomata tunnisteiden ja niiden ihmishuettavien literaalisten *nimikkeiden* (label) välinen ero RDF-verkoissa: tunnisteet, kuten

nimike

<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 kielille tarvitse kehittää omia verkkoja tai indeksejä tiedonhakua varten. Nimikkeiden avulla puolestaan voidaan rakentaa silta koneluettavien käsitteiden ja ihmisluettavien ilmausten välillä. RDF-tietomalli soveltuu tämän erottelun vuoksi 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 palvelimen kautta. Tällä on merkitystä muun muassa päättelyssä, jossa erilaisten kaarien avulla voidaan tehdä erilaisia johtopäätöksiä. Päättelyyn palataan tarkemmin ontologioiden esittelyn yhteydessä.

5

RDF-verkon esittämiskielet

Kieleni rajat merkitsevät
maailmani rajoja.

Ludwig Wittgenstein
(1889–1951)

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

syntaksi

notaatio

sarjallistaminen

5.1 Data kolmikkoina: N-Triples

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

N-Triples-notaatio

¹ <https://www.w3.org/TR/n-triples/>

RDF-verkossa merkkijonotyyppisellä literaalilla ei voi olla ominaisuuksia, mutta sillä voi kuitenkin olla kielen ilmaiseva valinnainen määre¹ muodossa:

literaali@kielitunnus

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

Seuraavassa esimerkissä englanninkielisen DBpedian RDF-verkosta poimitut kaksi kolmikkoa kertovat *Tuntematon sotilas* -elokuvan ohjaajan ja nimen. Kolmas kolmikko rikastaa kuvausta kertomalla filmin aiheen suomalaisen KOKO-ontologian mukaan. Tässä notaatiossa resurssit on merkitty URI-tunnisteilla kulmasulkujen sisään; merkkijonomuotoinen data taas on merkitty lainausmerkkien väliin. Lainauksen lopussa on merkkijonon kielen ilmaiseva tunnus, tässä englantia tarkoittava *@en*. *#*-merkillä alkavat tekstit saman rivin loppuun saakka ovat vain ihmislukijalle tarkoitettuja *kommentteja* (comment), jotka kone ohittaa merkityksettöminä.

```
# 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 Laine

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

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

N-Triples on yksinkertainen tapa RDF-verkon esittämiseksi, ja sen lukeminen tiedostosta on koneelle helppoa ja nopeaa yksinkertaisen syntaksin ansiosta. Kolmikoiden kirjoittaminen käsin tai niiden muodostaman kokonaisuuden hahmottaminen ei kuitenkaan ole välttämättä helppoa ihmiselle.

¹ Literaaliin voi liittyä myös tietotyyppi, kuten että kyseessä on kokonaishaku tai päivämäärä. Tietotyyppisiin palataan tarkemmin tuonnempana.

² <http://www.w3.org/International/articles/language-tags/>

5.2 Yksinkertaisempaa koodia: Turtle

Täydellisten kolmikoiden luetteleminen tekee N-Triples-koodista mutkasta, toisteista, työläästi kirjoitettavaa ja vaikeasti luettavaa. Esimerkiksi pitkät URI-tunnisteet haittaavat luettavuutta, ja subjekti-URI joudataan toistamaan jokaisen kolmikon yhteydessä. Tämän vuoksi käyttöön on otettu monia käteviä lyhennystapoja, jotka ovat käytävissä RDF:n *Turtle-notaatiossa*.¹ Turtle on nykyisin käytetyin ja helppolukuisin RDF-notaatio. Seuraavaksi esitellään käyttökelpoisia muotoja, joilla Turtle laajentaa ja yksinkertaistaa N-Triples-kielen ilmaisumahdollisuuksia.

5.2.1 Nimiavaruudet

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

1. Osoitteen loppuosaa viimeisen kauttaviivan tai #-merkin jälkeen kutsutaan *paikallisnimeksi* (local name).
2. Osoitteen alkuosan hierarkkista hakemistopolkua paikallisnimeen saakka kutsutaan *nimiavaruudeksi* (namespace).

paikallisnimi

nimiavaruus

Esimerkiksi tunnisteessa

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

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 URI:t voidaan ilmaista lyhennysmerkinnällä:

```
etuliite:paikallisnimi
```

Myös URI:en ympärillä olevat kulmasulut jätetään pois. Turtleen siirryttäessä tunnisteissa ja literaaleissa mahdollisesti käytetyt Turtle-notaatioon kuuluvat merkit, kuten välilyönnit, rivinvaihdot ja erilaiset sulut saattavat aiheuttaa erityisongelmia. Turtle-standardi mahdol-

¹ <http://www.w3.org/TR/turtle/>

listaa kaikkien Unicode-merkkien käyttämisen literaaleissa erityisellä \-merkillä alkavalla merkkauksella. Esimerkiksi rivinvaihto merkataan \n. IRI-tunnisteissa taas on käytössä oma lainausmekanisminsa.¹ Esimerkissä päänvaivaa aiheuttaa DBpedian käytäntö, jossa tunnisteet voivat sisältää sulkumerkkejä, joilla on Turtlessa muutakin käyttöä. Tässä tapauksessa ongelma hoituu yksinkertaisimmin jättämällä nuo tunnisteet N-Triples-muotoon, jota Turtle tukee sellaisenaan.

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 ilmaista paljon selkeämmin:

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

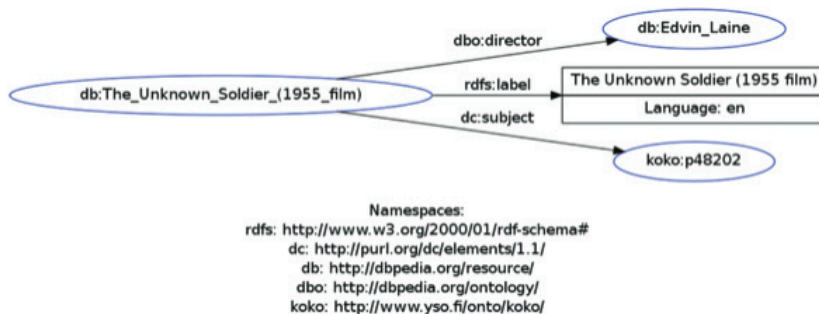
# Filmin nimi englanniksi
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
rdfs:label # Ominaisuus label kertoo nimikkeen
"The Unknown Soldier (1955 film)"@en . # Literaaliarvo

# Tuntemattoman sotilaan aiheena on sota
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
dc:subject #Aiheen kertova ominaisuus
koko:p48202 . # Käsité "sota" KOKO-ontologiassa
```

Nämä kolmikot on esitetty nimiavaruuksien avulla graafina kuvassa 5.1, joka on tuotettu Linked Data Finland -palveluun asennetulla RDF Grapher -visualisaattorilla² (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, RDF/JSON), joihin palataan tuonnempana. Myös tuotettavan kuvan formaatin voi valita (PNG, JPG, SVG, PDF, EPS jne.). Visualisaattorin avulla

¹ Ks. spesifikaatio <https://tools.ietf.org/html/rfc3987>.

² <http://www.ldf.fi/service/rdf-grapher/>



Kuva 5.1. Kolmikkoja nimiavaruuksien avulla esitettynä.

voi samalla *validoida* (validate) RDF-muotoista dataa, toisin sanoen tarkistaa, että se on syntaktisesti oikein muodostettua. Myös mun muassa W3C tarjoaa verkossa RDF-datan visualisointi- ja validointipalvelun.¹

validointi

Nimiavaruuksien avulla webin eri julkaisijat voivat ottaa vapaasti käyttöön (paikallis)nimiä resurssien tunnisteita varten (esimerkiksi *subject*) ilman pelkoa tunnisteiden sekoittumisesta, toisin sanoen siitä, että joku toinen olisi jo ottanut tunnisteiden käyttöön eri merkityksessä. Esimerkiksi ranskalaisen paikkatieto-organisaation luoma tunnus *france:Paris* voisi viitata Ranskan pääkaupunkiin, *texas:Paris* Texasissa olevaan samannimiseen kaupunkiin yhdysvaltalaisessa paikkatietorekisterissä ja *julkikset:Paris* perijätär Paris Hiltoniin.

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

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

Tuntematon sotilas -elokuvaan voi viitata muodolla:

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

RDF-määrittelyn (toisin sanoen tiedoston) oman eli *kantanimiavaruuden* (base) ilmaisemiseen käytetään muotoa:

kantanimiavaruus

```
@base nimiavaruus .
```

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

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

¹ <https://www.w3.org/RDF/Validator/>

jälkeen nimiavaruudeton lyhennetty tunniste

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

tarkoittaa samaa kuin:

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

5.2.2 Monta samaa ominaisuutta

Jos resurssilla on useita samannimisiä ominaisuuksia, ne voidaan luettelaa pilkuilla erotettuina. Esimerkiksi *Tuntemattoman sotilaan* erikieliset nimikkeet voidaan luotella muodossa

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

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

ilman, että tarvitsee kirjoittaa auki erillisiä kolmikoita.

5.2.3 Eri ominaisuudet samalla kertaa

Resurssin eri ominaisuuksia voidaan luotella näppärästi yhdellä kertaa puolipisteellä erotettuna, jolloin subjektia ei tarvitse toistaa moneen kertaan. *Tuntematon sotilas* -esimerkin kolme kolmikkoo 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/> .

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

5.2.4 Ominaisuuden arvo omilla ominaisuuksilla

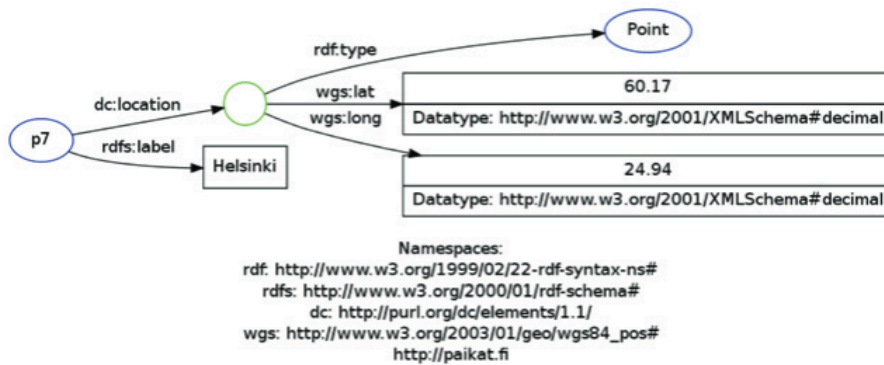
Ominaisuuden arvona oleva toinen resurssi ominaisuuksineen voidaan esittää hakasulkujen avulla erotettuna. Seuraavassa esimerkissä on kuvattu *Helsingin* paikkatieto tällaisen resurssin avulla:

```

@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
    ] .

```



Kuva 5.2. Graafi, jossa on nimetön solmu.

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 luoma tunniste, mikä yksinkertaistaa verkon tekstuaalista kuvausta. Tunniste on ainutkertainen URI-tunniste, johon ei ole mahdollista viitata muualta. Tästä ei ole haittaa, koska nimettömän solmun tunniste ei ole kiinnostava ja solmua käytetään ainoastaan keräämään yhteen pisteeseen liittyvää koordinaattitietoa kertaluontoisesti.

nimetön solmu

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 tunnisteen järjestelmä luo aina uudelleen. Se voikin 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 samastamaan.

5.2.5 Tietotyypit

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

literaali [^] *tietotyyppi*

Tietotyyppinä 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). Seuraavassa on liitetty resurssiin `:esim` erityyppisiä literaaleja, ja syntyvä graafi on visualisoitu kuvassa 5.3. Literaalin määreet on esitetty literaalia kuvaavan suorakaiteen sisällä.

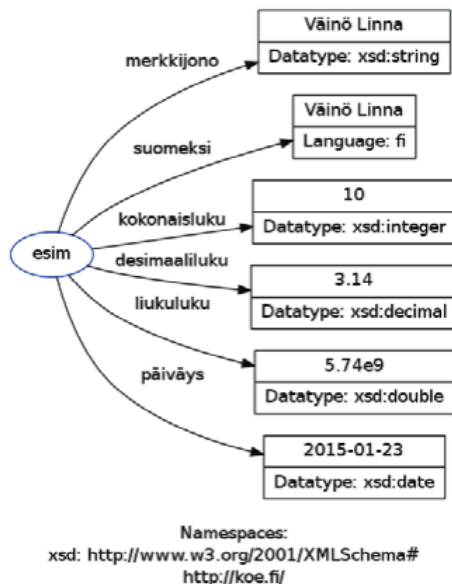
```

@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, järjestelmät osaavat asettaa tiedon tyypiksi jonkin numeerisen tyyppin, vaikkei tietotyyppiä olisi erikseen merkitty. Esimerkiksi 3.14 on desimaaliluku ja 3.14e1 vastaava liukuluku, jossa e (tai E) merkkää eksponenttiosuuden.



Kuva 5.3. Resurssin literaaliarvoja verkkona.

5.3 Verkon esittäminen RDF/XML-kielellä

Alkuperäinen RDF-kielen syntaksi määriteltiin XML-kielen avulla. Tämä notaatio on edelleen käytössä monissa yhteyksissä, kuten RDF-spesifikaatioissa, vaikka Turtlesta onkin tullut käytetyin tapa RDF-verkkojen esittämisessä. Seuraavaksi esitetään siksi vain esimerkkinä RDF/XML-kielestä jo edellä esitelty kuvaus *Tuntemattomasta sotilaasta* – tarkemmin RDF/XML-kieleen voi tutustua verkossa.¹

```
<?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 xml:lang="en">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 tämän jälkeen <RDF>-merkkauksen sisällä. RDF-alkumerkkauksen sisällä määritellään nimiavaruudet tagin `xmlns:lyhenne` attribuutteina. RDF-merkkauksen sisällä voidaan kuvailla resurssit <rdf:Description>-merkkauksilla, joiden `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.

¹ <http://www.w3.org/TR/rdf-syntax-grammar/>

- *Geo* on paikkatietoa tuottava maanmittauslaitos.
- *Onto* on asiasanastoja ja ontologioita ylläpitävä kirjastopalvelu.

Näillä organisaatioilla 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 esimerkissä käytetään 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
@prefix dc: <http://purl.org/dc/elements/1.1/> . # Dublin Core
```

Organisaatioilla on tietokannoissaan taulukkomuotoista tietoa, esimerkiksi relaatiotietokantoja, jota ne haluavat julkaista yhdessä linkitettynä datana. 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*, jonka maalasi Akseli Gallen-Kallela Pariisissa vuonna 1891. (Ateneumin taidemuseon kuva.)

Tällainen muunnos ei kuitenkaan tuota hyvin linkitettyä dataa seuraavista syistä:

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 muun muassa kuuluisa hyönteistutkija.
3. *Uusiin resursseihin on voitava viitata ulkopuolelta.* 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 (aliominaisuuksien käyttöön palataan tarkemmin tuonnempana). Näin standardit yhdistävät eri organisaatioiden ominaisuusresurssien tulkinnan ja tekevät ne ymmärrettäviksi koneelle.
2. *Ominaisuuksien arvojen linkitys.* Käytetään ominaisuuksien arvo-

na literaalidatan sijasta resursseja, jotka toiset tahot ovat mahdollisesti määritelleet tarkemmin. Esimerkissä *Bio* julkaisee omassa nimiavaruudessaan biografista tietoa henkilöistä RDF-muodossa. Näistä kokoelmista löytyy muun muassa Akseli Gallen-Kallelan URI-tunniste `b:A_Gallen-Kallela` ja marsalkka Mannerheimin 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 valmiiksi jonkin sarakkeen arvona, 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 .
```

Oletetaan, että biografiakeskukselta on saatavilla taulukon 5.2 mukaista dataa. Se 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 .
```

Paikkatiedon osalta käytössä taas on Geo-maanmittauslaitoksen taulukko, jossa kerrotaan kuntien sijoittumisesta Suomen eri alueille. Taulukosta 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 (tuonnempana tarkemmin esiteltävää) SKOS-standardia käyttäen muun muassa asiasanastoja, joista selviää esimerkiksi, 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. Kenties herrat tunsivat hengenheimolaisuutta keskenään tämän yhteyden vuoksi?

Mikäli eri organisaatioiden käyttämät tunnisteen 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 hyödyntämisestä. Tämä oli kansallisen FinnONTO-projektin keskeisenä tavoitteena.¹ Vaihtoehtona on vaivalloinen eri organisaatioiden datojen yhdistäminen jälkikäteen.

¹ <http://seco.cs.aalto.fi/projects/finnonto/>

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 luetellaan seuraavaksi laajemman yleiskuvan muodostamiseksi RDF-suosituksesta.

5.5.1 Säiliöt

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

säiliö

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

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.

Säiliön jäsenet liitetään säiliöön ominaisuuksilla, joiden muoto on `rdf:_n`, missä $n > 0$ on positiivinen kokonaisluku. Jäsenet voidaan RDF/XML-kielessä ilmaista numeroimatta ominaisuudella `rdf:li`. Seuraavassa esimerkissä 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="http://esim.fi/henkilö/sanasto#Jean"/>
        <rdf:li
          rdf:resource="http://esim.fi/henkilö/sanasto#Aino"/>
        <rdf:li
          rdf:resource="http://esim.fi/henkilö/sanasto#Juhani"/>
        </rdf:li>
      </rdf:Bag>
    </s:henkilö>
  </rdf:Description>
</rdf:RDF>
```


	Luokka	Merkitys
monijoukko	<code>rdf:Bag</code>	<i>Monijoukko</i> eli joukko, jossa voi olla useita samoja alkioita. Alkioihin viitataan ominaisuudella <code>rdf:_n</code> , jossa <i>n</i> on kokonaisluku, mutta monijoukon alkioiden järjestyksellä ei ole merkitystä. Monijoukon tyypiksi merkitään luokka <code>rdf:Bag</code> ominaisuudella <code>rdf:type</code> .
sekvenssi	<code>rdf:Seq</code>	<i>Sekvenssi</i> (sequence) on monijoukko <code>rdf:Bag</code> , jonka alkioiden oletetaan olevan <code>rdf:_n</code> -ominaisuuksien mukaisessa järjestyksessä. Sekvenssin tyypiksi merkitään luokka <code>rdf:Seq</code> ominaisuudella <code>rdf:type</code> .
vaihtoehto	<code>rdf:Alt</code>	<i>Vaihtoehto</i> (alternative) on kuten <code>rdf:Bag</code> , mutta tässä tapauksessa säiliön alkioista yksi kerrallaan ajatellaan valituksi, oletusarvona ensimmäinen alkio (<code>rdf:_1</code>). Vaihtoehto-säiliön tyypiksi merkitään luokka <code>rdf:Alt</code> ominaisuudella <code>rdf:type</code> .

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

kokoelmaluokka

5.5.2 Kokoelmat

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

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

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

Listan alkiot ovat RDF-suosituksen mukaan tyyppiä `rdf:List` (ts. `rdf:List`-luokan yksilöitä), joita ketjutetaan toisiinsa ominaisuuksilla **pää** `rdf:first` (listan ensimmäinen alkio eli *pää* (head)) ja `rdf:rest` (listan loput alkiot eli *häntä* (tail)). Erityinen tyhjä lista `rdf:nil` häntänä päättää listan. Ohessa 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="http://esim.fi/henkilö/sanasto#Jean"/>
      <rdf:Description
        rdf:about="http://esim.fi/henkilö/sanasto#Aino"/>
      <rdf:Description
        rdf:about="http://esim.fi/henkilö/sanasto#Juhani"/>
    </s:henkilö>
  </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, kun lista kirjoitetaan yksinkertaisesti sulkuihin perinteiseen tapaan. Turtle-jäsennin jäsentää tämän automaattisesti verkoksi, jossa listat on esitetty RDF-verkkona ominaisuuksien `rdf:first` ja `rdf:rest` 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 mikäkin tieto on peräisin. Tällainen metatieto voi olla tarpeen muun muassa esitettäessä tietoon liittyviä väitteitä, esimerkiksi koskien sitä, miten varmaa tieto on tai *provenienssitietoa* (provenience data) siitä, mistä tieto on peräisin. Se voi olla hyödyllistä tiedon hallinnan kannalta tai tiedon luotettavuutta arvioitaessa.

Tarkastellaan esimerkkinä kolmikkoa:

reifikaatio

provenienssi

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

:Maa :muoto :litteä .
```

Tavoitteena on kertoa, että tämä tieto joko on totta tai ei ole 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 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.¹ TriG on Turtle-notaation laajennus, jolla kolmikkojoukot voidaan määritellä useampana nimettynä graafina, 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 eli selaimen ohjelmointiin. Ylivoimaisesti käytetyin ohjelmointikieli tähän on JavaScript. JavaScriptin keskeinen rakenteisen tiedon esittämismuoto on

¹ <https://www.w3.org/TR/2014/REC-trig-20140225/>

JavaScript Object System Notation 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*.¹ 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 helpolukuinen.

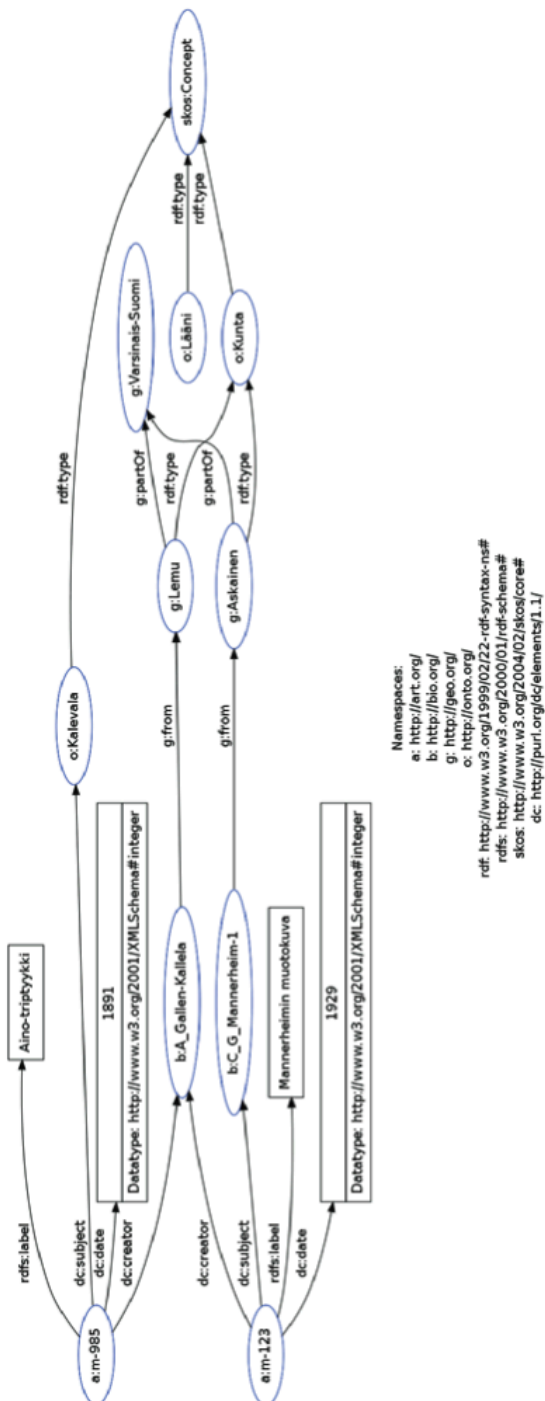
JSON-LD määrittelee joukon ominaisuuksia ja sopimuksia, joiden avulla RDF-rakenteita voidaan esittää JSON-muodossa. Nämä ominaisuudet ovat @-alkuisia. Keskeinen ominaisuus on @context, jonka avulla määritellään JSON-LD-rakenteen tulkintakonteksti, esimerkiksi käytettävissä olevia lyhennysmerkintöjä vähän samaan tapaan kuin nimiavaruuksia käytettäessä. Eri JSON-LD-objektit voivat viitata samaan kontekstiin URL-viittauksen avulla. @type-ominaisuudella taas voidaan kertoa solmun tyyppi. Seuraavassa esimerkissä on 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 kontekstin jälkeen:

¹ <http://json-ld.org/spec/latest/json-ld/>

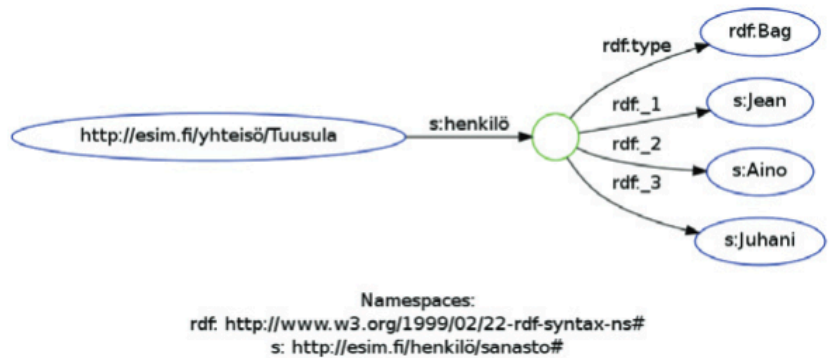
```
{
  "@context": {
    "helka": "http://heldig.fi/kalenteri#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "helka:aika": {
      "@type": "xsd:dateTime"
    }
  },
  "helka:otsikko": "HELDIG Kick-off Symposium",
  "helka:paikka": "Helsingin yliopisto, Pieni juhlasali",
  "helka:aika": "2016-10-05"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

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

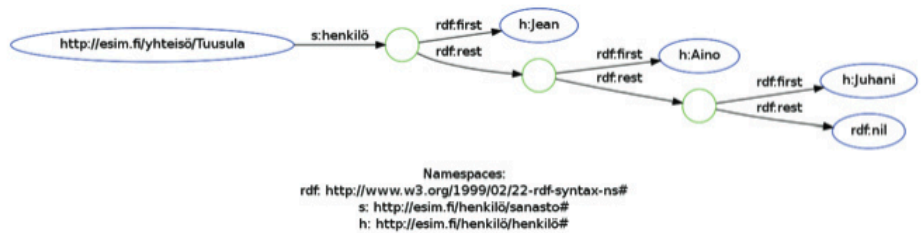
```
_:b0 <http://heldig.fi/kalenteri#aika>
    "2016-10-05"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
_:b0 <http://heldig.fi/kalenteri#otsikko>
    "HELDIG Kick-off Symposium" .
_:b0 <http://heldig.fi/kalenteri#paikka>
    "Helsingin yliopisto, Pieni juhlasali" .
```

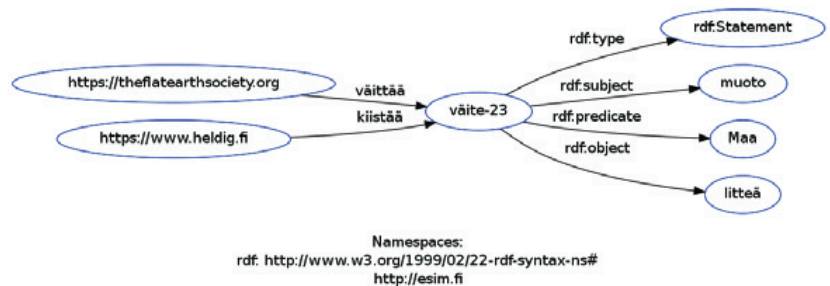
Kuva 5.5. Eri organisaatioiden data yhdistyy automaattisesti laajemmaksi semanttiseksi verkoksi.



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.

6

Tiedon haku ja ylläpito: SPARQL

Joka paljon lukee, se haluaa
joskus kirjoittaakin.

George Crabbe (1754–1832)

Semanttisen webin sovellukset kehitetään usein siten, että käyttöliittymä ja sovelluksen toiminnallisuus toteutetaan selaimessa JavaScriptillä ja erilaisilla siihen perustuvilla kirjastoilla ja kehitysympäristöillä. Selaimen sovellus kysyy 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 hallinnoimissaan HTTP-osoitteissa.

SPARQL on yksi keskeisimmistä semanttisen webin standardeista ja ohjelmoinnin työkaluista. Sen ensimmäinen versio SPARQL 1.0¹ julkaistiin vuoden 2008 tammikuussa. Standardia laajennettiin merkittävästi

¹ <http://www.w3.org/TR/rdf-sparql-query/>

vuoden 2013 maaliskuussa julkaistussa SPARQL 1.1 -versiossa, jonka dokumentaatio koostuu yhdestätoista eri osadokumentista. Kyselyjen muodostaminen ja kielen käyttäminen on kuitenkin varsin suoraviivaista.¹

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 koneelleen. Verkossa on käytettävissä lukuisia julkisia SPARQL-datapalvelupisteitä, joiden dataa voi kysellä ja hyödyntää ohjelmallisesti HTTP-kyselyiden avulla. Esimerkiksi Wikipedian dataa voi kysellä DBpedia

<http://dbpedia.org/sparql>

ja Wikidatan

<http://query.wikidata.org>

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

<http://ldf.fi/>

Sivuilla on kyselyiden testaamista varten 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.² Sen yläreunassa käyttäjä on valinnut käytettäväksi DBpedia SPARQL-datapalvelun <http://dbpedia.org/sparql>. Tämän alle on kirjoitettu kysely SPARQL-kielillä. Oikeassa reunassa olevaa nappia painamalla YASGUI on hakenut DBpediasta kyselyn vastaukset ja näyttää ne taulukkomuodossa käyttöliittymän alaosassa. Kyselyiden

¹ Järjestelmän dokumentaatio ja ohjeistusta löytyy W3C:n verkkosivuilta: <http://www.w3.org/TR/sparql11-query/>.

² Käyttöliittymää voi käyttää osoitteessa <http://yasgui.org/>.

The screenshot shows the YASGUI interface with the following SPARQL query:

```

1 PREFIX : <http://dbpedia.org/resource/>
2 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
4 SELECT ?person ?birth ?death
5 WHERE {
6   ?person dbo:birthPlace :Helsinki .
7   ?person dbo:birthDate ?birth .
8   ?person dbo:deathDate ?death .
9   FILTER (?birth < "1850-01-01"^^xsd:date) .
10 }
11 ORDER BY ?birth

```

Below the query editor, the results are displayed in a table view. The table has columns for 'person', 'birth', and 'death'. It shows 5 results, all of which are truncated with 'read more' links.

person	birth	death
1 :Christina_Abrahamsdotter	"1432"^^xsd:gYear	"1492"^^xsd:gYear
2 :Mikael_Agricola	"1510"^^xsd:gYear	"1557-04-09"^^xsd:date
3 :Klaus_Fleming	"1535"^^xsd:gYear	"1597-04-13"^^xsd:date
4 :Gustav_Evertsson_Horn	"1614-05-26"^^xsd:date	"1666-02-27"^^xsd:date
5 :Gustaf_Bonde_(1620-1667)	"1620-02-01"^^xsd:date	"1667-05-25"^^xsd:date

Kuva 6.1. YASGUI-käyttöliittymä SPARQL-kyselyille. Haun kohteena ovat ennen vuotta 1850 Helsingissä syntyneet henkilöt DBpediassa. (<http://yasgui.org>.)

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 palvelu on käytettävissä verkon kautta. Esimerkiksi Linked Data Finland -palvelussa käytetty Apache Jena Fuseki RDF-tietokanta ja -palvelu voidaan ottaa käyttöön seuraavalla tavalla:¹

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 Windowsissa voit kaksoisklikata tiedostoa `fuseki-server.bat`). Fuseki on Java-pohjainen ja toimii samalla tavalla eri käyttöjärjestelmissä.
3. Avaa selaimessasi seuraava paikallisosoite, jota äsken käynnistämäsi palvelin kuuntelee:

`http://localhost:3030`

4. Avautuvan käyttöliittymän linkkien ja painikkeiden avulla voit la-

¹ <https://jena.apache.org/documentation/fuseki2/>

data järjestelmään RDF-tiedostoja ja luoda graafeja.

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*. Yhtä näistä kutsutaan *oletusgraafiksi* (default graph). Palvelupisteeseen kohdistetut kyselyt ja operaatiot kohdistuvat aina oletusgraafiin, ellei muuta graafia ole erikseen mainittu. Oletusgraafiksi voidaan palvelupisteessä spesifioida paitsi jokin tietty graafi myös niiden unioni, jolloin kyselyt kohdistuvat näppärästi oletuksena kaikkeen palvelupisteen dataan.

SPARQL-kieleen kuuluvat välineet sekä

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

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

6.2 Peruskyselyt

SPARQL-kielen ydin koostuu neljästä eri kyselytyypistä:

1. SELECT-kyselyllä voidaan hakea vastauksia kyselyihin. Vastauksena palautetaan arvosijoituksia 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. Seuraavaksi tarkastellaan kyselyiden perustyyppisiä ja 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 graafihahmo graafissa olevaan dataan. Vastauksena saadaan kaikki mahdolliset yhdistelmät hahmon muuttujien arvoja, joilla 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. (YASGUI-työkalu.)

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 URI (käytettävä oletusgraafi)

# Kyselyn graafihahmon muotoilu
WHERE {
  kolmikot, vaihtoehdot (FILTER, OPTIONAL ja UNION)-rajoitteet
}

# Vastauksen tulosten muotoilukomennot
järjestys (ORDER BY), määrä (LIMIT) ja valinta (OFFSET)
```

Kanta-URI:n 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 vastaukseen 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 seitsemään tulokseen (LIMIT).

```
# DBpedian resurssit ja ontologia  
PREFIX : <http://dbpedia.org/resource/>  
PREFIX dbo: <http://dbpedia.org/ontology/>  
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 määrittelyä voidaan monipuolistaa FILTER-, OPTIONAL- ja 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

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

Taulukko 6.1. SPARQL-muodoissa käytettävissä olevia funktioita.

ratkaisuja; vain sellaiset muuttujasijoitukset jäävät jäljelle, joilla suodatimen ehtolauseke palauttaa arvon tosi. Esimerkiksi seuraavassa haetaan DBpedia:sta vain sellaisia syntyperäisiä helsinkiläisiä, jotka ovat syntyneet 1800-luvulla tai aiemmin.

```

PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
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 muun muassa taulukossa 6.1 lueteltuja loogisia, matemaattisia, vertailu- ja muita funktioita. Lukufunktioiden avulla voidaan tehdä datamuunnoksia. Jos esi-

merkiksi 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-nimisiä syntyperäisiä helsinkiläisiä säännöllisellä lausekkeella seuraavassa kyselyssä. Säännöllisen lausekkeen `^Matti` merkki `"^"` tarkoittaa merkijonon alkua, eli tässä haetaan vain Matti-alkuisia nimiä.¹

```
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 rauhan-sopimusta Helsingissä syntyneet:

```
PREFIX :      <http://dbpedia.org/resource/>
PREFIX dbo:  <http://dbpedia.org/ontology/>
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) # Parii-
sin rauhansopimus
}
ORDER BY DESC (?syntyi) # Järjestä myöhäisin ensin
```

Tällöin vastauksessa ei kaikkien henkilöiden osalta ole tietoa kuolinajasta.

¹ Säännölliset lausekkeet ovat erittäin käyttökelpoisia SPARQL-kyselyitä muotoiltaessa. Niihin voi tutustua verkossa esimerkiksi osoitteessa http://www.w3schools.com/jsref/jsref_obj_regexp.asp.

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/>
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 sen selvittäminen, 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 Nordenskiöld-niminen henkilö, selviää seuraavalla kyselyllä:

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

Eräs tyypillinen tiedon tarve on selvittää, mitä ominaisuuksia jollain tietyllä resurssilla on. Tähän on käytettävissä DESCRIBE-kysely, jolla asia selviää yksinkertaisemmin 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-datapalveluun¹ palauttaa Mika Waltarin *Sinuhe egyptiläiseen* (1945) 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}
```

6.2.4 Uuden RDF-verkon luominen: CONSTRUCT

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

```
CONSTRUCT graafihahmo-1
[FROM graafi-URI]
WHERE graafihahmo-2
```

Siinä sovitetaan ensin *graafihahmo-2* FROM-osoitteessa mainittuun graafiin tai oletusgraafiin, jos sellaista ei ole määritetty. Tuloksena olevat

¹ Yleisten kirjastojen ylläpitämä Kirjasammon 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 (Mäkelä ym., 2011a). Järjestelmä kehitettiin alun perin osana FinnONTO-tutkimushanketta, ja se sisältää nykyään alkuperäisen kaunokirjallisuuden ohella mm. tietokirjallisuutta.

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. Seuraavaksi esitellään näistä yhteenvetona keskeisimpiä; kattavat ohjeet kyselyiden muodostamiseen löytyvät kyse-lykielen dokumentaatiosta W3C-järjestön sivuilta.¹

6.3.1 Ominaisuuspolut

ominaisuuspolku *Ominaisuuspolulla* (property path) tarkoitetaan kahden solmun välis-
tā ominaisuuksien ketjua graafissa. Yksinkertaisimmillaan ketjun muo-
dostaa yksi ominaisuus, jolloin polun pituus on 1. Useammasta omi-
naisuudesta muodostuvia ominaisuuspolkuja voidaan kuvata SPARQL-
polkulauseke kyselyissä joustavasti *polkulausekkeilla*. Niitä voidaan muodostaa taulu-
kon 6.2 operaattoreiden avulla.

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)` DBpedia:ssa:

¹ <https://www.w3.org/TR/sparql11-query/>

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
$!x$	ominaisuuspolkua x ei ole

Taulukko 6.2. Polkulausekkeiden määrittelymuodot.

```

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 arvosijoitus etukäteen tai laskemalla arvoja kyselyn suorituksen aikana.

Muuttujille voidaan kyselyssä asettaa alkuarvoja muodolla

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

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

$\{(a_{11} \ a_{12} \ \dots a_{1n}) \ (a_{21} \ a_{22} \ \dots a_{2n}) \ \dots\}$

Esimerkiksi seuraava 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 SELECT-kyselyyn sisään. Esimerkiksi seuraava kysely hakee DBpediasta Helsingissä syntyneet, jo kuolleet henkilöt¹ 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 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.

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 vastaus-

¹ Tätä kirjoitettaessa 758 eri henkilöä.

joukkoja ja annetaan vasta analyysin tulokset vastauksena, kutsutaan *aggregaattikyselyiksi* (aggregate query).

aggregaattikysely

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

Aggregaattikyselyssä rajataan ensin ne muuttujat, joiden arvojen muodostamaa kombinaatiojoukkoa (aggregaattia) analysoidaan. 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 muuttujalla on kussakin 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 päivämäärä on 11.1.1732, mutta tietämyskannan täydentyessä arvo voi muuttua:

```
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) sovelle-

taan muuttujan ?syntyi arvoihin vastauksessa, eli tässä tapauksessa yhteen joukkoon. Jos käytettäisiin esimerkiksi muotoa `GROUP BY ?syntyi`, 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 olisivat silloin eri syntymäajat (mikä ei olisi kovin mielenkiintoista).

Jos halutaan löytää varhaisimman syntymäajan ohella myös silloin syntynyt henkilö, tässä suomalais-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 kunkin maan pääkaupungissa syntyneiden henkilöiden varhaisimmat syntymäajat. Nyt tulosjoukkoina ovat kussakin pääkaupungissa syntyneet henkilöt, joiden syntymäaikojen minimi koostetaan maittain vastaukseen varhaisimmasta myöhäisimpään:

```
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
```

Taulukkoon 6.3 on koottu `MIN`-funktion ohella käytettävissä olevat aggregaattifunktiot, joita sovelletaan aggregoinnin aikana luotuihin vastausjoukkoihin. Vastausjoukkojen generointi kyselyn hahmosta määritellään `GROUP BY` -muodolla; jos muotoa ei ole, oletetaan vastausjoukossa olevan vain yksi joukko, alkuperäinen vastaus. Aggregointifunktioita so-

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. Aggregointifunktiot, joita sovelletaan erikseen jokaiseen aggregoituun vastausjoukkoon muuttujan x suhteen.

velletaan sitten jokaiseen vastausjoukkoon 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 DBpediaan mukaan on merkitty puhuttavan eniten eri kieliä, ja luettelee kielet ja niiden lukumäärät maittain englanniksi pilkulla ja välilyönnillä erotettuina listoina kuvan 6.3 mukaisesti. Tämän perusteella ei kuitenkaan voida tietää, missä valtioissa todellisuudessa puhutaan eniten eri kieliä. Esimerkiksi kaikkia Kiinassa puhuttavia 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	$\approx 20^{xsd.integer}$
2	http://dbpedia.org/resource/East_Timor	Tetum language, Kemak language, Portuguese language, Bekais language, Habun language, Bunak language, Galoli language, Kawai mina languages, Makasae language, Mambai, Uab Meto language, Makalero dialect, Fataluku language, Idalaka language, Makuv'a language, Tokodede, Atauru language	$\approx 17^{xsd.integer}$
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	$\approx 16^{xsd.integer}$

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

6.3.4 Alikyselyt

SPARQL 1.1 mahdollistaa SELECT-kyselyiden käyttämisen osana toista kyselyä. Tällöin graafihahmossa olevan kolmikον paikalle upotetaan *alíkysely* kaarisulkujen sisällä. Kyselyt suoritetaan sisältä ulos siten, että ensin suoritetaan sisin kysely. Sen vastauksessa olevat muuttujien arvosi-
 joitukset sijoitetaan tämän jälkeen ulompaan kyselyyn yksi toisensa jälkeen, ja vastausten unionista muodostetaan ulomman kyselyn vastaus. Jos tämäkin on alíkysely, prosessi jatkuu, kunnes lopulta saadaan laskettua kaikkein uloimman kyselyn vastaus.

6.3.5 Federoitu kysely

Edellä on tehty kyselyjä vain yhteen SPARQL-palvelupisteeseen. Toisinaan 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-kielissä on kuitenkin myös mahdollista yhdistää vastaukset yhden kyselyn sisällä

niin sanottuna *federoituna kyselynä* (federated query). Federoitu kysely ilmaistaan upottamalla kyselyn graafihahmon kolmikoiden sekaan SERVICE-muoto:

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

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

federoitu kysely

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.
2. *Graafien päivityskomennoilla* voidaan lisätä ja poistaa kolmikoita yhdessä graafissa.

graafien hallinta

graafien päivitys

Keskeisiä graafien hallinnan komentoja ovat seuraavat:

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, jolloin 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 ni-

mettyyn graafiin teokset.

Graafien hallinnan komentojen tarkempi kuvailu ja esimerkkejä käytöstä löytyy W3C:n SPARQL Update -dokumentaatiosta.¹

6.5 Graafin datan päivittäminen

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

- INSERT DATA-komennolla voidaan lisätä kolmikoita palvelupisteeseen.
- 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, jonka tunniste on 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 eksplisiittisesti lueteltuja kolmikoita. INSERT- ja DELETE-komennoilla lisäykset ja muutokset voidaan tehdä muuttujia hyödyntäen graafihahmon kuvaamille kolmikoille, jotka on kuvattu WHERE-muodolla vastaavaan tapaan kuin SELECT-kyselyssä. Erillistä kolmikoiden päivityskomentoa ei ole, vaan muutokset tehdään poistamalla ja lisäämällä kolmikoita. Seuraavassa esimerkissä muutetaan RDF-dokumentin kaikkien Masa-nimisten henkilöiden etunimi viralliseen muotoon Matti:

¹ <https://www.w3.org/TR/sparql11-update/>

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

```
DELETE { ?henkilö foaf:givenName 'Masa' }
```

```
INSERT { ?henkilö foaf:givenName 'Matti' }
```

```
WHERE { ?henkilö foaf:givenName 'Masa' }
```


7

Linkitetyn datan julkaiseminen

Jospa olemmekin olemassa
vain toistemme muistissa.

Stanisław Jerzy Lec
(1909–1966)

Tässä luvussa esitetään linkitetyn datan julkaisuperiaatteet ja tähtiluokitus verkossa sekä kerrotaan, miten HTTP-URI-tunnisteista voidaan saada lisää tietoa. Semanttista dataa voidaan julkaista paitsi SPARQL-palvelupisteissä myös verkkosivuille upotettuina merkkauksina muun muassa hakukoneita varten.

7.1 Linkitetyn datan neljä periaatetta

Linkitetyn datan julkaisemisessa verkossa pyritään noudattamaan Tim Berners-Leen esittämää neljää *linkitetyn datan periaatetta* (linked data principle).¹

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 stan-

¹ <https://www.w3.org/DesignIssues/LinkedData.html>

dardien 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 yksilöivinä 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-linkkeillä. 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 sovittava 1) käytännöistä erilaisten tunnisteiden esittämisestä ja 2) mekanismista, jolla erilaisiin HTTP-osoitteisiin verkossa vastataan. Jos esimerkiksi selaimen kirjoitetaan URI-osoite, 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 on esitelty, 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     # Helsingin RDF-data
http://dbpedia.org/page/Helsinki     # Em. datan WWW-sivu
```

Jos webin *aluenimi* (domain) ottaa käyttöön WWW-sivuosoitteiden ohella dataosoitteita, kuten edellä, pitää sen palvelimen osata ratkaista, palautetaanko HTTP-kyselyyn jokin verkkosivu vai dataa. Tätä valintaa kutsutaan osoitteiden *dereferoimiseksi* (dereference) eli *ratkomiseksi*. Osoitteiden analysointi voidaan tehdä edustapalvelimella, joka tutkii saamansa HTTP-kutsut ja reagoi niihin palveluntarjoajan haluamalla tavalla.

ratkominen

Verkko toimii siten, että kun sille (ts. jollekin verkon palvelimelle) lähe-

otsikko (HTTP)

tetään HTTP-kutsu joltain sovellukselta, esimerkiksi verkkoon kytketyn tietokoneen selaimelta, kutsuun sisältyy HTTP-osoite, *otsikkotieto* (header), ja varsinainen *kutsun sisältö* (message body), joka voi olla esimerkiksi SPARQL-kysely. Otsikkotiedolla voidaan asettaa pyyntöjä kutsun käsittelyä varten. Siinä voidaan muun muassa kertoa haluttu kyselyn vastauksen formaatti, kuten HTML, RDF tai JSON. Esimerkiksi seuraavan HTTP-kutsun otsikkotiedot kertovat, että vastauksen pitäisi olla HTML-muotoista tekstiä UTF-8-merkistöllä koodattuna.

```
...
Accept: text/html
Accept-Charset: utf-8
...
```

Osoite otsikkotietoineen lähetetään ensin sille verkon palvelimelle, johon päätelaitteen sovellus (esimerkiksi selain) on kytketty. Tämän jälkeen ratkaistaan HTTP-kutsun osoitteen ja verkon nimipalvelimien muodostaman verkoston avulla se, mikä verkon palvelin kutsun käsittelee ja se, missä se fyysisesti sijaitsee. Sitten kutsu reititetään tämän palvelimen vastattavaksi. Internetin 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, luoda näiden avulla uusia HTTP-URI-skeeman mukaisia tunnuksia ja päättää siitä, miten vastaviin 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 URL-osoitteet ohjautuvat tätä kirjoitettaessa Aalto-yliopiston SeCo-tutkimusryhmän erälle edustapalvelimelle, johon asennettu Varnish-ohjelmisto tutkii HTTP-osoitteiden muodon ja niihin liittyvät otsikkotiedot ja ohjaa kutsut tämän perusteella edelleen toisten sovelluspalvelimien vastattaviksi.

LDF.fi-palvelu sisältää kymmeniä nimettyjä *datapalveluja* (service). Esimerkiksi Sotasampo.fi-sovellus¹ (WarSampo) perustuu LDF.fi:n palveluun nimeltä *warsa* (WarSampo). Jokainen palvelu voi sisältää joukon

¹ <http://www.sotasampo.fi>

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 noin 160 000 valokuvan metatiedot ovat *photographs*-graafissa

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

ja yksittäisten kuvien URI:t on muodostettu tämän aliosoitteina. Seuraavassa 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 liittyvä data luetaan muodolla

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

esimerkiksi näin:

`http://ldf.fi/warsa/data?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

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

sisältöneuvottelu

Yleisesti URI-tunnisteiden muodostamisessa on käytössä kaksi eri tapaa:¹

¹ Ks. tarkemmin ”Cool URIs for the Semantic Web”: <https://www.w3.org/TR/cooluris/>.

1. Hash URI:t
2. 303 URI:t

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 edellä oleva käsite `Class` on määritelty kirjoittamalla edellä mainittua hash-merkkiä käyttävä osoite selaimeen, saa palvelin ainoastaan tunnisteen alun `http://www.w3.org/2000/01/rdf-schema` tutkitavakseen. 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 URI:en käytön etuna on yksinkertainen ja suoraviivainen toimintamalli, jossa palautetaan kaikki nimiavaruuden määritelmät sisältävä RDF-tiedosto. Malli on kuitenkin haasteellinen, jos tiedosto on iso. Esimerkiksi DBpediassa hash URI:en 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, haluaako asiakasjärjestelmä vastaukseksi WWW-sivun vai dataa. Vastaus välitetään asiakasohjelmalla HTTP 303 -vastauksena (status code 303), joka tarkoittaa kyselyn uudelleenohjaimista 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 Mannerheimin valokuvaan

liittyvä URI

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

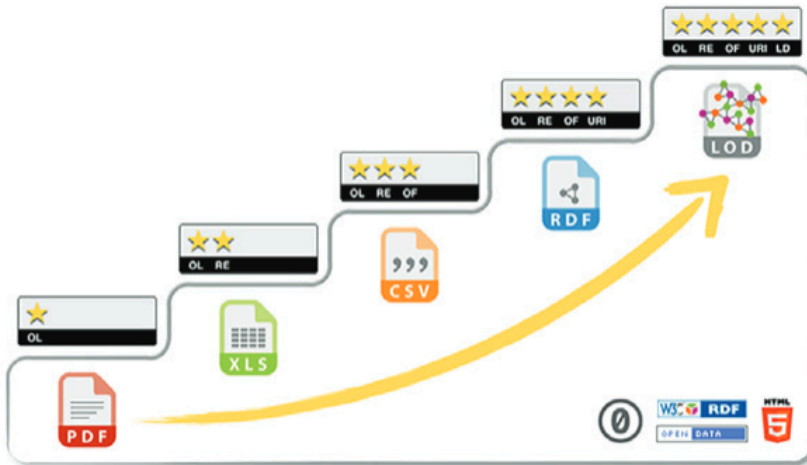
palauttaa RDF-tietoa Turtle-muodossa, jos kutsun otsikossa on tieto

```
Accept: text/turtle
```

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

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

7.4 Datajulkaisujen tähtiluokitus



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

Tim Berners-Lee on kehittänyt avoimen datan palveluiden arviointia varten kuvassa 7.1 esitetyn tähtiluokituksen, jonka tavoitteena on kannustaa

julkaisijoita julkaisemaan dataa verkossa mahdollisimman käyttökelpoisessa ja avoimessa muodossa:

★

Ensimmäisen tähden saa, jos ylipäättään avaa dataa verkossa, vaikkapa PDF-dokumenttina.

★★

Toinen tähti tulee siitä, että data on julkaistu rakenteisessa muodossa, esimerkiksi Excel-tilukkona 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 tilukkomuotoinen data on parempi julkaista standardoidussa avoimessa CSV-muodossa kuin Microsoftin Excel-formaatissa.

★★★★

Neljännän 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 sisäisesti 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 *mikroformaattien* (microformat) avulla, jotka ovat eri sovellusalueille kehitettyjä, yhteisesti sovittuja tapoja esittää tietoa HTML-sivuilla koneiden käytettäväksi.

Yksi esimerkki mikroformaatista on hRecipe, jonka avulla esitetään resepteihin liittyvää tietoa hakukoneita varten. Kuvassa 7.2 on esimerk-

mikroformaatti

Classic Macaroni Salad
See how to make a classic deli-style pasta salad. »

1 of 22 Photos

Prep Time: 15 Min | Cook Time: 15 Min | Ready In: 30 Min

Servings (Help)

Original Recipe Yield 6 servings

Ingredients

- 1 pound uncooked pasta
- 1 tablespoon olive oil
- 2 cloves garlic, crushed
- 1/2 cup chopped red onion
- 1 pound skinless, boneless chicken breast meat - cut into bite-size pieces
- 1 (14 ounce) can marinated artichoke hearts, drained and chopped
- 1 large tomato, chopped
- 1/2 cup crumbled feta cheese
- 3 tablespoons chopped fresh parsley
- 2 tablespoons lemon juice
- 2 teaspoons dried oregano
- salt and pepper to taste
- 2 lemons, wedged, for garnish

Directions

More Recipes Like This

- Greek Chicken and Chicken
- Marinated Greek Chicken Kabobs
- Greek Pasta with Tomatoes and White Beans
- Mediterranean Chicken Salad
- Greek Veggie Salad

kitchenapproved

- Add to Recipe Box
- Add to Shopping List
- Print this Recipe
- Share/Email
- supportingmembers
- Create Menu
- Customize Recipe
- Kitchen-friendly View

Kuva 7.2. Reseptisivu verkossa. (<http://allrecipes.com>.)

ki WWW-sivulla olevasta reseptistä, jossa käytetään ympyröidyn listan ainesosia. Listaa vastaavassa HTML-kielisessä kuvauksessa (kuva 7.3) on kerrottu, että listassa on ainesosia (`class="ingredients"`), 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 täsmähaakuun, jossa hakua voi tarkentaa ainesosien avulla. Esimerkiksi kuvassa 7.4 käyttäjä on valinnut hakuun reseptit, joissa käytetään sitruunaa.

Idea mikroformaateista on otettu käyttöön HTML5-standardissa¹ uudella nimellä ja uusilla, kehittyneemmällä esitysmuodoilla *mikrodata* (microdata).² Semanttisen webin puolella sama ajatus metadatan upottamisesta WWW-sivuille on synnyttänyt RDFa-spesifikaation, joka

mikrodata
RDFa

¹ Suositus julkaistiin vuonna 2014 osoitteessa <https://www.w3.org/TR/html5/>.

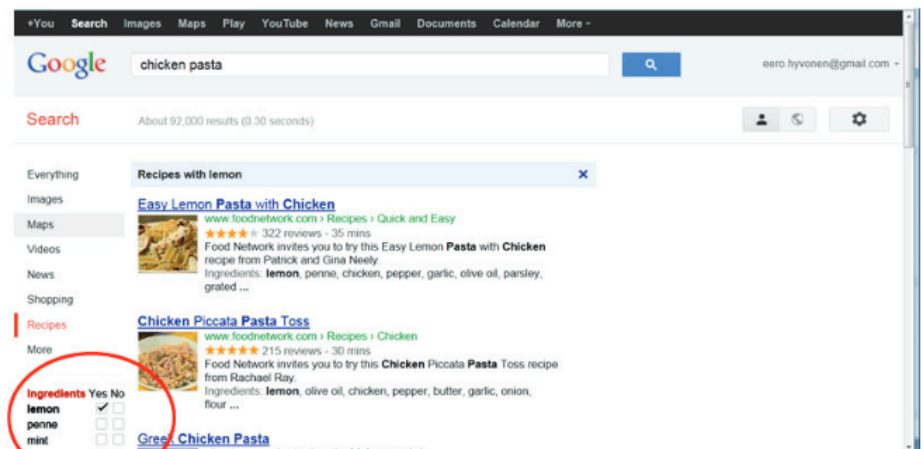
² <http://www.w3.org/TR/microdata/>

```

1279 <div class="ingredients" style="margin-top: 10px;">
1280 <h3>
1281   Ingredients</h3>
1282
1283   <ul>
1284
1285     <li class="plaincharacterwrap ingredient">
1286       1 pound uncooked pasta</li>
1287
1288     <li class="plaincharacterwrap ingredient">
1289       1 tablespoon olive oil</li>
1290
1291     <li class="plaincharacterwrap ingredient">
1292       2 cloves garlic, crushed</li>
1293
1294     <li class="plaincharacterwrap ingredient">
1295       1/2 cup chopped red onion</li>
1296
1297     <li class="plaincharacterwrap ingredient">
1298       1 pound skinless, boneless chicken breast meat - cut into bite-size pieces</li>
1299
1300     <li class="plaincharacterwrap ingredient">
1301       1 (14 ounce) can marinated artichoke hearts, drained and chopped</li>
1302
1303     <li class="plaincharacterwrap ingredient">
1304       1 large tomato, chopped</li>
1305
1306     <li class="plaincharacterwrap ingredient">
1307       1/2 cup crumbled feta cheese</li>
1308
1309     <li class="plaincharacterwrap ingredient">
1310       3 tablespoons chopped fresh parsley</li>
1311
1312     <li class="plaincharacterwrap ingredient">
1313       2 tablespoons lemon juice</li>
1314
1315

```

Kuva 7.3. Merkkaus hRecipe-mikroformaatin avulla kuvan 7.2 reseptisivulla. (<http://allrecipes.com>.)



Kuva 7.4. Google tukee reseptien hakua hRecipe-merkkausten avulla. (<http://google.com>.)

määrittelee tavan upottaa RDF-muotoista dataa verkkosivuille.¹ Verkkosivuille upotettujen merkkauksen irrottamiseen HTML/XML-koodista on spesifioitu GRDDL-mekanismi.² Yhä suosittumaksi metadatan upotusformaatiksi on muodostumassa JSON-LD, jolloin metadatan merkkaukset 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 ilmiäisessä ei kuitenkaan ole mahdollista.

Erityisen merkittäväksi mikrodatan formaatiksi on muodostunut isojen hakukoneyhtiöiden Google, Microsoft, Yahoo ja Yandex vuonna 2011 lanseeraama Schema.org.³ Sen määrittelyjen avulla voidaan verkkosivuille, sähköpostiviesteihin ja muihin aineistoihin upottaa 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 muun muassa sivujen indeksoinnin 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 rikkaamassa muodossa ja esittää WWW-sivut ja entiteetit strukturoidusti lisätiedoilla varustettuina.

Schema.org

Metadatan merkkaukset ovat tärkeitä hakukoneiden ohella myös sosiaalisen median sovelluksille, kuten Facebook, Twitter ja Instagram, jotka ovat lanseeranneet asiakkaittensa käytettäväksi omia tapojaan sisältöjen merkkaukseksi, verkkosivujen näyttämiseksi ja näkyvyyden parantamiseksi omissa sovelluksissaan. Facebookin käyttämä Open Graph -järjestelmä⁴ perustuu RDFa-standardiin.

¹ <http://www.w3.org/TR/xhtml1-rdfa-primer/>

² <https://www.w3.org/TR/grddl/>

³ <http://schema.org>

⁴ <https://developers.facebook.com/docs/sharing/opengraph>

III

TIETÄMYKSEN ESITTÄMINEN

Metadata ja ontologiat

Käsitteiden määrittely on
viisauden alku.

Sokrates (n. 470–399 eaa.)

Semanttisen webin tietosisällöt ovat konetulkittavia kuvauksia reaali-maailman ilmiöistä, asioista tai tiedoista. Esimerkiksi kirjaston kokoelman teokset voidaan esittää yksilöllisinä (meta)data-alkioina (resursseina), joita kuvaavia ominaisuuksia ovat teoksen kirjoittaja, julkaisija, julkaisuvuosi, painopaikka ja niin edelleen. Ominaisuuksien arvot voivat olla joko literaalidataa, kuten merkkijonoja, lukuja tai päiväyksiä, tai viittauksia toisiin käsitteisiin.

Tietosisältöjen kuvaukset perustuvat kahdenlaisiin määrittelyihin:

1. *Metadatatamalli* (metadata model) eli *metadataskeema* (metadata schema) kertoo, minkälaisilla ominaisuuksilla erityyppisiä yksilö-metadataskeema
käsitteitä kuvaillaan.
2. *Aiheontologiat* (domain ontology) määrittelevät ominaisuuksienaiheontologia
arvoina käytettävät käsitteet, kuten esineet, paikat, henkilöt, tapahtumat ja ajat.

Seuraavaksi tarkastellaan ensin metadatan käsitettä ja metadatan erilaisia muotoja. Tämän jälkeen esitellään tarkemmin kaksi tärkeää, laajemmassa käytössä olevaa yleiskäyttöistä metadatatamallia: verkkodokumenttien esittämiseen tarkoitettu Dublin Core ja kulttuurialan sisältöjen harmonisoinnissa käytetty CIDOC CRM. Lopuksi määritellään, mitä

”ontologialla” tietotekniikassa tarkoitetaan.

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 kone-tulkittavaa tietoa:¹

Metadata is machine understandable information for the Web.

tietämyksen
esittäminen

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

Perinteisesti metadatatä ovat tuottaneet ammattilaiset luetteloidessaan erilaisia kokoelmia.³ Webissä metadatatä tuottavat myös harrastajat julkaistessaan vaikkapa ottamiaan valokuvia tai Twitter-viestejä verkossa. Nykyisin metadatatä tuotetaan yhä enemmän automaattisesti muun muassa tiedonlouhinnan keinoin.

Webin sisältökohteisiin liittyvää metadatatä voidaan luokitella muun muassa seuraavalla tavalla:⁴

1. **Hallinnollinen metadata**, jota käytetään kokoelmien hallinnassa: esimerkiksi kohteen hankintaan ja säilytyspaikkaan liittyvä tieto.

¹ <http://www.w3.org/Metadata/>

² Tietämyksen esittämiseen voi tutustua tarkemmin esim. teoksissa (Sowa, 2000; Brachman & Levesque, 2004).

³ Hyvä johdatus metadatan maailmaan on esim. (Baca, 2008) ja (Gartner, 2016).

⁴ Ks. (Gilliland, 2008).

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 suositelujärjestelmissä sisältöjen hakemiseen ja linkittämiseen.
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 kirjallisista teoksista voidaan esittää metatietoina sen kirjoittaja, julkaisija, julkaisuaika, hyllyluokitus ja sisältöä kuvaavia asiasanoja.

metadataskeema

Metadatatalleja 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.¹
- **Verkkotiedon skeemat**, joilla voidaan kuvata webissä julkaistavia dokumentteja. Näistä tunnetuin on Dublin Core.²
- **Harmonisointiskeemat**, joilla voi yhtenäistää eri skeemojen avulla luetteloituja tai kuvailtuja tietoja. Tähän kategoriaan voidaan lukea kirjastojen Functional Requirements for Bibliographic Records (FRBR) -käsittemalliperhe, jota kehittää IFLA (International Federation of Library Associations and Institutions).³ Museoalalla taas on käytössä ISO-standardi CIDOC CRM, ontologia, jota kehittää ICOM-järjestön (International Council of

¹ <http://collectiontrust.org.uk/spectrum/>

² <http://dublincore.org/>

³ <https://www.ifla.org/node/2016>

Museums) työryhmä.¹

- **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), jossa tietoa kysellään HTTP-protokollalla tietyillä kyselymuodoilla ja metadata siirretään XML-muodossa.²

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 aikamääre, ja syntaktisia vaatimuksia tiedon esittämislle, esimerkiksi että aikamääreet esitetään ISO 8601 -standardin mukaisesti. Lisäksi ominaisuuden arvojen lukumäärälle ja ominaisuuden pakollisuudelle tai valinnaisuudelle voidaan asettaa *kardinaliteettirajoite* (cardinality constraint). Metadataskeeman käyttämisestä voidaan antaa erillisiä *luettelointiohjeita* (cataloging rules).

Seuraavaksi kuvataan tarkemmin verkkotiedon metadatatamalleja esimerkkinä Dublin Core ja tämän jälkeen metatiedon harmonisointia CI-DOC CRM -mallin avulla.

8.2 Dublin Core -malli

Dublin³ 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 standardoitua⁴ ominaisuutta eli *elementtiä* (element): DC Metadata Element Set⁵ (DCES) (ks. taulukko 8.1).

Kaikkien DC-elementtien kardinaliteetti on $0..n$, eli ne ovat valinnaisia ja niillä voi olla useita arvoja. Esimerkiksi kirjalla voi olla useita kirjoittajia (*creator*). Elementit esitetään taulukon mukaisessa järjestyksessä *title*,

¹ <http://www.cidoc-crm.org/>

² <https://www.openarchives.org/pmh/>

³ Dublin viittaa USA:n Ohiossa sijaitsevaan Dublinin kaupunkiin, jossa sijaitsee kirjastoalalla keskeisen vaikuttajaorganisaation OCLC (Online Computer Library Center) pääkonttori.

⁴ NISO Standard Z39.85-2001 ja ISO Standard 15836-2003

⁵ <http://dublincore.org/documents/dces/>

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

creator, *subject* ja niin edelleen.

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

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 muun muassa siitä, miten DC-sovelluksia voidaan määritellä.

Esimerkkinä DC:n soveltamisesta tarkastellaan TerveSuomi-järjestelmää, jonka pilottiversio kehitettiin osana kansallista FinnONTO-hanketta. Järjestelmän taustalla oli havainto siitä, että terveyden edistämiseen liittyviä ohjeita ja aineistoja tuottivat Suomessa Terveystieteen ja hyvinvoinnin laitoksen lisäksi muun muassa yli sata terveysjärjestöä.

Taulukko 8.2. DCMI Metadata Terms.

abstract	accessRights	accrualMethod	accrualPeriodicity
accrualPolicy	alternative	audience	available
bibliographicCitation	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		

TerveSuomi-järjestelmän ideana oli luoda maahamme hajautetun tiedon tuotannon malli ja keskitetty semanttinen portaali, johon eri toimijoiden aineistot voitaisiin koota ja rikastaa niitä toistensa avulla sisällöllisesti dataa linkittämällä. Näin voitaisiin samalla poistaa turhaa päällekkäistä sisällöntuotantoa ja parantaa terveysjärjestöjen keskinäistä työnjakoa.¹

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.² Nimiavaruus dc viittaa Dublin Coreen ja dct Dublin Core Terms -määrittelyyn, joita on laajennettu erällä 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 muun muassa se, et-

¹ TerveSuomi-järjestelmä (HealthFinland) (Suominen ym., 2009a) sai kansainvälisen Semantic Web Challenge -teknologiapalkinnon vuonna 2008 Saksassa, ja siitä tuotteistettiin yksi Terveysten ja hyvinvoinnin laitoksen kansallisista portaaleista.

² Malli on kuvattu tarkemmin dokumentissa (Suominen ym., 2009b).

tä 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 ”päättely” on esitetty asiasanaviittauksilla Yleiseen suomalaiseen ontologiaan YSO¹ (nimiavaus `yso`). Kun järjestelmään kuuluvien esimerkin kaltaisten verkkosivujen ja dokumenttien metatiedot esitettiin samaa metadatatamallia 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 suosittelujärjestelmä aineistojen etsimistä ja selailua varten.

8.3 CIDOC CRM: datan 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 ovat luonteeltaan pikemminkin tietoa tapahtumista kuin dokumenteista tai objekteista, jotka toki kuuluvat myös kuvattavaan reaali maailmaan.

Reaali maailmaan liittyvän (meta)tiedon esittämistä varten on kehitetty *tapahtumaperustaisia* (event-based) metatietomalleja, joista tun-

tapahtumaperustainen

¹ YSO-ontologia (Seppälä & Hyvönen, 2014) on kehitetty Kansalliskirjaston Yleisestä suomalaisesta asiasanastosta YSA, ja se muodostaa edellä esitellyn laajemman yhdistelmäontologian KOKO keskeisimmät käsitteet.

	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)
	Muokausaika	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
luokitteut	Tallennuskielto	ts:noindex	0..1	Boolean	“true”, “false”, “1” tai “0”
	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
sisällönkuvaus	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

^aErikieliset versiot sallittu, kunhan kullakin kielellä on annettu vain yksi arvo.

^bPakollinen, jos arvo voi olla muuta kuin Teksti.

Kuva 8.1. TerveSuomi-järjestelmän metadataskeema (Suominen ym., 2009b). 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.

netuin on museoalan CIDOC CRM -standardi.¹ 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 toimijoina ovat 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 da-

¹ Standardin perusteita on esitelty esim. artikkeleissa (Doerr, 2003, 2009) ja hankkeen kotisivulla <http://cidoc-crm.org>.

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.

Päätteitä 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 liuketta.



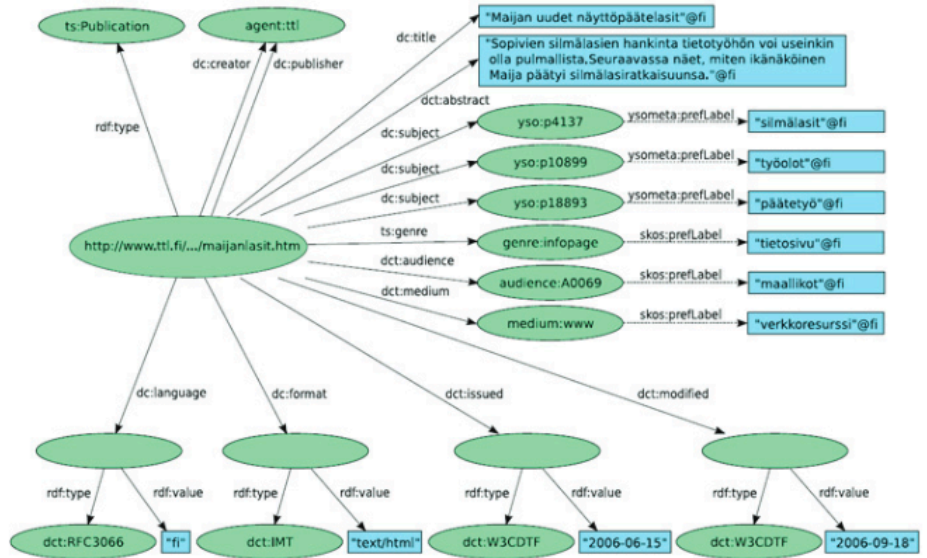
Kuva 8.2. Työterveyslaitoksen julkaisema verkkodokumentti TerveSuomi-järjestelmässä. (<http://www.ttl.fi>.)

tan harmonisointimalli. Sen avulla eri aineistojen eri tavoilla esitetty metadata voidaan muuntaa yhteentoimivalla tavalla toisiaan rikastamaan.

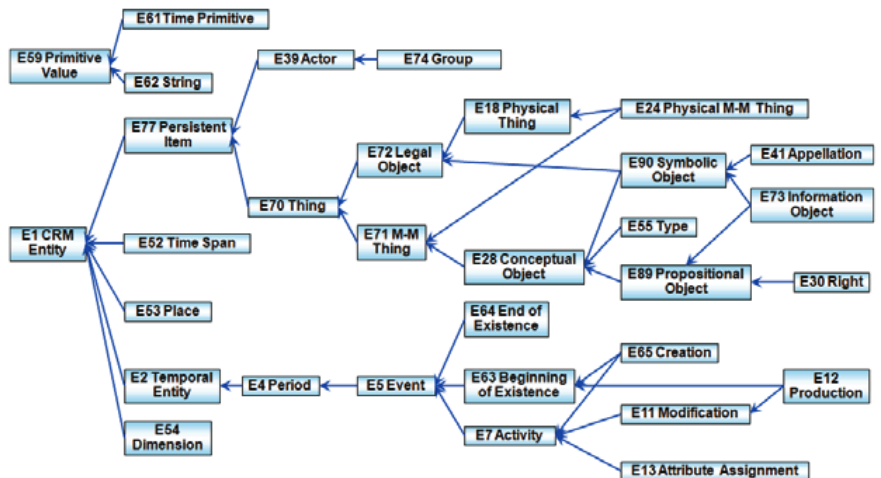
CIDOC CRM -spesifikaatio¹ sisältää 90 luokkaa, jotka on numeroitu *E1–E90*) sekä 149 ominaisuutta (*P1–P149*), joiden avulla luokkien yksilöitä voidaan yhdistää toisiinsa. Järjestelmän ydinluokkia ja niiden hierarkiaa on esitetty kuvassa 8.4. Myös CIDOC CRM -ominaisuudet muodostavat RDF Schema -mallin mukaisen hierarkian.

Esimerkiksi museossa olevaa esinettä koskeva metatieto voidaan esit-

¹ Tässä tarkasteltava mallin versio on määritelty viitteessä (Crofts ym., 2011).



Kuva 8.3. Kuvan 8.2 metatiedot RDF-verkkona kuvan 8.1 metatietomallin mukaisesti kuvattuna. (Kuva: Osma Suominen.)



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

tää 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), 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*).

apellaatio

Sisältöjä kuvaillaan tyypillisesti tapahtumaluokan (*E5 Event*) ja sen aliluokkien 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, joilla on tapahtumissa erilaisia rooleja (esimerkiksi *P11 had participant*).

CIDOC CRM -järjestelmää kehittää aktiivinen yhteisö kansainvälisen ICOM CIDOC -organisaation alaisuudessa (International Committee for Documentation¹). Mallin soveltamiseen käytännössä palataan kirjan viimeisessä osassa.

8.4 Ontologian käsite

Ontologialla (ontology) tarkoitetaan tietotekniikassa *formaalia, yhteisesti sovittua käsitteellistä kuvausta maailmasta*.²

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 tuottamien tietojen yhteentoimivuuden.
- *Käsitteellinen kuvaus* maailmasta tarkoittaa sitä, että ontologian esittämä malli vastaa jossain mielessä kuvaamaansa ilmiötä tai asiaa.

Tämä tekninen määritelmä ontologiasta poikkeaa huomattavasti termin perinteisestä merkityksestä tieteessä, jossa ontologia on olevaisen perimmäistä olemusta tutkiva filosofian osa-alue.³

¹ <http://icom.museum/the-committees/international-committees/international-committee-for-documentation/>

² Semanttisen webin ontologian käsitettä on määritelty mm. artikkelissa (Guarino ym., 2009).

³ Ontologioita semanttisen webin kontekstissa on käsitelty laajasti teoksessa (Staab & Studer, 2009).

sovellusala

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 sitä automaattisesti. Jos esimerkiksi tiedetään, että 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 metadatatamalleja. Rajanveto ei ole aina selvää, sillä semanttisessa webissä sekä metadatatamallit että käsiteontologiat esitetään yhtenäisesti RDF-perustaisena semanttisena verkkona päinvastoin kuin esimerkiksi tietokantajärjestelmissä, joissa tietokannan tietomalli (skeema) ja data on erotettu toisistaan.

tietämyksen

organisoimisjärjestelmä

Aiheontologia kuvataan tyypillisesti käyttämällä hyväksi jotain *tietämyksen organisoimisjärjestelmää* (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.

Linkitettyssä datassa pitäydytään usein yksinkertaisissa RDFS- ja SKOS-verkoissa. Seuraavaksi esitellään ensin näitä kahta yksinkertaista KOS-standardia ja sitten huomattavasti monipuolisemman OWL-kielen mahdollisuuksia. Kaikki esiteltävät järjestelmät ja kielet ovat semantiikaltaan yleiskäyttöisiä, toisin sanoen sovellusalaista riippumattomia. 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 niiden merkitysten avulla.

9

RDF Schema

Kaikki yleistykset ovat
vaarallisia, tämäkin.

Alexandre Dumas (nuorempi)
(1824–1895)

9.1 Luokat ja yksilöt

RDFS tuo semanttiseen webiin *luokan* (class) käsitteen ja idean *päätte-
lystä* (inference), jonka avulla RDF-verkkoa voidaan rikastaa automaat-
tisesti uusilla kolmikoilla. Luokalla tarkoitetaan yksinkertaisesti jouk-
koa *yksilöitä* (instance, individual), jotka kuuluvat luokkaan ja joilla on
sen ominaisuudet. Esimerkiksi Maalaukset-luokkaan kuuluvat yksittäi-
set maalaukset, kuten *Taistelevat metsot* tai *Aino-triptyykki*. Yksilön ja
sen luokan suhde ilmaistaan ominaisuudella `rdf:type`:

luokka

yksilö

```
@prefix : <http://esimerkki.fi/teos/> .  
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
  
:Taistelevat_metsot rdf:type :Maalaus .  
:Aino-triptyykki rdf:type :Maalaus .
```

Luokkien URI:t 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 .
```

9.2 Luokkien hierarkia

luokkahierarkia

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

```
@prefix : <http://esimerkki.fi/fauna/> .
@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 käytäntö on, että luokkahierarkia mahdollistaa yksinkertaisen hierarkkisen yksilö-luokkasuhteen päättelyn: yksilön voidaan aina päätellä olevan kaikkien yläluokkiensa yksilö, vaikkei tätä tietoa olisi erikseen RDF-graafissa esitetty kaikille yläluokille. 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 sen kertominen, että maalauksen yhtenä aiheena on metso, sillä aiheen kuuluminen myös tämän yläluokkiin voidaan päätellä automaattisesti.

9.3 Ominaisuuksien hierarkia

`rdfs:Class`-luokan ohella toinen keskeinen RDFS:n luokka on ominaisuuksien luokka `rdf:Property`, joka on määritelty RDF:n nimiavaruu-

nessa. 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 määrittellä *ominaisuushierarkioita* (property hierarchy) `rdfs:subPropertyOf`-suhteella. Voidaan esimerkiksi määrittellä, että sekä teoksen kirjoittaja että maalauksen maalaja ovat *aliominaisuuksia* (subproperty) ominaisuudelle `:teoksen_luoja`:

ominaisuushierarkia

aliominaisuus

```
:teoksen_kirjoittaja rdfs:subPropertyOf :teoksen_luoja .
:maalauksen_maalaja rdfs:subPropertyOf :teoksen_luoja .
```

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

9.4 Alue- ja arvorajoitteet

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

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

rajoite

aluerajoite

arvorajoite

Esimerkiksi se, että henkilöt kirjoittavat romaaneja, voidaan ilmaista näin:

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

Tarkemmin RDF- ja RDF Schema -määrittelyihin ja niihin liittyviin muihin spesifikaatioihin voi perehtyä W3C:n kotisivuilla.¹

¹ <https://www.w3.org/standards/techs/rdf>

10

SKOS – Simple Knowledge Organization System

Kaikki pitäisi tehdä niin
yksinkertaiseksi kuin
mahdollista, mutta ei yhtään
sitä yksinkertaisemmaksi.

Albert Einstein (1879–1955)

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 YKL¹ ja Helsingissä HKLJ.² Sisällön tarkempaan kuvailuun on käytetty muun muassa 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 käsitteiden kuvaaminen sellaisenaan semanttisena rakenteena on usein hyödyllistä sovellusten kannalta.

¹ <http://ykl.kirjastot.fi/>

² <http://hklj.kirjastot.fi/>

Jotta eri aloilla kehitettyjä laajoja luokituksia ja sanastoja voitaisiin hyödyntää mahdollisimman vähällä vaivalla semanttisessa webissä, on niiden kuvaamista varten kehitetty oma standardinsa, Simple Knowledge Organization System (SKOS).¹ Se tarjoaa valmiit RDF-perustaiset ratkaisut, joilla useimmat yksinkertaiset luokitukset ja sanastot voidaan esittää yhteismitallisessa semanttisessa muodossa. SKOS-sanasto on määritelty seuraavassa nimiavaruudessa, josta käytetään jatkossa nimeä *skos*:

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

10.1 Käsitemalli

Ominaisuus	Merkitys
skos:inScheme	Ilmaisee käsitelyksilön käsitemallin.
skos:hasTopConcept	Viittaa käsitemallista sen pääkäsitteisiin.
skos:topConceptOf	Kertoo pääkäsitteen käsitemallin.

Taulukko 10.1. Käsittemallin ja sen käsitteiden väliset viittaukset SKOS-suosituksessa.

käsittemalli	SKOS esittää tietämystä <i>käsittemallina</i> (concept scheme), joka koos-
käsite	tuu joukosta <i>käsitteitä</i> (concept). Käsittemalli määritellään luokan
	<code>skos:ConceptScheme</code> yksilönä, ja yksittäiset käsitteet luodaan luokas-
	ta <code>skos:Concept</code> . Käsittemalliin kuuluu yksi tai useampia <i>pääkäsitteitä</i>
pääkäsite	(top concept), jotka liitetään toisiinsa taulukon 10.1 kolmen eri ominai-
	suuden avulla. Pääkäsitteille voidaan sitten määritellä merkitykseltään
	<i>suppeampia</i> (narrower) alikäsitteitä, jotka muodostavat asteittain tar-
	kentuvia käsitteiden hierarkioita.

10.2 Käsitteen nimikkeet

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

¹ <http://www.w3.org/TR/skos-reference/>

Käsitteellä voi olla kolmenlaisia nimikkeitä (ks. taulukko 10.2):

1. Jokaista kielestä yksi nimike on valittu *suositelluksi nimikkeeksi* (preferred label).
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.

suositeltu nimike

vaihtoehtoinen nimike

piilonimike

Ominaisuus	Merkitys
skos:prefLabel	Käsitteen suositeltu nimike, joita on yksi jokaisella eri kielellä. Sekaannusten välttämiseksi nimikkeet kannattaa määritellä yksilöivästi kussakin kielessä niin, ettei kahdella eri käsitteellä ole samaa suositeltua nimikettä, vaikka se periaatteessa on mahdollista.
skos:altLabel	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 10.2. Ominaisuudet SKOS-käsitteen eri nimikkeiden ilmaisemiseksi.

Seuraavaksi on esimerkkinä määritelty biologinen eläinsanasto, jolla on kaksi pääkäsitettä linnut ja kalat, joista linnut-käsitteellä on suppeampi käsite lokit.

```
@prefix : <http://www.esimerkki.fi/fauna/> .
@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 ;
```

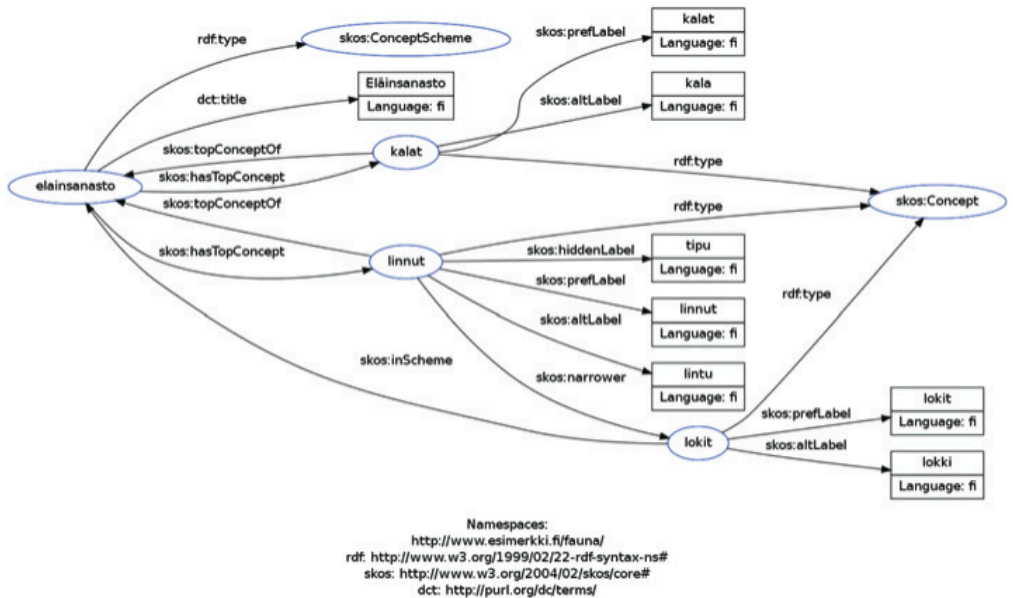
```

skos:altLabel "lintu"@fi ;
skos:hiddenLabel "tipu"@fi ;
skos:topConceptOf :elainsanasto ;
skos:narrower :lokit . # Merkitykseltään suppeampi käsite

: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 .

```



Kuva 10.1. SKOS-sanasto RDF-verkkona.

Esimerkkiä vastaava RDF-graafi on esitetty kuvassa 10.1. 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 eikä niinkään reaali maailman 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). Usein tämä olisi kuitenkin tarpeellista. Esimerkiksi henkilön nimi voi naimisiinmenon yhteydessä muuttua tai samalla paikalla voi olla eri aikoina eri nimi, jolloin nimike pitäisi voida *kvalifioida* (qualify) ajan suhteen ja kertoa sen voimassaoloaika.

kvalifiointi

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 luokan `skosxl:Label` yksilöinä, jotka liitetään emokäsitteeseen ominaisuuksien `skosxl:prefLabel`, `skosxl:altLabel` ja `skosxl:hiddenLabel` arvoina. Nimikeluokan yksilölle voidaan sitten antaa ominaisuudella `skosxl:literalForm` nimikkeen literaaliarvo, ja samaan tapaan voidaan nimikkeisiin liittää muutakin tietoa, esimerkiksi nimikkeen voimassaoloaika. Nimikkeitä voidaan myös linkittää toisiinsa `skosxl:labelRelation`-ominaisuuden aliominaisuuksilla, joita voi määrittellä tarpeen mukaan.

10.3 Notaatit

Luokituksissa ja sanastoissa käytetään usein erityisiä tunnisteita eli *notaatioita* niissä esiintyvien termien yksilöintiin. Nimikkeistä poiketen notaatit 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 notaatit 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>
```

10.4 Semanttiset relaatiot

Ominaisuus	Merkitys
<code>skos:broader</code>	Viittaus tarkemmasta käsitteestä yleisempään.
<code>skos:narrower</code>	Viittaus yleisemmästä käsitteestä tarkempaan.
<code>skos:related</code>	Muu assosiatiivinen viittaus käsitteestä siihen liittyvään toiseen käsitteeseen.

Taulukko 10.3. SKOS-käsittemalliin kuuluvien käsitteiden väliset semanttiset relaatiot.

semanttinen relaatio	SKOS-käsitteiden merkityksen määrittely tietokoneille perustuu joukkoon yksinkertaisia <i>semanttisia relaatioita</i> , jotka yhdistävät käsitteet laajemmaksi käsiteverkoksi. Keskeisin suhdetyyppi on <i>hierarkkinen relaatio laajemman</i> (broader) ja <i>suppeamman</i> (narrower) käsitteen välillä.
hierarkkinen relaatio	
laajempi käsite	Sen avulla käsitteistä muodostuu puumaisia rakenteita. Lisäksi voidaan kertoa, että kahden käsitteen välillä on jokin muu tarkemmin määritteleminen semanttinen assosiaatiosuhde. Nämä semanttiset relaatiot on lueteltu taulukossa 10.3.
suppeampi käsite	

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 kyselyä voidaan laajentaa esimerkiksi siten, että mukaan otetaan myös hakusanoja merkitykseltään suppeampia termejä tai hakusanoihin vain assosiatiivisesti liittyviä termejä.

Huomionarvoista kuitenkin on, että hierarkkinen suhde (`skos:broader` tai `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!

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

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

Koska suppeammasta ominaisuudesta voidaan päätellä yleisemmän ominaisuuden olevan myös voimassa, voitaisiin edellisessä 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 esimerkiksi kyselyn laajentamiseen. Seuraavassa 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 sovelluskohtaisilla 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.

10.5 Ryhmittelevät termit

		Top of the AAT hierarchies
		 Objects Facet
		 Furnishings and Equipment (hierarchy name)
		 Furnishings (hierarchy name)
		 furnishings (artifacts)
		 <furnishings by form or function>
		 mirrors
		 <mirrors by form>
		 <mirrors by form of frame>
		 architectural mirrors
		 bilbao mirrors
		 cheval glasses
		 Constitution mirrors
		 girandole mirrors

Kuva 10.2. Getty-säätiön AAT-sanaston peileihin liittyviä hierarkkisia ja ryhmitteleviä käsitteitä, jotka on esitetty kulmasuluissa. (<http://www.getty.edu/research/tools/vocabularies/aat/>.)

Monissa sanastoissa on varsinaisten indeksoinnissa käytettävien termien ja luokitusten ohella myös *ryhmitteleviä käsitteitä* (group concept), jotka on tarkoitettu helpottamaan sanaston jäsentämistä ihmiskäyttäjän kannalta mutta joita ei käytetä tietojen kuvailussa. Esimerkiksi Getty-säätiön Art and Architecture Thesaurus (AAT) -sanastossa¹ on ryhmittelevä käsite² *<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 10.2). Tällaiset käsitteet voidaan esittää SKOS-mallissa *kokoelmina* (collection) luokan *skos:Collection* yksilöinä. Kokoelmaan kuuluvat käsitteet tai toiset kokoelmat luetellaan kokoelmayksilön ominaisuuden *skos:member* arvoina.

Mikäli kokoelman jäsenet halutaan esittää jossain tiettyssä järjes-

¹ <http://www.getty.edu/research/tools/vocabularies/aat/>

² Näitä kutsutaan AAT-sanastossa *ohjaustermeiksi* (guide term), mutta koska tätä termiä käytetään muissa sanastoissa yleensä merkityksessä ”vaihtoehtoinen termi”, tässä puhutaan ryhmittelevistä käsitteistä.

tyksessä, 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-päätelyn 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.

järjestetty kokoelma

10.6 SKOS-sanastojen siltaaminen

Ominaisuus	Merkitys
<code>skos:closeMatch</code>	Lähes sama käsite kahdessa eri käsitemallissa
<code>skos:exactMatch</code>	Sama käsite kahdessa eri käsitemallissa
<code>skos:broadMatch</code>	Laajempi käsite toisessa käsitemallissa
<code>skos:narrowMatch</code>	Suppeampi käsite toisessa käsitemallissa
<code>skos:relatedMatch</code>	Merkitykseen liittyvä käsite toisessa käsitemallissa

Taulukko 10.4. Eri käsitemalleihin kuuluvien SKOS-käsitteiden siltaamiseen tarkoitetut ominaisuudet.

Edellä on käsitelty yhden sanaston käsitteiden kuvaamista SKOS-mallissa. Eri aineistojen sisällönkuvaailussa käytetään usein monia eri 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 10.4 mukaisesti. Näiden avulla on mahdollista kuvata kahden *eri* sanaston käsitteiden välisiä suhteita.¹

siltausominaisuus

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

¹ Ontologioiden *siltaus* (mapping, alignment, matching) on oma semanttisen webin tutkimusalueensa, jota on esitelty esim. teoksessa (Euzenat & Shvaiko, 2007).

```

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-päätelyssä voidaan esimerkiksi kolmikosta

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

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 .

```

10.7 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 10.5 on esitty yhteenveto käytettävissä olevista dokumentointiominaisuuksista.

10.8 Päättelysäännöt ja integriteettiehtoja

integriteettiehto

SKOS-malli sisältää 46 *integriteettiehtoa* (integrity constraint), joiden oletetaan olevan voimassa valideissa SKOS-sanastoissa. Esimerkiksi ehto S9 edellyttää, että sama resurssi ei voi olla samanaikaisesti luokan `skos:ConceptScheme` ja `skos:Concept` yksilö. Integriteettisääntö-

Ominaisuus	Merkitys
<code>skos:note</code>	Yleinen dokumentointiominaisuus
<code>skos:changeNote</code>	Käytetään käsitteen pienten muutosten kuvaamiseen käsittemallin ylläpitoa varten
<code>skos:definition</code>	Käsitteen merkityksen kuvaus tai määritelmä
<code>skos:editorialNote</code>	Käsitteen hallintaan liittyvä kommentointi, esimerkiksi huomautus keskeneräisestä työstä
<code>skos:example</code>	Ominaisuudella annetaan esimerkkejä käsitteen käytöstä
<code>skos:historyNote</code>	Käsitteeseen tehdyt aiemmat muutokset
<code>skos:scopeNote</code>	Erityisesti käsitteen käyttöön liittyvää kuvausta

Taulukko 10.5. SKOS-käsitteiden dokumentointiin tarkoitetut ominaisuudet. Näiden arvona on ihmisen luettavaksi tarkoitettu merkkijono.

jen avulla voidaan määritellä SKOS-sanaston semantiikkaa, täsmentää sen käyttöä sekä validoida sanastoja.

11

Web Ontology Language OWL

Sanat ovat koukkuja ajatusten
ripustamista varten.

Henry Ward Beecher
(1813–1857)

Ontologiat ovat koukkuja
sanojen ripustamista varten.

Eero Hyvönen

RDF Schema ja SKOS ovat yksinkertaisia kieliä semanttisen webin (aihe)ontologioiden esittämiseen. 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 W3C on standardoinut ilmaisuvoimaisen kielen, Web Ontology Language OWL:n.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:

¹ <https://www.w3.org/2001/sw/wiki/OWL>

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

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

11.1 Johdatus logiikkaan

Logiikka muodostaa semanttisen webin semanttisen perustan. Sen avulla määritellään verkon datan ja ontologioiden merkitys täsmällisesti siten, että sekä ihmiset että koneet voivat ymmärtää sen yksikäsitteisesti. Logiikka mahdollistaa datan automaattisen rikastamisen uutta tietoa päättelämällä sekä automaattisen *ongelmien ratkonn*an (problem solving). Tässä mielessä logiikka voidaan rinnastaa aritmetiikkaan, jonka avulla voidaan myöskin 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.

11.1.1 Logiikan idea

Aristotelesta voidaan pitää sekä ensimmäisenä ontologina että loogikon, sillä hän esitti ensimmäisen mallin päättelystä mekaanisena järjestelmänä. Aristoteleen järjestelmä perustuu tiedon esittämiseen väitelauseina (väittäminä) eli *propositio*ina (proposition). Propositionot esittävät sovellusmaailmaa (domain) kuvaavia *tosiasioita* (fact), ja Aristotelellä niitä oli neljää eri tyyppiä:

propositio
tosiasia

1. "Kaikki A ovat B "
2. "Jokin A on B "
3. "Mikään A ei ole B "
4. "Jokin A ei ole B "

Propositioista voidaan johtaa uusia propositioneja *syllogismien* (syllo-

syllogismi gism) avulla. Syllogismit ovat muotoa

$$C_1, \dots, C_n \rightarrow H$$

olevia sääntöjä, ja ne voidaan tulkita seuraavasti: jos säännön ehdot eli *premissit* (premise)¹ $C_1, \dots, C_n$ pitävät paikkansa, myös *päätelmä* (conclusion) H pitää paikkansa. Esimerkiksi kolmesta muotoa *Kaikki A ovat B* olevasta propositiosta voidaan muodostaa syllogismi

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).

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, toisin sanoen uusia kaaria.

11.1.2 Lauselogiikka

tekoäly Semanttinen web perustuu *tekoälytutkimuksen* (Artificial Intelligence)²
 predikaattilogiikka piirissä paljon käytettyyn *predikaattilogiikkaan* (predicate logic). Predikaattilogiikan perustana on yksinkertaisempi *lauselogiikka* (propositional logic), jonka ilmaisuvoimaa se laajentaa. Lauselogiikassa tietoa esitetään propositioina (väitelauseina), 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).

¹ Aristoteleen syllogismeissa oli vain kaksi ehtoa.

² Hyvä, laajempi johdanto tekoälyyn on teos (Russell & Norvig, 2010). Suomeksi aiheesta on ilmestynyt aiemmin mm. teos (Hyvönen ym., 1993).

Lauseita, jotka eivät sisällä toisia lauseita, kutsutaan *atomilauseiksi* (atomic formula) tai myöhemmin esiteltävässä, lauselogiikkaa yleisemmässä predikaattilogiikassa *atomikaavoiksi*.

atomikaava

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 11.1. Lauselogiikan konnektiivit.

Lauselogiikan ideana on tarjota mahdollisuus atomilauseiden yhdistämiseen konnektiiveilla, jolloin maailmaa voidaan kuvailla rikkaammin. Yleisimmin käytetyt konnektiivit ja niiden merkitykset on esitetty taulukossa 11.1.

A	B	$A \Rightarrow B$
T	T	T
T	E	E
E	T	T
E	E	T

Taulukko 11.2. Implikaation ($\Rightarrow$) totuustaulu.

Lauselogiikan lauseita voidaan rakentaa yhdistelemällä konnektiiveilla atomilauseita ja muita lauseita sisäkkäisiksi rakenteiksi. Näiden totuusarvot voidaan laskea kaikilla atomilauseiden arvokombinaatioilla käyttäen taulukon 11.1 konnektiivien merkityksiä, jotka määrittelevät *totuustauluna* (truth table). Esimerkiksi implikaation merkityksen määrittelevä totuustaulu on esitetty taulukossa 11.2.

totuustaulu

Tarkastellaan esimerkkinä totuustaulujen käytöstä lausetta

$$((A \Rightarrow B) \wedge A) \Rightarrow B,$$

joka on nimeltään *modus ponens*. Sen osien totuustaulut on esitetty taulukossa 11.3. Siitä huomataan, että modus ponens on aina tosi maailman-

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 11.3. *Modus ponens* -lauseen totuustaulun laskeminen.

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 kombinaatiolla.

ristiriita
toteutuva
kontingentti

11.1.3 Predikaattilogiikka

Predikaattilogiikka lisää olennaisesti lauselogiikan ilmaisuvoimaa. Keskeinen idea on atomilauseiden korvaaminen tietoa hienosyisemmin esittävillä *predikaateilla*:

predikaatti

$predikaatti(Arg_1, \dots, Arg_n)$

Esimerkiksi:

```
yliopisto(Aalto)
on-osa(Otaniemi, Espoo)
vaimo(Jean, Aino)
antoi(Jean, Aino, sormus)
```

Predikaatti kuvaa sen argumenttien välistä suhdetta, relaatiota. Argumentit voivat olla vakiosymboleita eli *vakioita* (constant), kuten edellisissä esimerkeissä, mutta myös *muuttujia* (variable), joita merkitään jatkossa $x, y, \dots$. Esimerkiksi se, että Akseli on naimisissa jonkun tuntemattoman naispuolisen henkilön x kanssa, voidaan ilmaista muuttujalla:

vakio
muuttuja

vaimo(Akseli, x)

Predikaatin argumenttina voi vakion ja muuttujan ohella olla myös *funktio* (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 sen kutsua voidaan käyttää edustamaan henkilön henkilötunnusta.

Matemaattisesti funktiolla $f(x_1, \dots, x_n)$ 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).

termi

perustermi

peruskaava

Predikaattien lisäksi toinen predikaattilogiikan keskeinen innovaatio ovat *kvanttorit* (quantifier), joilla voidaan ilmaista väitelauseita muuttujan eri arvojen yli. *Universaalikvanttorilla* (universal quantifier)

kvanttori

universaalikvanttori

$\forall x$ *lause*

voidaan kertoa, että *lause* pitää paikkansa kaikilla (for all) arvoilla x . *Eksistentiaalikvanttorilla* (existential quantifier)

eksistentiaalikvanttori

$\exists x$ *lause*

taas voidaan kertoa, että *lause* pitää paikkansa ainakin jollain x :n arvolla. Esimerkiksi käsitteet *Aikuinen* ja *Poikamies* voidaan määritellä näin:

$$\forall x \text{ Aikuinen}(x) \Leftrightarrow \text{Henkilö}(x) \wedge \text{ikä}(x) \geq 18$$

$$\forall x, y \text{ Poikamies}(x) \Leftrightarrow \text{Mies}(x) \wedge \text{Aikuinen}(x) \wedge \neg \exists y \text{ Naimisissa}(x, y)$$

Predikaattilogiikassa tietämys esitetään joukkona *lauseita* (statement), joiden syntaksi eli rakenne voidaan määritellä seuraavilla säännöillä:¹

lause

¹ Predikaattilogiikasta on olemassa useita eri muunnoksia. Näiden määrittelyssä voi olla mukana myös esim. kahden objektin *samuutta* (equality) osoittava symboli =.

$$\begin{aligned}
&Lause \mapsto Atomilause | Lause \text{ Konnektiivi } Lause | \\
&\quad Kvanttori \text{ Muuttuja } Lause | \\
&\quad \neg Lause | (Lause) \\
&Atomilause \mapsto Predikaatti(Termi, Termi, \dots) | Termi = Termi \\
&\quad Termi \mapsto Vakio | Muuttuja | Funktio(Termi, Termi, \dots) \\
&Konnektiivi \mapsto \vee | \wedge | \Rightarrow | \Leftrightarrow \\
&Kvanttori \mapsto \exists | \forall \\
&Vakio \mapsto a | b | \dots \text{vakioidennimet} \\
&Muuttuja \mapsto x | y | \dots \text{muuttujiennimet} \\
&Predikaatti \mapsto \text{omistaa} | \text{myi} | \dots \text{predikaattiennimet} \\
&Funktio \mapsto \text{ikä} | \text{sotu} | \text{summa} | \dots \text{funktioidennimet}
\end{aligned}$$

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 edellä mainittujen 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 väitelauseen (proposition) mallinnuksen kohteena olevaan maailmaan (domain) liittyen soveltamalla predikaattia vakiotermeihin tai -funktiokutsuihin. Esimerkiksi:

```
omistaa(Matti,Auto521)
myi(Matti,Auto521,Pekka)
```

Predikaatin tulkinnasta riippuen väitelause on joko tosi tai epätosi. Atomilauseiden totuusarvon määrittely perustuu predikaattien, funktioiden ja vakioiden sovelluskohtaisiin määritelmiin, joissa tarkastelun kohteena oleva maailma on kuvattu matemaattisena joukkona M olioita, joihin viitataan vakioilla. Funktiot määritellään matemaattisina funktioina¹ joukossa M ja predikaatit relaatioina $R \subseteq M_1 \times M_2 \times \dots M_n$. Va-

¹ Funktion avulla voidaan laskea yksikäsitteinen arvo sen argumenteista; tässä argumentit ja funktion arvo otetaan joukosta M .

kioiden, funktioiden ja predikaattien sovellusaluekohtaista määritelmää kutsutaan *tulkinnaksi* (interpretation).

tulkinta

Mutkikkaampien lauserakenteiden totuusarvo voidaan määrittää luontevasti semanttisilla säännöillä. Niitä soveltamalla mielivaltaisen logiikan lauseen totuusarvo voidaan selvittää seuraavien sääntöjen ja atomilauseiden totuusarvojen avulla:

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. Tällöin väitelauseen totuusarvo on riippuvainen siitä, mikä arvo muuttujalle valitaan tai funktiokutsulle saadaan. Tähän perustuu predikaattilogiikan käyttäminen kyselykielenä, jossa kysely muotoillaan joukoksi muuttujia sisältäviä lauseita. Vastauksena on joukko muuttujasijoituksia, joissa muuttujille on annettu arvoiksi sellaiset vakiot, 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 SPARQL:n kaltaisena kyselykielenä vaan sen avulla päätellään uusia RDF-kolmikoita tietämuskantaan. Ontologiset luokkakäsitteet määritellään yksipaikkaisina predikaatteina. Esimerkiksi käsitteet *Aikuinen* ja *Poikamies* voidaan määritellä näin:

$$\forall x \text{ Aikuinen}(x) \Leftrightarrow \text{Henkilö}(x) \wedge \text{ikä}(x) \geq 18$$

$$\forall x,y \text{ Poikamies}(x) \Leftrightarrow \text{Mies}(x) \wedge \text{Aikuinen}(x) \wedge \neg \exists y \text{ Naimisissa}(x,y)$$

Aikuinen(*x*) ja *Poikamies*(*x*) vastaavat luokkia eli niiden yksilöiden (vakioiden) joukkoja, joille predikaatti on tosi. Tulkinnessa määriteltyjen vakioiden, funktioiden ja predikaattien avulla voidaan päätellä, mihin luokkiin yksilöt kuuluvat.

Oletetaan esimerkiksi, että on määritelty *Naimisissa*-predikaatilla, joka on naimisissa kenenkin kanssa sekä henkilöiden sukupuoli. 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 (tietäen oletuksien), 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 aksioomia, jotka määrittelevät kieleen kuuluvien predikaattien merkityksen. Näiden avulla päättely on mahdollista. Esimerkiksi `RDF(S):n rdfs:subClassOf`on kaksipaikkainen transitiivinen predikaatti, johon pätee:

$$\forall x,y,z \text{ rdfs:subClassOf}(x,y) \wedge \text{ rdfs:subClassOf}(y,z) \Rightarrow \text{ rdfs:subClassOf}(x,z)$$

aksiooma

Aksiooma on perusmääritelmä, joka on universaalisti tosi (tautologia) kaikissa mahdollisissa eri tulkinnoissa (maailmoissa). Aksiooma mahdollistaa näin uusien kaarien päättelyn RDF-verkkoon samalla tavalla kaikissa semanttisen webin sovelluksissa.

OWL-ontologian luokkakäsitteiden eli termien määrittelystä käytetään nimitystä *TBox* (terminological box). Ominaisuuksien määrittelymien 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.

TBox

RBox

ABox

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 usein nopeuttaa ajonaikaista laskentaa.

11.2 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 viime kädessä semanttisena RDF(S)-verkkona ja voidaan kirjoittaa esimerkiksi Turtle-notaatiolla.

OWL:n logiikan lauseiden kirjoittaminen verkkona on kuitenkin usein kömpelöä, ja siksi OWL-kielelle on kehitetty yksinkertaisempia notatioita samaan tapaan kuin Turtle kehitettiin yksinkertaisemmaksi RDF-verkkojen esittämisenä notatioksi 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.¹
2. **OWL-XML** XML-perustainen erityiskieli OWL:n ilmaisemiseksi.
3. **Funktionaalinen syntaksi**, jota käytetään erityisesti OWL-

funktionaalinen syntaksi

¹ <https://www.w3.org/TR/owl2-mapping-to-rdf/>

Manchesterin syntaksi

spesifikaatioissa, muun muassa W3C:n OWL-määrittelydokumenteissa.

4. Manchesterin syntaksi, joka pyrkii olemaan mahdollisimman yksinkertainen notaatio niin, että sen käyttö vaatii mahdollisimman vähän logiikan osaamista.¹

Lisäksi OWL-kieltä varten on käytettävissä graafisia editoreita, kuten Protégé² ja TopBraid Composer,³ joiden avulla OWL-kielen monia rakenteita voi muokata interaktiivisesti hiiren avulla ja vain vähän tekstiä editoiden.

Kaikissa tapauksissa kielen semantiikka on kuitenkin sama predikaattilogiikan avulla määritelty. Vaikka OWL perustuu melko sofistikoituun logiikkaan, sen tavoitteena on olla kieli, jota voitaisiin käyttää ilman syvällistä logiikan osaamista. Käytännön sovelluksissa OWL:sta otetaan kuitenkin usein käyttöön vain sen yksinkertaisempia muotoja.

OWL lisää RDF Scheman ilmaisuvoimaa merkittävästi tarjoamalla useita uusia tapoja luokkien, yksilöiden ja ominaisuuksien määrittelemiseen. Seuraavaksi annetaan yleiskuva näistä tavoista esimerkkien avulla. Esitys perustuu W3C:n OWL Primer -dokumenttiin, jossa määritellään monipuolisesti sukulaissuhteiden ontologiaa.⁴ 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). Seuraavissa esimerkeissä käytetään erityisesti funktionaalista syntaksia. Siinä esitysmuotona ovat sisäkkäiset funktiokutsut, joiden muoto on:

$$funktio(argumentti_1 \ argumentti_2 \ ... \ argumentti_n)$$

Esimerkiksi:

```
EquivalentClasses(
  :Äiti
  ObjectIntersectionOf( :Nainen :Vanhempi )
)
```

Kaikissa syntakseissa käytetään RDF(S)- ja OWL-spesifikaatiossa

¹ <https://www.w3.org/TR/owl2-manchester-syntax/>

² Protégé on Stanfordin yliopistossa USA:ssa kehitetty, paljon käytetty open source -ontologiaeditori, joka on ladattavissa ohjeineen osoitteesta <http://protege.stanford.edu>.

³ TopBraid Composer on kaupallinen vaihtoehto, josta on ollut saatavilla myös ilmainen, rajoitettu akateeminen versio.

⁴ <https://www.w3.org/TR/owl2-primer/>

määriteltyjä luokkia ja ominaisuuksia, jotka on kuvattu teknisesti RDF(S)-muodossa seuraavassa 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 määritelty luokan rdfs:Class aliluokaksi. 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.¹

11.3 Ontologiadokumentti

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 ilmaistuna
- Ontologian kuvailu ihmislukijalle annotointi-ominaisuuksien avulla

Ontologian perustana ovat 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

¹ <https://www.w3.org/TR/owl2-syntax/>

Kaikkien ontologiassa kuvattujen entiteettien tyyppi ilmaistaan ontologiassa deklaraatioiden avulla.¹ 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 reaali maailman yksilöitä ja niiden välisiä suhteita datana (ABox), voidaan kuvauksen perusteella datasta päätellä uusia suhteita (kolmikoita) entiteettien välille, eli rikastaa dataa älykkäästi.

Seuraavissa esimerkeissä esitellään ontologisten käsitteiden määrittelyä OWL-kielillä. Esitys perustuu laajempaan, W3C:n julkaisemaan esimerkkiin sukulaisuussuhteiden määrittelystä OWL-dokumenttina, joka on esitetty alkukielellä liitteessä A.

11.4 Luokkien määrittely

OWL perustuu RDFS:n tapaan luokkien ja ominaisuuksien hierarkioihin, 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.

¹ Manchesterin syntaksissa deklaraatioita ei kuitenkaan käytetä, vaan kone päättelee ne automaattisesti.

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 määrittelyn terminologian mukainen data, esimerkiksi se, että resurssi `x-4564` on luokan `Henkilö` yksilö ja että hän opiskelee, esitetään luokkien yksilöinä ja näiden välisinä suhteina (ABox) RDF:n avulla.

11.4.1 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 Helsingillä on eri tunniste Maanmittauslaitoksen Paikannimirekisterissä, Geonames-rekisterissä ja DBpediassa. Tämän vuoksi OWL:ssa ei tehdä niin sanottua *yksikäsitteisten nimien oletusta* (Unique Name Assumption, UNA), jonka mukaan eri tunnisteilla kuvatut resurssit kuvaavat eri asioita. Tämä tiedon esittämistä yksinkertaistava oletus voidaan yleensä tehdä tietokantajärjestelmissä, joissa data on peräisin yhdestä lähteestä ja tunnisteet on muodostettu systemaattisesti koko tietokannan perusteella.

yksikäsitteisten nimien
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 funktionaalista syntaksia 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.¹ Esimerkissä ”ö” on korvattu kirjaimella ”o”, koska edellä mainittu verkkopalvelu käyttää perinteistä ASCII-merkistöä.

Vastaavasti voidaan kertoa, että joukko luokkia on pareittain erillisiä, jolloin millään joukon luokalla ei ole yhtään yhteistä yksilöä minkään toisen luokan kanssa:

```
DisjointClasses( :Mies :Nainen :Koira :Kissa )
```

Tämä ilmaistaan RDF-verkolla huomalla yksilö luokasta

```
owl:AllDisjointClasses
```

ja luettelemalla erilliset luokat listana, joka asetetaan owl:members ominaisuuden arvoksi:

```

# [] on nimetön resurssi
[] rdf:type owl:AllDisjointClasses ;
    owl:members ( :Mies :Nainen :Koira :Kissa ) .

```

11.4.2 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 ekvivalenssin avulla seuraavasti:

¹ Palvelu on käytettävissä Manchesterin yliopiston verkkosivuilla <http://owl.cs.manchester.ac.uk/tools/webapps/owl-syntax-converter/>, ja myös Linked Data Finland -julkaisualustalla on vastaava palvelu: <http://www.ldf.fi/service/owl-converter/>.

```
EquivalentClasses(
  :Viikonpäivä
  ObjectOneOf( :Maanantai :Tiistai :Keskiviikko :Torstai
               :Perjantai :Lauantai :Sunnuntai )
)
```

Lisäksi luokka voidaan määritellä implisiittisesti määrittelemällä sen ominaisuuksille *ominaisuusrajoitteita* (property restriction). Luokan yksilöitä ovat tällöin ne resurssit, jotka ovat rajoitteiden mukaisia.

ominaisuusrajoite

Yksinkertaisin ominaisuusrajoite 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 ominaisuusrajoiteyksilö.

ObjectHasValue-muodon ohella käytössä on ObjectHasSelf-muoto, jolla voi esittää vaatimuksen siitä, että yksilön ominaisuus viittaa yksilöön itseensä (self restriction). Esimerkiksi narsistin käsite merkityksessä ”itseään rakastava” voidaan määritellä näin:

```
EquivalentClasses (
  :Narsisti
  ObjectHasSelf ( :rakastaa )
)
```

Ominaisuusrajoite voi koskea myös ominaisuuden arvojen lukumäärää eli olla *kardinaliteettirajoite* (cardinality restriction) tai rajata arvo-

kardinaliteettirajoite

jen tyyppiä eli olla *tyyppirajoite*. Lukumäärälle voidaan asettaa minimi (`owl:ObjectMinCardinality`), maksimi (`owl:ObjectMaxCardinality`) ja tarkka arvo (`owl:ObjectExactCardinality`). Esimerkiksi se tieto, että Matilla on korkeintaan kaksi poikaa, voidaan ilmaista funktionaalisella syntaksilla näin:

```
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:

$$\forall x, y \text{ OnnellinenHenkilö}(x) \Leftrightarrow \text{lapsi}(x, y) \wedge \text{OnnellinenHenkilö}(y)$$

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.

11.4.3 Luokkien määrittely joukko-opilla

Luokkia voi määritellä myös joukko-opillisten operaatioiden unioni (`ObjectUnionOf`), leikkaus (`ObjectIntersectionOf`) ja komplementti (`ObjectComplementOf`) avulla. Esimerkiksi Äiti-luokka voidaan määrittellä 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 luokkaan kuulumiselle. Esimerkiksi määrittelyn

```
SubClassOf(
  :Isoisä
  ObjectIntersectionOf( :Mies :Vanhempi )
)
```

mukaan `Isoisä`-luokan yksilöt ovat välttämättä `mies`- ja `vanhempi`-luokan yksilöitä, mutta `mies`- ja `vanhempi`-luokkiin kuulumisesta ei käänteisesti välttämättä seuraa, että yksilö on `isoisä`.

11.5 Ominaisuuksien määrittely

OWL:ssa on kolmenlaisia ominaisuusluokkia:

1. *Objektiominaisuudet* ovat luokan `owl:ObjectProperty` yksilöitä ja ilmaisevat resurssien välisiä suhteita. objektiominaisuus
2. *Tietotyyppiominaisuudet* 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 päättelyä. annotointiominaisuus

Seuraavaksi tarkastellaan näitä ominaisuusluokkia erikseen.

11.5.1 Objektiominaisuudet

Objektiominaisuuksille voidaan määritellä seuraavia matemaattisia suhteita ja piirteitä, jotka ovat varsin hyödyllisiä päättelyssä.

Käänteisominaisuus

käänteisominaisuus Ominaisuudelle voidaan määritellä *käänteisominaisuus*. Esimerkiksi *aviomies*-ominaisuuden käänteisominaisuus on *aviovaimo*:

```
:aviomies owl:inverseOf :aviovaimo .
```

Ideana on, että jos jompikumpi suhde tiedetään, toinen voidaan automaattisesti päätellä ja lisätä vastaava kolmikko RDF-tietämyskantaan.

Ominaisuuksien poissulkevuus

poissulkeva ominaisuus Kaksi ominaisuutta voidaan määritellä toisensa *poissulkeviksi* (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:

```
:puoliso owl:propertyDisjointWith :lapsi .
```

Symmetrinen ominaisuus

symmetrinen ominaisuus Ominaisuus voidaan määritellä symmetriseksi luomalla ominaisuusyksilö luokasta `owl:SymmetricProperty`. Esimerkiksi *aviopuoliso* on symmetrinen suhde:

```
:aviopuoliso rdf:type owl:SymmetricProperty .
```

Symmetrinen suhde voidaan päätellä automaattisesti ”molempiin suuntiin”: jos x on y :n *aviopuoliso*, symmetrisesti myös y on x :n *aviopuoliso*.

OWL:ssa voi myös ekplisiittisesti kertoa, että jokin ominaisuus ei ole symmetrinen. Esimerkiksi vanhemmat eivät voi olla lapsiensä lapsia:

```
:lapsi rdf:type owl:AsymmetricProperty .
```

Refleksiivinen ominaisuus

refleksiivinen ominaisuus *Refleksiivinen ominaisuus* liittää yksilön automaattisesti itseensä. Esimerkiksi *sukulainen* on refleksiivinen ominaisuus, koska jokainen on itsensä *sukulainen*:

```
:sukulainen rdf:type owl:ReflexiveProperty .
```

Käänteisesti ominaisuus voidaan myös ilmoittaa *irrefleksiiviseksi* (ir-reflexive property). Esimerkiksi kukaan ei voi olla itsensä isä:

```
:isä rdf:type owl:IrreflexiveProperty .
```

irrefleksiivinen
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.

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 .
```

Yksiavioisessa maailmassa myös *aviopuoliso* on käänteisesti funktionaalinen ominaisuus.

käänteisesti
funktionaalinen
ominaisuus

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, voidaan päätellä, että x on myös z :n perillinen:

```
:perillinen rdf:type owl:TransitiveProperty .
```

transitiivinen
ominaisuus

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, että lapsenlapsesta lähtee ominaisuuskaari **vanhempi** tämän vanhempaan ja tästä samanlainen kaari edelleen isovanhempaan.

Avaimet

avainmekanismi

Yksilön tunnistaminen sen ominaisuuksien perusteella on OWL-päätelyn yksi keskeinen tehtävä. Eräs hyvä väline tähän ovat edellä kuvatut käänteisesti funktionaaliset ominaisuudet. Toinen vielä geneerisempi tapa on käyttää OWL:n *avainmekanismia*, jossa luokkaan liitetään lista ominaisuuksista, joiden arvokombinaatio yksilöi luokan yksilöt yksikäsitteisesti. Esimerkiksi henkilön yksilöinti etunimen, sukunimen ja syntymävuoden avulla voidaan ilmaista näin:

```
:Henkilö owl:hasKey ( :etunimi :sukunimi :syntymäaika ) .
```

11.5.2 Tietotyyppiominaisuudet

OWL tarjoaa välineitä paitsi objektiominaisuuksien myös tietotyyppiominaisuuksien määrittelyyn. Tietotyyppien arvoalue voidaan esimerkiksi määritellä rajoituksilla ja toisten tietotyyppien kombinaatioilla vastaavaan tapaan kuin XML Schemassa. Esimerkissä on 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 )
)
```

Arvoalueiden määrittämisessä voidaan käyttää myös joukko-opillisia operaatioita.

11.5.3 Annotointiominaisuudet

Annotointiominaisuuksien (annotation property) avulla voidaan OWL-ontologiassa määriteltäviin entiteetteihin lisätä ihmislukijaa varten niitä kuvaavaa dokumentointia. Esimerkiksi:

annotointiominaisuus

```
AnnotationAssertion( rdfs:comment :Orpo
  "Henkilö, jonka vanhemmat ovat kuolleet."
)
```

Entiteettien lisäksi kommentteja voidaan lisätä myös aksiomiin, kuten:

```
SubClassOf(
  Annotation( rdfs:comment "Naiset ovat henkilöitä." )
  :Nainen
  :Henkilö
)
```

Tämä annotointi vastaa seuraavaa 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 .
```

11.6 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

OWL Full

profiili kuitenkin se, ettei kieli ole loogisesti ratkeava. On siis olemassa OWL-ontologioita, joissa päättelijä ei koskaan päädy lopputulokseen, ja päätely voi olla laskennallisesti hidasta. Ratkeavuus- ja tehokkuusongelmat voidaan ratkaista rajoittamalla kielen muotoja tapauskohtaisesti OWL:n *profileihin* (profile) eli murteisiin, joita ovat OWL EL, OWL QL ja OWL RL. Näiden unionista, joka on laajin tehokkaasti ratkeava OWL-profiili, OWL DL käytetään nimitystä *OWL DL* (Description Logic).

Kukin profiili on suunniteltu käytettäväksi omantyyppisissään ontologioissa, joiden käsittelyyn voidaan toteuttaa tehokkaita päättelykoneita. Samalla kuitenkin joudutaan rajoittamaan laskennallisista syistä kielen ilmaisuvoimaa OWL DL -profiiliin nähden tavoilla, joiden perusteet eivät välttämättä ole kovin intuitiivisia sovellusten kannalta. Seuraavaksi esitetään lyhyet luonnehdinnat OWL:n profileista; näissä käytettävissä olevat OWL-ilmaisut ja rajoitteet on kuvattu tarkemmin W3C:n OWL-spesifikaatiossa.¹

11.6.1 OWL 2 EL

OWL EL soveltuu erityisesti ontologioille, joissa on paljon luokkia. Profiilia käytetään esimerkiksi käsiteltäessä isoja lääketieteellisiä ontologioita, kuten Snomed CT, joiden hierarkioissa voi olla jopa satoja tuhansia luokkia.²

OWL EL -päättelykoneet kykenevät tehokkaasti tarkistamaan ontologian toteutuvuuden (satisfiability), laskemaan luokkasuhteet ja päättelämään yksilöiden kuulumisen luokkiin.³ Tehokkuuden hintana ovat rajoitukset muun muassa negaation, disjunktion ja universaalikvanttorin käytössä, eikä esimerkiksi käänteisiä ominaisuuksia voida käyttää.

11.6.2 OWL 2 QL

OWL QL puolestaan soveltuu sellaisten ontologioiden esittämiseen, joissa on paljon yksilöitä eli dataa. Murteen päättelykone voidaan ohjelmoida tehokkaasti relaatiotietokannan päälle, mikä soveltuu hyvin juuri isojen datajoukkojen käsittelyyn.

¹ <https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/>

² <https://www.snomed.org/snomed-ct/>

³ Päättely voidaan tehdä polynomisessa ajassa, eli laskenta-aika ei kasva hallitsemattomalla tavalla eksponentiaalisesti ontologian koon mukaan.

Tehokkuuden hintana on muun muassa rajoituksia universaalikvanttorin käytössä ja se, että ominaisuusketjuja ei voida käyttää eikä ilmaista resurssien samuutta (equality).

11.6.3 OWL 2 RL

OWL RL:ää käytetään tyypillisesti päättelyn tukemiseen. Profiilia on siksi rajoitettu siten, että kieli voidaan toteuttaa tehokkaasti logiikkaohjelmoinnin avulla sääntöjärjestelmänä.¹

RL-profililla 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 -profilien lisäksi käytössä saattaa vielä olla alkuperäiseen OWL 1 -määrittelyyn kuuluvia profileja: OWL 1 DL on osajoukko laajemmasta OWL 2 DL -profilista, ja OWL Lite on yksinkertainen hierarkioiden käsittelyyn sopiva profiili.

¹ Sääntöihin ja logiikkaohjelmointiin palataan tarkemmin tuonnempana.

12

Päätelysäännöt

Ajattelen, olen siis olemassa.

René Descartes (1596–1650)

En ajattele, en siis ole
olemassa.

*Georg Christoph Lichtenberg
(1742–1799)*

ongelmien ratkonta

Logiikka muodostaa semanttisen webin perustan. Sen avulla määritellään verkon datan ja ontologioiden merkitys täsmällisellä tavalla, jota sekä ihmiset että koneet voivat ymmärtää yksikäsitteisesti. Logiikka mahdollistaa datan automaattisen rikastamisen uutta tietoa pääättelemällä sekä automaattisen *ongelmien ratkonn*an (problem solving).

12.1 Semanttisen webin looginen tulkinta

Semanttisen webin kielten RDF, RDFS ja OWL merkitys määritellään edellisessä luvussa kuvatus klassisen ensimmäisen kertaluvun predikaatilogiikan (predicate logic) avulla. Tämä logiikka muodostaa nykyisten loogisten järjestelmien perustan, ja tietokoneet voivat hyödyntää monia sen yksinkertaistettuja muotoja tehokkaasti päättelytehtävissä.

RDF-verkon looginen tulkinta on suoraviivainen. Jokainen kolmikko

<subject, predicate, object>

esimerkiksi $\langle \text{Suomi, on-osa, Eurooppa} \rangle$, voidaan tulkita tosiasian tai väittämän esittävänä propositiona

predicate(subject, object)

kuten:

on-osa(Suomi, Eurooppa)

RDF-verkko on tällöin joukko samanaikaisesti voimassa olevia väittämiä, jotka muodostavat *tietämyskannan* (knowledge base).

tietämyskanta

W3C on julkaissut RDF- ja RDFS-kielten semantiikan formaalin loogisen spesifikaation.¹ Siinä RDF(S)-rakenteet, kuten `rdfs:subClassOf`, on määritelty *päätelysääntöinä* (entailment rule), joiden periaatteellinen muoto on:

Jos RDF-tietämyskanta E sisältää kolmikot $C_1, \dots, C_n$, niin kolmikko H voidaan lisätä kantaan E uutena tosiasiana.

Taulukossa 12.1 on esimerkkinä esitetty kaksi RDFS-määrittelyn päätelysääntöä, joilla kuvataan luokkahierarkian merkitys päätelyn kannalta.

Sääntö	Jos tietämyskanta E sisältää	niin lisää:
rdfs9	$x \text{ rdf:type } y, y \text{ rdfs:subClassOf } z$	$x \text{ rdf:type } z$
rdfs11	$x \text{ rdfs:subClassOf } y, y \text{ rdfs:subClassOf } z$	$x \text{ rdfs:subClassOf } z$

Taulukko 12.1. Kaksi RDF(S):n semantiikan määrittelyn päätelysääntöä.

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:

¹ <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>


```

:tuoli-1324 rdf:type :tuoli .
:tuoli-1324 rdf:type :istuin .
:tuoli-1324 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.

Päättele voidaan tehdä kahdessa eri vaiheessa:

1. Etukäteen ennen tiedon käyttämistä (preprocessing).
2. Dynaamisesti kyselyn aikana.

Etukäteen tehtävä päättele mahdollistaa tehokkaan ajonaikaisen toiminnan, kun osa työstä on jo tehty. 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/Solr¹, pystyvät kuitenkin käsittelemään tehokkaasti varsin isojakin RDF-verkkoja, joissa voi olla miljardeja kolmikoita, joten päätteilyssä ei tässä mielessä tarvitse yleensä pihistellä.

Päätelmien laskenta kyselyn aikana on joustavampaa, mutta se voi olla laskennallisesti haastavaa erityisesti isommilla tietämyskannoilla. Lisäksi samoja päätelmiä, esimerkiksi yksilön luokkien päätteilyä luokkahierarkian avulla, joudutaan tekemään uudelleen ja uudelleen eri kyselyjen yhteydessä.

Yksinkertaista päätteilyä kyselyn aikana voidaan tehdä paitsi tietämyskantaan 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 päätteilyyn.

¹ <http://lucene.apache.org/>

OWL-ontologiakieli variaatioineen tarjoaa RDF(S)-kieltä rikkaamman välineistön tietämyksen esittämiseen ja päätelmien tekemiseen. OWL-profiilien looginen perusta¹ ovat *kuvailulogiikat* (description logics). Vastaavaan tapaan kuin RDF(S)-kielellä myös OWL:lla on logiikkaan perustuva semantiikka, joka viime kädessä tarkoittaa sitä, että OWL-ontologian avulla voidaan päättelämällä tuottaa uutta tietoa, toisin sanoen uusia kaaria, RDF-tietokantaan.

kuvailulogiikka

OWL:n tyypillinen käyttötapa 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-väittämienä (assertion) (ABox). Luokkamäärittelyt pyritään esittämään yksinkertaisena hierarkiana ja välttämään moniperintää. Loppukäyttöä varten määrittelyjä rikastetaan OWL-päätelykoneen 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.

12.2 Sääntöjen käyttö

Ontologinen päättely RDF(S)- ja OWL-kielissä kohdistuu terminologisten ja ontologisten relaatioiden, kuten yksilö-luokkasuhteiden (`rdf:type`), aliluokkasuhteiden (`rdfs:subClassOf`), aliominaisuuden (`rdfs:subPropertyOf`) ja samuuden päättelyyn. Tällaisten yleisten, sovelluksista riippumattomien suhteiden ohella sovelluksissa on tarvetta käsitellä myös ala- ja sovelluskohtaisia suhteita, kuten vaikkapa kreikkalaisten kuolevaisuutta. Tämä on mahdollista sääntöjärjestelmien avulla.

Ehkä käytetyin looginen järjestelmä sääntöperustaiseen päättelyyn on *Hornin logiikka* (Horn Logic) (HL). HL on predikaattilogiikan osajoukko, joka käsittelee vain tietynlaisia loogisia lauseita, niin sanottuja *Hornin klausuuleja* (Horn clause). Klausuulilla (clause) tarkoitetaan logiikassa *literaalien* (literal) disjunktia

Hornin logiikka

Hornin klausuuli

literaali (logiikassa)

$$C_1 \vee \dots \vee -C_n$$

jossa $\vee$ merkitsee disjunktia ("tai"-operaattori) ja C_i ovat literaaleja.

¹ Hyvä teos tästä on (Hitzler ym., 2010).

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, jonka merkinä seuraavassa on H :

$$\neg C_1 \vee \dots \vee \neg C_n \vee H$$

Tällainen klausuuli voidaan logiikan sääntöjen mukaan esittää implikaationa, jonka muoto on

$$C_1 \wedge \dots \wedge C_n \Rightarrow H$$

eli Hornin klausuulit ovat loogisia sääntöjä.

pää Päätelmää H kutsutaan säännön *pääksi* (head), ja C_i ovat säännön
 ehto *ehtoja* (condition).

Sääntöjen avulla voidaan määritellä uusia n -paikkaisia relaatioita (predikaatteja). RDF-tietomallissa relaatiot ovat binaarisia ja vastaavat verkon kaaria. Esimerkiksi seuraavassa 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)
```

tosiasia Jos Hornin klausuulilla ei ole ehtoja, H on *tosiasia* (fact). Tosiasioden avulla voidaan kuvata sovellusmaailmaan tietty tila, kuten jonkin perheen sukupuu:

```
⇒ mies(Jussi)
⇒ mies(Paavo)
⇒ nainen(Maija)
⇒ nainen(Jaana)
⇒ vanhempi(Jussi,Maija)
⇒ vanhempi(Jussi,Paavo)
...
```

tavoite 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

kysely

$\text{veli}(X,Y) \Rightarrow$

palauttaisi edellisessä 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 kehitetty kokonainen ohjelmoinnin paradigma, *logiikkaohjelmointi* (logic programming).¹ Tunnetuin logiikkaohjelmoinnin kieli on *Prolog*. Logiikkaohjelma koostuu joukosta Hornin 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. Tässä käytetään *käänteistä todistusmenetelmää* (proof by contradiction). Siinä tutkitaan, millä sijoituksilla kyselyn negatio johtaa loogiseen ristiriitaan, mistä voidaan päätellä juuri niiden 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 $\langle \text{subject}, \text{predicate}, \text{object} \rangle$ kuten RDF:n loogisessa spesifikaatiossa. Hornin säännöillä voidaan näin kuvata tietämystä ja myös toteuttaa tarvittavat päättelyt. Prolog-järjestelmiin, kuten SWI-Prolog,² 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 -kehitysympäristöön³ sisältyy mahdollisuus päättelysääntöjen

sääntöjärjestelmä

¹ Logiikkaohjelmoinnin teoriaan voi tutustua esimerkiksi teoksessa (Lloyd, 1987). Hyviä oppikirjoja logiikkaohjelmointiin ovat mm. (Bratko, 2012; Sterling & Shapiro, 1994; Clocksin & Mellish, 2003).

² <http://www.swi-prolog.org/>

³ <https://jena.apache.org/>

esittämiseen ja soveltamiseen ehdoista päätelmiin (forward chaining) tai päätelmistä ehtoihin (backward chaining).

Toinen varsin käytännönläheinen tapa toteuttaa sääntöperustaista päättelyä on käyttää hyväksi SPARQL-kyselyitä. SPARQL Update -kielellä voidaan tunnistaa tietämyskannan rakenteita (SELECT) ja generoida (CONSTRUCT) ja lisätä sen mukaisesti uusia kolmikoita tietämyskantaan. SPARQL-kielen hahmonsovitusta muistuttaa päättelyä, ja kieleen sisältyy päättelyä tukevia ominaisuuksia, kuten ominaisuusketjut. SPARQL-sääntöjen avulla on kehitetty erityinen päättelyjärjestelmä SPARQL Inferencing Notation SPIN, joka on käytössä esimerkiksi TopBraid Composer -ontologiaeditorissa.¹ Sen avulla voidaan muun muassa kontrolloida, miten ja missä järjestyksessä sääntöjä sovelletaan. SPIN-järjestelmään sisältyy tapa esittää SPARQL-kyselyt RDF-muodossa, mikä mahdollistaa niiden tallentamisen ja muokkaamisen kolmikkokannassa. Systeemiin sisältyy mekanismeja uusien funktioiden määrittelemiseksi kyselyissä ja uudelleenkäyttöä varten toisissa säännöissä.

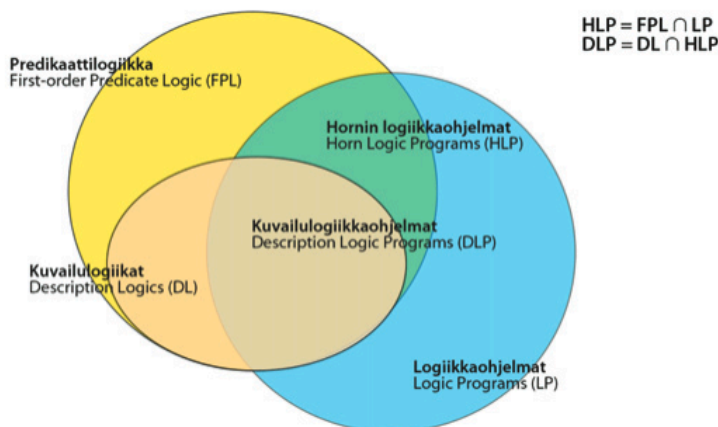
Päättelyä voidaan suorittaa myös OWL-päättelykoneilla. Käytettävissä on lukuisia OWL-päättelykoneita, jotka saavat syötteenä OWL-ontologian ja rikastavat sitä uusilla kolmikoilla. Tällaisia ovat esimerkiksi Pellet, FaCT++ ja HermiT.² Näitä voidaan käyttää vaivattomasti esimerkiksi Protégé-ontologiaeditorissa valmiiksi integroituina toiminnallisuuksina.

12.2.1 Hornin logiikan suhde kuvailulogiikoihin

Hornin logiikka ja logiikkaohjelmointi tuntuivat muodostavan luontevan perustan päättelylle semanttisessa webissä. 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. Seuraavaksi näitä eroja kuvataan hieman tarkemmin.

¹ <http://spinrdf.org/>

² Ks. esim. W3C:n keräämä lista OWL-toteutuksista: <https://www.w3.org/2001/sw/wiki/OWL/Implementations>.



Kuva 12.1. Predikaattilogiikan ja logiikkaohjelmien osa-alueita ja niiden leikkauksia (Antoniou ym., 2012).

OWL-profiilit perustuvat kuvailulogiikoihin (description logic). Ne ovat Hornin logiikan tavoin predikaattilogiikan tehokkaasti laskettavia osajoukkoja, jotka on tarkoitettu ontologiseen terminologiseen päättelyyn käsitelohierarkioiden avulla. Keskeinen kysymys ontologisen päättelyn ja sääntöpohjaisen päättelyn yhdistämisen kannalta on, miten eri kuvailulogiikat suhteutuvat Hornin logiikkaan. Lisäksi haasteena on kysymys siitä, miten OWL-päättelykoneet ja logiikkaohjelmoinnissa käytetyt teoreemantodistimet suhteutuvat toisiinsa. Voidaanko esimerkiksi OWL-päättelykone toteuttaa Hornin logiikan ja logiikkaohjelmoinnin avulla? Tällöin 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 (ks. kuva 12.1). 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 Hornin sääntöjä ei voida aina esittää kuvailulogiikan keinoin.

Kuvailulogiikoita ja Hornin logiikoita voidaan kuitenkin yhdistää eri-

laisilla strategioilla:¹

1. Käytetään vain Hornin klausuuleita. Tällöin kuvailulogiikoista otetaan käyttöön vain ne piirteet, jotka voidaan esittää sääntöinä. Tällaisista logiikoista käytetään nimitystä Description Logic Programs (ks. kuva 12.1). Esimerkiksi OWL RL -profiili voidaan toteuttaa Hornin sääntöjen avulla.
2. Laajennetaan OWL:ia 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 muotoilla ehtoja sille, millaisia sääntöjä OWL-ilmaisujen yhteydessä voidaan käyttää. Haasteena kuitenkin on, että nämä muotoilut eivät ole kovin intuitiivisia käyttää ja laajennukset voivat olla laskennallisesti hankalia.

Ilmaisuvoimaltaan erityisen rikas ehdotus OWL:n ja Hornin logiikan yhdistämiseksi on Semantic Web Rule Language SWRL, jonka säännöt voivat olla muotoa:²

$$A_1, \dots, A_n \Rightarrow B_1, \dots, B_m$$

DL-turvallisuus

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 niin sanottuihin *DL-turvallisiin* (DL-safe) sääntöihin. Sääntö on DL-turvallinen, jos sen kaikki muuttujat esiintyvät jonkin säännön vasemman puolen sellaisen atomilauseen argumenttina, jota ei ole käytetty kuvailulogiikan määrittelyissä.

Vaikka matemaattinen ratkaisu ontologisen ja sääntöperustaisen päätelyn yhdistämiseksi olisi olemassa, OWL-päätelyn ja logiikkaohjelmoinnin yhdistämisessä on vielä lisähaasteita. Logiikkaohjelmoinnissa tehdään perinteisesti kaksi käytännöllistä oletusta, joita OWL-maailmassa ei tehdä: suljetun maailman oletus ja yksikäsitteisten nimien oletus. Seuraavaksi tarkastellaan näitä erikseen.

¹ Ks. esim. (Hitzler ym., 2010).

² <https://www.w3.org/Submission/SWRL/>

12.2.2 Suljetun maailman oletus

Logiikkaohjelmoinnissa ja Prolog-kielessä tehdään yleensä *suljetun maailman oletus* (Closed World Assumption) (CWA). Se tarkoittaa, että väittämät, joita ei tiedetä tai voida todistaa todeksi oletetaan epätoodeksi, toisin sanoen oletetaan, että kaikki tarpeellinen maailman tieto tunnetaan. Tämä vahva ja usein erittäin käyttökelpoinen oletus on tyyppinen logiikkaohjelmoinnin ohella myös perinteisissä tietokantajärjestelmissä. Jos esimerkiksi ei voida päätellä Jussin olevan nainen – hänet esimerkiksi tiedetään mieheksi – tiedetään, että hän ei ole nainen. Ilman suljetun maailman oletusta tätä ei voida tietää, koska voi olla niin, että tiedot Jussista eivät ole täydellisiä.

suljetun maailman
oletus

Suljetun maailman oletus on kuitenkin haasteellinen tilanteissa, joissa maailma ei ole suljettu, vaan tiedot siitä ovat puutteellisia. Tilanne on tällainen avoimessa webissä, jossa tieto on fragmentaalista. Tällöin oletus siitä, että se, mitä ei tiedetä, on epätotta, johtaa virheellisiin päätelmiin. Jos ei esimerkiksi tiedetä, satoiko Pihtiputaalla eilen vai ei ja kysytään ”Satoiko Pihtiputaalla eilen?”, suljetun maailman olettava tekoäly vastaa kieltävästi. *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 on päädytty käyttämään avoimen maailman oletusta.

avoimen maailman
oletus

Monissa tilanteissa tällainen OWL-päätelijän toiminta saattaa kuitenkin tuntua hämmentävältä ja itsestään selvältä tuntuvien asioiden kertominen turhalta. Jos esimerkiksi tiedetään, että Jussi on mies, ei voida 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 pienentyä päättelyn tuloksena. Tällaisia logiikoita kutsutaan *epämonotonisiksi* (non-monotonic).

monotoninen logiikka

epämonotoninen logiikka

12.2.3 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 monikieliselläkin ihmisellä voi olla vain yksi äidinkieli, ja että Hannan äidinkieli on suomi. Jos sitten käy ilmi, että hänen äidinkieltensä 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-päätelyssä 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, jolla vain on kaksi nimeä. OWL-päätelijä tekee siksi päätelmän siitä, että suomen ja ruotsin on oltava sama asia, koska muuten tilanne olisi ristiriitainen. Moinen päätelmä on hämmentävä mutta loogisesti oikein ilman lisätietoja. Jotta OWL-päätelijä ymmärtäisi tilanteen olevan ristiriitainen, sille pitää erikseen kertoa, että suomi ja ruotsi eivät voi olla sama asia.

OWL-ontologioissa joudutaan tämän vuoksi 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 niiden halutaan todella viittaavan eri henkilöihin, eriyä joudutaan jokaisen nimiparin osalta erikseen kertomaan `owl:differentFrom` ominaisuudella, ellei asianlaita ole muuten pääteltävissä. Jos joukossa on n nimeä, joudutaan lisäämään $n * (n - 1) / 2$ kaarta, eli esimerkin tapauksessa 45 kaarta. Käytännössä tämä voidaan onneksi tehdä OWL-kielessä yhdellä `owl:allDifferent`-ilmaisulla, jossa luetellaan kaikki pareittain eri objekteihin 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 yleensä 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 päätel-

lä, viittaavatko kaksi URI-tunnistetta samaan asiaan vai eivät, 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 asioita tarpeettomasti.

12.2.4 Yhteenveto

Yhteenvetona voidaan todeta, että puhtaat loogiset päättelykoneet, kuten OWL-päätelijät, eivät tee 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 tietämyskantaan uusia kolmikoita, mikä rikastaa tietoa semanttisesti ja näkyy sovelluksissa älykkyytenä. Uudet kolmikot voidaan joko lisätä tietämyskantaan etukäteen 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.¹

12.3 Käyttötapauksia säännöille

Sääntöjärjestelmiä voidaan hyödyntää muun muassa seuraavilla tavoilla.

- *Sisällön rikastaminen.* Sovelluskohtaisen *arkipäätelyn* (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*,

arkipäätely

¹ Logiikan ja semanttisten verkkojen historiaa ja vastaavuutta on käsitelty mm. teoksessa (Sowa, 2000). Semanttisen webin loogisiin perusteisiin voi tutustua mm. teoksissa (Antoniou ym., 2012; Hitzler ym., 2010) ja W3C-järjestön standardien dokumentaatioissa *RDF Semantics* <http://www.w3.org/TR/2004/REC-rdfmt-20040210/>, *OWL 2 Web Ontology Language RDF-Based Semantics* <https://www.w3.org/TR/owl2-rdf-based-semantics/> ja *RIF Basic Logic Dialect* <https://www.w3.org/TR/rif-bld/>

jos tämä on pääteltävissä automaattisesti ontologian perusteella.

- *Metadataformaattien muunnokset.* Säännöillä voidaan toteuttaa datan muunnoksia metadataformaattien välillä. Jos esimerkiksi tiedetään Dublin Core -metadatan 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 kokoelmätietoja.
- *Datan yhdistäminen ja linkitys.* Sääntöjen avulla voidaan 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 (1802–1885) kirjoittamia teoksia, voidaan hakea suosituksiksi muita saman maan (Ranska) kirjailijoiden teoksia samalta aikakaudelta. Sääntöjä ketjuttamalla voidaan löytää mutkikkaampiakin yhteyksiä. Sääntöjen etuna on, että sääntöjen tai niiden ketjujen logiikka voidaan ilmaista selväsanaisesti *selityksenä* (explanation) sille, miksi kyseinen suositus on annettu. Tätä ideaa on käytetty aiemmin tekoälyn asiantuntijajärjestelmissä.
- *Aineistojen projisointi faseteille.* Fasettihaussa sääntöjä voidaan käyttää hakukohteiden projisoimiseen eli ”ripustamiseen” hakukategorioihin 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 automaattisen muodostamisen* (knowledge discovery).

selitys

looginen täydellisyys

eheys

Sääntöjä käytettäessä päättelykoneen ei välttämättä tarvitse olla loogisesti *täydellinen* (complete), toisin sanoen kyetä johtamaan kaikki loogiset seuraukset. Hyödyllistä voi olla, jos edes osa mahdollisista päätelmistä voidaan tehdä. Päättelykoneen pitäisi kuitenkin olla *eheä* (sound), eli se ei saisi tehdä virheellisiä päätelmiä.

IV

SOVELLUKSET JA INFRASTRUKTUURI

13

Sovellusten kehittäminen

Kaikki hyvät periaatteet on jo kirjattu. Enää ei tarvitse muuta kuin panna ne käytäntöön.

Blaise Pascal (1623–1662)

Tässä luvussa esitellään linkitetyn datan sovellusten kehittämistä, rakennetta ja toimintaa. Keskeinen ero moniin perinteisiin web-järjestelmiin on sovellusten datalähtöisyys: sovellusten perustana on rajapintojen kautta käytettävä linkitetty data, joka voi olla hajautettu fyysisesti ympäri maailmaa sijoitetuille palvelimille. Datalähtöisyys ja hajautetun heterogeenisen tiedon linkitys asettaa uusia haasteita organisaatioiden sisällöntuotannolle ja keskinäiselle yhteistyölle: semanttisessa webissä tarvitaan aiempaa enemmän yhteisesti sovittuja pelisääntöjä ja aineistojen avointa jakamista datan linkittämiseksi kustannustehokkaasti yli organisaatiorajojen ja *datasiilojen* (data silo).¹ Luvussa esiteltävän julkaisumallin keskiössä onkin yhteinen semanttisen webin tietoinfrastruktuuri, jota Suomessa on kehitetty vuodesta 2003 alkaen.

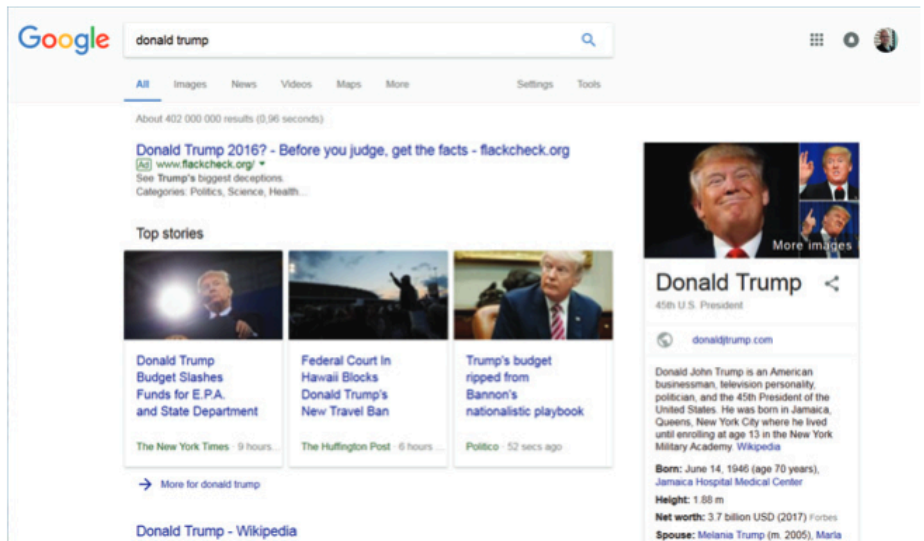
datasiilo

¹ Datasiilo on erillinen tietovarasto, jonka data ei yhdisty muiden tietovarastojen sisältöihin.

13.1 Yleiset ja alakohtaiset sovellukset

entiteettihaku

Linkitetyn datan sovellusten kirjo on yhtä laaja kuin webin sovellusten sateenkaari. Esimerkiksi Googlen hakukone hyödyntää semanttisia merkkauskia ja linkitettyä dataa: sisällön tuottajat voivat lisätä WWW-sivuille Schema.org-spesifikaation¹ mukaista semanttista metatietoa sivujen sisällöstä, jotta hakukone voisi paremmin ymmärtää niiden sisältöä. Tämä mahdollistaa muun muassa *entiteettihau* (entity search), jossa hakukone etsii WWW-sivujen sijasta semanttisen webin entiteettejä, kuten paikkoja, henkilöitä tai elokuvia, 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 13.1 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 muun muassa hänen syntymäsairaalaansa, kouluihin, joita hän kävi, sekä lisätietoon Melanie-vaimosta ja lapsista.



Kuva 13.1. Google hyödyntää linkitettyä dataa entiteettihauksessa sekä tulosten ja lisätietojen esittämisessä. (<http://google.com>.)

¹ <http://schema.org>

Facebook taas käyttää linkitettyä dataa sosiaalisten verkostojen ja niihin liittyvän tiedon esittämiseen. Soveltajat pääsevät käyttämään dataa avoimen Graph API -rajapinnan¹ 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 2012 analysoitiin yli 3 miljardin WWW-sivun satunnainen otos, joista yli 30 prosenttia sisälsi metadataa. Määrä oli kasvanut merkittävästi aiemmista vastaavista testeistä.²

Globaalin käytön ohella linkitettyä dataa hyödynnetään mitä erilaisimmissa alakohtaissa sovelluksissa. Esimerkkejä linkitetyn datan käyttötapauksista ja sovelluksista on koottu muun muassa W3C:n sivuille. Aiheina ovat muun muassa lääketieteeseen ja terveyteen liittyvät sovellukset, kulttuurialan sovellukset, musiikkiala, teollisuus, paikkatieto- ja mobiilijärjestelmät.³ Linkitetyn datan soveltuvuus mitä moninaisempien alojen käyttöön huomattiin myös Suomessa kansallisessa FinnONTO-hankesarjassa (2003–2013) ja sitä seuranneessa Linked Data Finland -kehitystyössä ja siihen liittyvissä hankkeissa. Näissä on kehitetty muun muassa kulttuurialan semanttisia portaaleja (MuseoSuomi.fi, Kulttuurisampo.fi, Kirjasampo.fi ja Sotasampo.fi) ja Terveystieteen ja hyvinvoinnin laitoksen verkkopalveluita (TerveSuomi.fi). Lisäksi on ohjelmoitu demonstraattoreita Wärtsilän voimalaitosten dokumentointiin ja Nokian 5G-verkkojen hallintaan, julkaistu Semanttinen Kalevala kansanrunouden saralla, tutkittu valistuksen ajan kirjeenvaihtoa Euroopassa vuosina 1500–1800 osana laajaa eurooppalaista konsortiota, sovellettu teknologiaa kielikorpusten tutkimuksessa sekä käytetty teknologiaa lainsäädännön ja oikeustapausten julkaisemiseen avoimena datana kehittämällä oikeusministeriön Finlex-palvelusta Semanttinen Finlex.⁴ Kaiken tämän työn taustalla ollut FinnONTO:n laaja-alainen visio yhteisen kansallisen tietoinfrastruktuurin luomisesta osana kansainvälistä semanttisen webin kehittämistä. Työssä on keskitytty eri aloilla tarvittavien onto-

¹ <https://developers.facebook.com/docs/graph-api>

² Ks. (Mika & Potter, 2012).

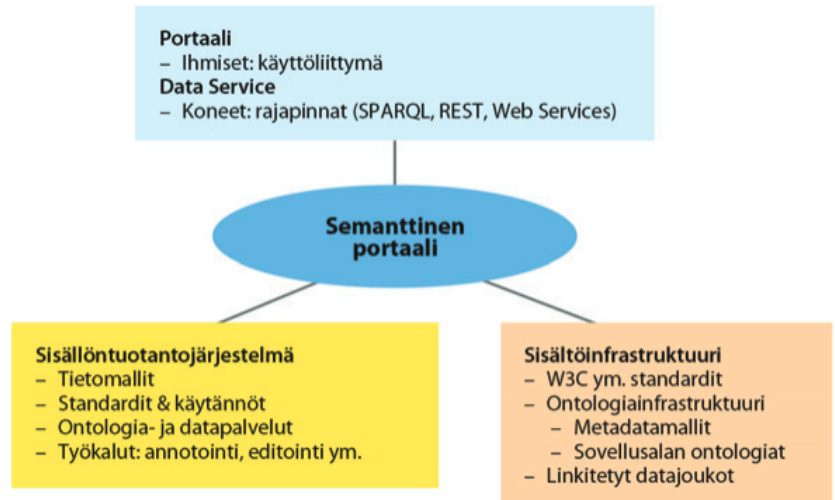
³ <https://www.w3.org/2001/sw/sweo/public/UseCases/>

⁴ Semanttinen Finlex -palvelu julkaistiin vuonna 2016 osoitteessa <http://data.finlex.fi>.

logisten asiasanastojen, auktoriteettitietojen (henkilöt ja organisaatiot), historiallisten tapahtumien, paikkatietojen, biologian taksonomioiden ja lääketieteellisten luokitusten julkaisemiseksi avoimena linkitettyinä datana sekä yhteentoimivien metatietomallien kehittämiseen.¹

Seuraavaksi 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 sekä muiden sovellusten kehittämisestä, ajatukseen ontologiainfrastruktuurin rakentamisesta ja tarjoamisesta avoimina verkkopalveluina asiakaskäyttöön ONKI/Finto-palveluissa sekä datan julkaisemiseen avoimina verkkopalveluina Linked Data Finland -alustalla.

13.2 Semanttisen portaalin osat

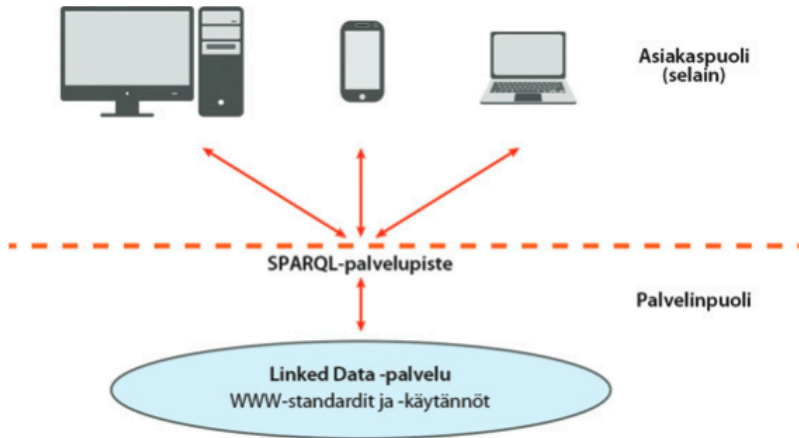


Kuva 13.2. Semanttisen webin portaalin kolme komponenttia.

Linkitetyn datan semanttisissa portaaleissa voidaan erottaa kolme pääosaa (ks. kuva 13.2).

¹ <http://journal.fi/tt/article/view/41559>

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 esittämismuodoista ja tavoista tuottaa tietoa. Sisällöntuotannossa hyödynnetään sovellusalueen standardeja, käytäntöjä ja työvälineitä datan tuottamiseksi alkulähteistä.
3. Koko systeemi käyttää **tietoinfrastruktuuria**, joka perustuu toisaalta 1) W3C:n sovellusalueesta riippumattomiin web-standardeihin ja toisaalta 2) alakohtaisiin metatietomalleihin ja ontologioihin sekä käytettävissä oleviin muihin datajulkaisuihin.



Kuva 13.3. SPARQL-datapalvelu, jota käyttää monta eri sovellusta.

Sovelluksissa on hyödyllistä erottaa selkeästi datapalvelu varsinaisista sovelluksista, jotka käyttävät dataa yleensä SPARQL-rajapinnan kautta. Ideana on, että saman rajapinnan varaan voidaan kehittää useita linkitetyn datan sovelluksia dataa muuttamatta (kuva 13.3). Tämä on kustannustehokasta, sillä datan muokkaaminen voi olla työlästä ja virhealtista puuhaa, koska palveluissa voi olla miljoonia tai jopa miljardeja kolmikoita. Sovelluksista, joiden varsinainen toiminnallisuus syntyy asiakaspuolella selaimessa, käytetään nimitystä *Rich Internet Application* (RIA). RIA-sovellusten etuna on mahdollisuus nopeiden käyttöliittymien

MVC-arkkitehtuuri

malli
näköymä
käsittelijä

kehittämiseen ilman kokonaisten 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, *näköymä* (view) käyttöliittymän ulkoasun ja *käsittelijä* (controller) toteuttaa käyttäjältä saadut käskyt muuttamalla mallia ja näköymää.

Seuraavaksi esitellään kuvan 13.2 kompotenteista ensin semanttisen portaalin idea ja ansaintalogiikka, jossa kaikki voittavat. Sitten käsitellään sisällöntuotantoon liittyviä kysymyksiä keskittyen tapauksiin, joissa tietoineistoina voivat olla painetut tekstit ja rakenteista tietoa sisältävät tietovarastot, kuten tietokannat. Myös äänestä, kuvista ja elokuvista voidaan louhia tietoa ja esittää sitä semanttisesti, mutta tällöin käytettävät hahmontunnistuksen 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.

Portaalimalli yhteisölliseen julkaisemiseen

Yleistykset näyttävät tietä
ajattelun suuriin valtakuntiin.

Ralph Waldo Emerson
(1803–1882)

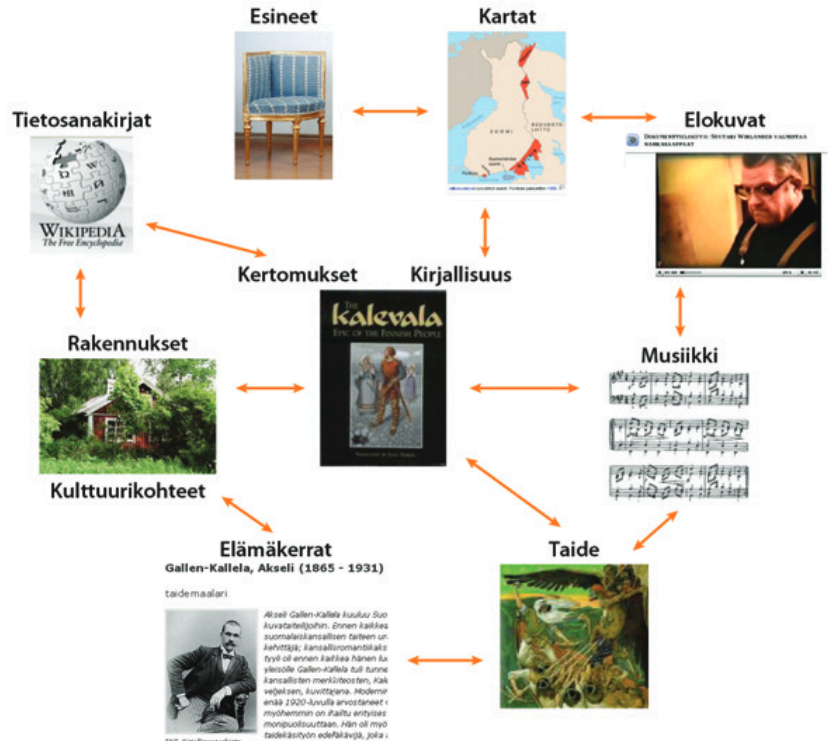
Semanttisen portaalin idean avulla voidaan tarttua yhteisölliseen sisälöntuotantoon ja julkaisemiseen liittyvään keskeiseen haasteeseen: *miten tietoa ja tiedon tuotannon prosesseja voidaan yhdistää yli organisaatio-rajojen ja datasiilojen?* Tässä luvussa tarkastellaan esimerkkinä kulttuurialan sisältöjen julkaisemista kansallisella tasolla verkossa.¹

14.1 Kulttuuriaineistot verkossa

Julkisen sektorin museoilla, kirjastoilla, arkistoilla ja mediaorganisaatioilla, kuten YLE, on paljon yhtenäinen kansallinen tehtävä ja tavoite julkaista kulttuuriaineistoja ja palvella asiakkaita nopeasti muuttuvassa digitaalisessa toimintaympäristössä. Eri organisaatioiden toisiinsa eri

¹ Kulttuuriaineistojen julkaisemista ja käyttöä semanttisessa webissä sekä tässä luvussa esitettävää mallia on esitelty tarkemmin teoksessa (Hyvönen, 2012).

tavoin liittyvien sisältöjen saaminen asiakkaiden ja tutkijoiden käyttöön joustavasti ja kustannustehokkaasti verkon välityksellä on niin tiedon julkaisijoiden kuin asiakkaidenkin yhteinen etu. Lisäksi yhteistyöllä voitaisiin vähentää kustannuksia ja saada käyttöön elintärkeää IT-osaamista ja resursseja, joita erityisesti pienten organisaatioiden on vaikea hankkia. Samalla voitaisiin selkeyttää organisaatioiden työnjakoa, eliminoida tarpeetonta päällekkäistä työtä ja kohdistaa eri organisaatioiden niukkoja resursseja optimaalisemmin niiden omille osaamisalueille.



Kuva 14.1. Kulttuurisisällöt ovat heterogeenisiä ja linkittyvät toisiinsa monin tavoin.

Sisältöjen yhteentoimivuuden ja organisaatioiden toiminnan kannalta tähän visioon liittyy kuitenkin isoja haasteita. Kulttuurialan sisällöt ovat formaateiltaan monimuotoisia, niissä käytetään eri kieliä, tiedot on kuvattu erilaisilla metadatatamalleilla ja käytössä on monia luettelointitapoja. Semanttisena haasteena on, että erilaiset sisällöt liittyvät toisiinsa mutkikkailta tavoilla (ks. kuva 14.1). Esimerkiksi henkilön tekstimuotoi-

nen elämäkerta sisältää viittauksia paikkoihin, sukulaisiin ja kollegoihin, tapahtumiin sekä tieteellisen ja taiteellisen työn tuloksiin, joista kaikista on tyypillisesti lisätietoa jossain muualla, mutta eri muodossa.

URI: http://dbpedia.org/resource/Pyotr_Ilyich_Tchaikovsky



Pjotr Tšaikovski (fi)
 Пётр Ильич Чайковский (ru)
 Pyotr Ilyich Tchaikovsky (en)
 Pjotr Tšajkovskij (sv)
 Pjotr Tšajkovskij (no)
 Pjotr Iljitsch Tschaikowski (de)
 Piotr Ilitch Tchaikovski (fr)
 Piotr Illich Chaikowski (es)
 Pétr Il'ic Čajkovskij (it)
 Pjotr Iljitsj Tšajkovski (nl)
 Piotr Ilitch Tchaikovsky (pt)
 Piotr Czajkowski (pl)
 Piotr Ilici Ceaikovski (ro)
 Pjotr Iljics Csajkovszkij (hu)

Kuva 14.2. Pjotr Iljits Tšaikovskin tunnist DBpediassa ja kirjoitusasuja eri kielillä Getty-säätiön Union List of Artist Names -ontologiassa.

Lisäksi tietojen kohteet identifioidaan eri tavoilla eri maissa, organisaatioissa ja kielissä. Jo pelkästään samojen henkilöiden tunnistaminen nimien perusteella eri tietokannoissa on osoittautunut hankalaksi. Kuvasessa 14.2 on esimerkkinä listattu Getty-säätiön ULAN-rekisterissä käytettyjä erikielisiä kirjoitusasuja Pjotr Iljits Tšaikovskista. Erilaisten kirjoitusasujen lisäksi henkilöistä voidaan käyttää myös erilaisia lyhenneilmaisuja, minkä lisäksi he voivat käyttää pseudonyymejä ja lempinimiä. Nimet voivat myös muuttua vaikkapa avioliiton seurauksena.

Tunnistejärjestelmien ja tietomallien sekamelska on seurausta tavasta, jolla kulttuurisisältöjä tuotetaan toisistaan riippumatta (ks. kuva 14.3). Tiedon esitysmuotoja ja luettelointikäytäntöjä on pyritty yhtenäistämään muun muassa standardointihankkeiden avulla, mutta kokonaisuutena tässä riittää vielä paljon tehtävää.

14.2 Hajautettu haku ja tiedon aggregointi

Eri toimijoiden tuottamia sisältöjä voidaan yhdistää kahdella eri strategialla.



Kuva 14.3. Kulttuurialan sisältöjä tuottavat eri toimijat yleensä toisistaan riippumatta ilman yhteistyötä.

1. **Hajautettu haku.** Hakukone voi yhdistää datan tiedon hakuvaiheessa 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 ainoastaan sopia yhtenäisestä tavasta esittää kyselyitä ja hakutuloksia.

Mekanismi sopii erityisesti tilanteisiin, joissa 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 näkymänsä tietokantaan, johon kysely pitää muuntaa. *Global as View* (GAV) -strategiassa taas sama globaali näkymä jaetaan paikallisille tietokannoille vastattavaksi. Muunnosprosesseissa tarvittavia ohjelmistoja 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 hankalaa.

Ratkaisumalli näihin ongelmiin on tiedon aggregointi eri lähteistä ja haun kohdistaminen näin aikaansaatuun yhteen harmonisoituun tietokantaan. Semanttinen web tarjoaa tässä lähestymistavassa uudenlaisen tavan tarttua samalla kertaa sekä sisällöllisen yhteentoimivuuden että hajautetun sisällöntuotannon haasteisiin (ks. kuva 14.4).

Tämän ratkaisumallin ytimessä ja kuvan 14.4 keskellä on ajatus jaetusta tietoinfrastruktuurista. Sen avulla jo sisällön alkuperäiset tuottajat voivat kukin tahollaan luetteloida ja kuvailla tietoa yhteentoimivassa muodossa. Tällöin yhteisöllinen data voidaan yhdistää automaattisesti laajemmaksi kokonaisuudeksi. Esimerkiksi Ateneumin taidemuseosta saatava tieto Akseli Gallen-Kallelan maalauksesta *Sammon ryöstö* yhdistyy automaattisesti taiteilijan elämäkertaan Suomalaisen Kirjallisuuden Seuran kansallisbiografiassa 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 MuseoSuomi-hankkeessa, jossa kehitettiin pilottina semanttisen webin teknologialla MuseoSuomi-palvelu (2004). Siinä yhdistettiin Kansallismuseon, Espoon kaupunginmuseon ja Lahden kaupunginmuseon

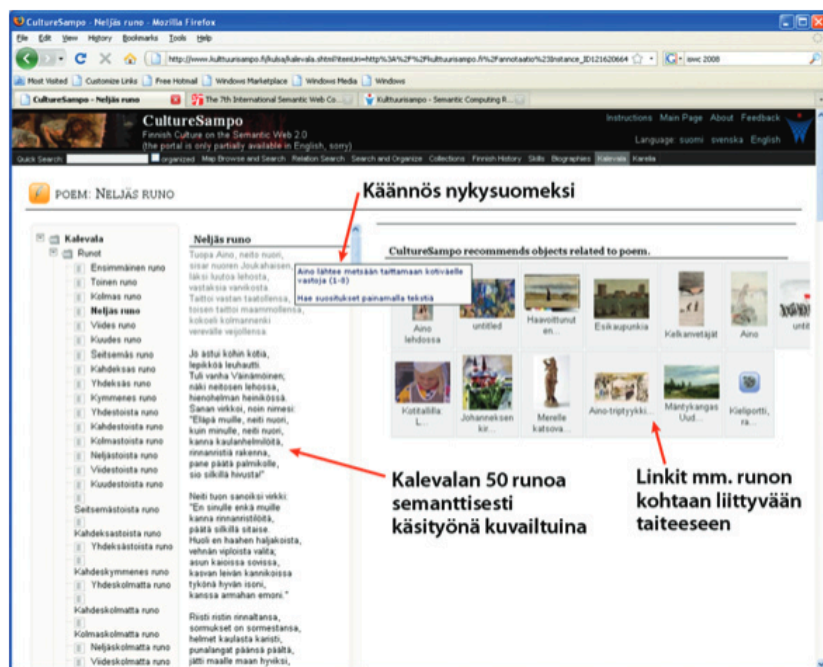


Kuva 14.4. Sisällöntuotannon hajautettu Sampo-malli, jossa tieto yhdistyy jaetun tietoinfrastruktuurin kautta.

kokoelmätietoja semanttiseksi verkoksi, jonka varaan kehitettiin uudenlainen semanttinen hakukone ja logiikkaohjelmointiin perustuva aineistojen suositteleva järjestelmä (semanttinen linkittäjä). 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 vuonna 2003 FinnONTO-hankkeena, joka laajeni sittemmin kymmenvuotiseksi FinnONTO-projektien sarjaksi. Työ jatkuu vieläkin uusina Linked Data Finland -datajulkaisualustaan liittyvinä hankkeina.

Sampo-malli

Kuvan 14.4 Sampo-malliksi ristittyä konseptia on testattu käytännössä useissa Sampo-järjestelmissä:



Kuva 14.5. Kulttuurisammon Semanttinen Kalevala, jossa *Kalevalan* tarinan tapahtumat, henkilöt ja paikat mallinnettiin, esitettiin linkitettyinä datana ja yhdistettiin automaattisesti muun muassa *Kalevala*-aiheiseen taiteeseen Ateneumin kokoelmissa sekä videoihin YLE:n Elävässä arkistossa. (<http://www.kulttuurisampo.fi>)

- *Kulttuurisampo.fi* – *Suomalainen kulttuuri semanttisessa webissä* julkaisiin vuonna 2008. Se yhdistää lähes 30:n eri sisällöntuottajan sisältöjä. Kulttuurisammon datan integraatio perustuu joukkoon yhteisiä ontologioita, kuten kotimaisista asiasanastoista luotu kymmenien tuhansien käsitteiden KOKO-ontologia ja aineistojen perusteella rakennetut paikka- ja henkilöontologiat. Metadatumallit 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 sovellusnäköymää, kuten Semanttinen Kalevala, jota on havainnollistettu kuvan 14.5 kuvankaappauksella sovelluksen käyttöliittymästä. Kulttuurisampo-demonstraattorin semanttisessa verkossa on noin 11,4 miljoonaa

Kulttuurisampo

tietojen välistä yhteyttä (kolmikkoo). Järjestelmää ei ole olennaisesti päivitetty vuoden 2008 jälkeen, mutta se on edelleen verkossa.¹

Kirjasampo

- *Kirjasampo – Suomen kaunokirjallisuus semanttisessa webissä* oli alun perin osa Kulttuurisampoa. Nykyisin se on kuitenkin yleisten kirjastojen (Kirjastot.fi) ylläpitämä oma palvelunsa, jolla oli vuonna 2016 1,6 miljoonaa kävijää. Järjestelmän metadata kattaa käytännössä kaiken Suomessa julkaistun kaunokirjallisuuden erittäin rikkaana semanttisena verkkona. Aineistoja on laajennettu myös tietokirjallisuuteen.²

Matkailusampo

- *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.³

Sotasampo

- *Sotasampo – talvi- ja jatkosota semanttisessa webissä* on Sampo-mallin sovellus, joka yhdistää sotahistoriaan liittyvää dataa lukuisista eri lähteistä linkitetyksi avoimen datan pilveksi. Tätä kirjoitettaessa Sotasammossa on 12 miljoonaa kolmikkoo, kuten tieto siitä, että talvisota päättyi 13.3.1940. Palvelu sisältää muun muassa metatiedot puolustusvoimien SA-kuva-arkiston 160 000 autenttisesti valokuvasta, Kansallisarkiston tietokannan viime sodissa menehtyneestä 95 000 sotilaasta ja tuhansista elossa selvinneistä, tunnetuista sotilaista, 35 000 historiallisen paikan ontologian luovutetuilta alueilta vanhoine karttoineen ja kymmeniä tuhansia sotapäiväkirjoja. Palvelun ytimessä on sota-ajan tapahtumista luotu sotahistoriallinen tapahtumaontologia. Sotasammon tiedot on mallinnettu, harmonisoitu ja yhdistetty laajentaen CIDOC CRM-ontologiaa, jota on esitelty tarkemmin luvussa 8.3. Sotasampoa esitellään tarkemmin luvussa 17.

¹ Kulttuurisampoa on esitelty tarkemmin esimerkiksi julkaisuissa (Hyvönen ym., 2009; Mäkelä ym., 2012).

² Kirjasammon datajoukko on kuvattu tarkemmin artikkelissa (Mäkelä ym., 2013) ja itse järjestelmä julkaisuissa (Mäkelä ym., 2011a).

³ Matkailusammon ideaa ja toteutusta on esitelty artikkelissa (Mäkelä ym., 2011b).

14.3 Edut käyttäjille

Edellä esitetty semanttisen portaalin Sampo-malli tarjoaa loppukäyttäjälle monia etuja:

- *Yksi globaali näkymä hajautettuihin aineistoihin.* Sisältöihin pääsee käsiksi yhden yhtenäisen käyttöliittymän kautta ilman, että tarvitsee opiskella yhdistettyjen tietokantojen erilaisia käyttöliittymiä.
- *Automattinen sisällön aggregointi.* Eri lähteissä oleva tieto kyselyyn liittyen voidaan yhdistää loppukäyttäjälle automaattisesti yhdeksi vastaukseksi.
- *Semanttinen haku.* Haku voidaan tehdä älykkäästi semanttisena hakuna.
- *Semanttinen selailu.* Haun ohella käyttäjälle voidaan tarjota tuloksiin linkittyviä muita tietoja semanttisten suosittelulinkkien avulla.
- *Muut palvelut.* Semanttisen datapalvelun varaan voidaan helposti kehittää muitakin palveluita, kuten data-analyysin työkaluja ja visualisointeja.

14.4 Edut julkaisijoille

Portaalimalli on edullinen myös datan julkaisijoille:

- *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 rikastavat toisiaan automaattisesti ja ilmaiseksi.
- *Aggregoidun sisällön käyttö uudelleen.* Portaalin sisältöä voidaan käyttää helposti uudelleen rajapintojen kautta tai dataa kopioimalla uusissa sovelluksissa, myös ulkoisten toimijoiden kautta.

14.5 Uusia haasteita

Uuden mallin käyttöönotossa on kuitenkin myös omat vaikeutensa. Aineistojen semanttinen yhteentoimivuus edellyttää organisaatioilta aiempaa enemmän yhteistyötä ja sopimista yhteisistä tietomalleista, tunnistuksista ja sisällön tuotannon käytännöistä. Tämä ongelma on toisaalta organisatorinen ja sosiaalinen, sillä perinteisten mallien ja toimintatapojen muuttaminen ei välttämättä ole helppoa. Toisaalta ongelmat voivat olla teknisiä ja käytännöllisiä. Esimerkiksi käytössä olevat tietojärjestelmät eivät välttämättä 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 organisaatioissa ei välttämättä ole tarjolla. Ala on myös nuori, 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 huolimatta linkitetyn datan teknologia on se suunta, johon kulttuuridatan julkaisemisessa ollaan menossa. Esimerkiksi yleiseurooppalaisessa Europeana-järjestelmässä käytetään nykyisin linkitettyä avointa dataa ja siihen perustuvaa tietomallia.¹ Kotimainen vastaava kulttuurialan aineistojen hakupalvelu Finna.fi perustuu tätä kirjoitettaessa perinteisempään teknologiaan.

¹ Europeana-portaalin sovellukset ovat käytettävissä osoitteessa <http://europeana.eu>. Järjestelmän perustana oleva linkitetyn avoimen datan kuvaus on sivulla <https://pro.europeana.eu/page/linked-open-data>. Dataa aggregoidaan Euroopan eri maista, ja mukana on aineistoja myös Suomesta Finna.fi-palvelusta.

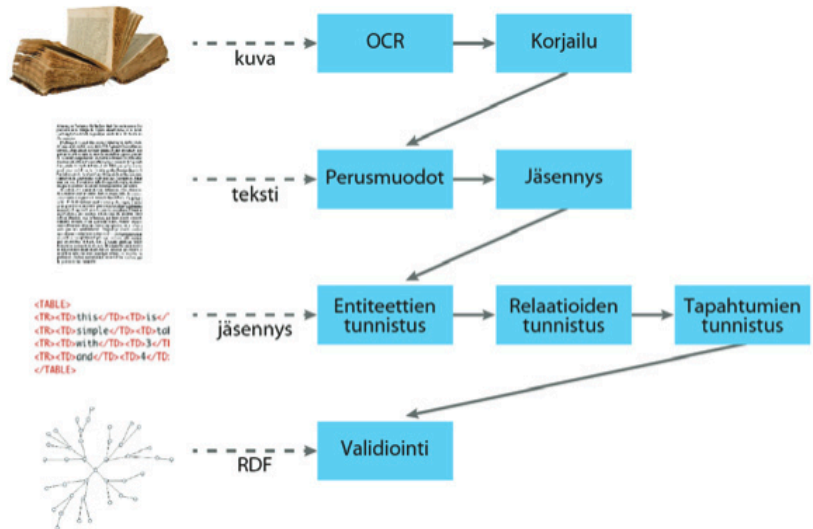
Sisällöntuotanto

Tarvitsee vain yhdistää.

Edward Morgan Forster
(1879–1970)

Tässä luvussa luodaan katsaus RDF-muotoisen metadatan tuotantoon ja sen työvaiheisiin. Lähtökohtana on tekstimuotoinen data. Se kuvaillee sovelluksen kohteena olevia asioita tai tietosisältöjä, jotka voivat liittyä esimerkiksi esineisiin, kirjoihin, äänitteisiin, filmeihin tai tapahtumiin. Kuvaus voi olla rakenteetonta tekstiä, puolirakenteista dataa, kuten WWW-sivuja, tai jo rakenteisessa muodossa esimerkiksi taulukkona tai tietokantakopiona (database dump). Haasteena on muodostaa tällaisesta tekstuaalisesta kuvauksesta rakenteista RDF-muotoista dataa, joka on linkitetty sisäisesti ja jota on rikastettu linkeillä muihin tietoihin. Esimerkkinä käytetään Sotasampo.fi-portaalin aineistoja ja niistä tuotettua dataa.

Kuva 15.1 havainnollistaa prosessia, joka tarvitaan fyysisen tekstidokumentin, kuten kirjan, sisällön muuttamiseksi ensin tekstiksi, joka voidaan sitten analysoida ja muuntaa kieliteknologian avulla rakenteiseksi dokumentiksi ja lopulta linkitetyksi dataksi. Esimerkiksi tekstissä 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. Seuraavaksi tarkastellaan näitä vaiheita hieman tarkemmin.



Kuva 15.1. Sisällöntuotannon vaiheita.

15.1 Tekstin tunnistaminen (OCR)

Monessa tapauksessa tietosisältö on saatavilla vain painetussa muodossa, esimerkiksi kirjana. Sotasammossa tällaisia aineistoja ovat muun muassa *Kansa Taisteli* -lehden PDF-muodossa olevat vuosikerrat sekä Kansallisarkistossa olevat, koneella kirjoitetut joukko-osastokortit.

Tällaisissa tapauksissa datan muodostamisen ensimmäisenä vaiheena on tekstin tunnistus *OCR-tunnistuksella* (Optical Character Recognition), jossa dokumentti digitoidaan kuvaamalla ja siinä olevat tekstit käsitellään hahmontunnistuksen menetelmillä. Tuloksena on tyypillisesti dokumentin tekstit sivukohtaisesti eroteltuina. Joissain tapauksissa myös kuvia voidaan erottaa.

Tekstin irrotukseen on saatavilla runsaasti työkaluja, kuten esimerkiksi avoimen koodin Tesseract¹ ja kaupallinen Abbyy FineReader,² ja moniin yleistyökaluihin, kuten PDF-dokumenttien luomiseen tarkoitettuun Adobe Reader -ohjelmaan, sisältyy OCR-mahdollisuus.³ Kaikkien OCR-työkalujen haasteena on, että OCR-prosessi tuottaa enemmän tai vähemmän virheellisiä tulkintoja painetun tekstin ja algoritmien laadus-

¹ <https://github.com/tesseract-ocr>

² <https://www.abbyy.com/en-eu/finereader/>

³ Aihetta käsitellään esim. teoksessa (Piotrowski, 2012).

ta riippuen. Monet tällaiset virheet ovat systemaattisia, ja niitä voidaan korjata esimerkiksi aineistokohtaisilla säännöillä, joissa virheelliset tekstihahmot korvataan virheettomilla. Esimerkiksi Sotasammossa käytettiin tekstien irrottamiseen digitoiduista *Kansa Taisteli* -lehdistä Abbyy-työkalua. OCR-virheiden korjaamiseen luotiin noin 160 erilaista korjaussääntöä, jotka toteutettiin säännöllisten lausekkeiden avulla.¹ Hankaluutena tässä on varmistaa, että muunnokset kohdistuvat vain todellisiin virheisiin eivätkä vahingossa korjaa oikein tulkittuja tekstikohtia vääriksi. Toisena OCR-tekniikan haasteena on yhdellä sivulla mahdollisesti olevien eri tekstien, sivunumeroiden ja otsikoiden, kuvatekstien, mainosten ja muiden rakenteiden 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. Viime kädessä on yleensä hyväksyttävä se tosiasia, että automaattisesti tuotetuissa aineistoissa on yleensä enemmän virheitä kuin käsityönä toimitetuissa.

15.2 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 lausekkeitä ja muunnossääntöjä. Esimerkiksi Suomalaisen Kirjallisuuden Seuran (SKS) julkaisemasta Kansallisbiografiasta on tuotettu Semanttinen Kansallisbiografia tällaisten muunnosten avulla: elämäkertojen loppuun on kiteytetty määrämuotoisella tavalla tietoa tekstin kohteena olevan henkilön elämänkäären keskeisistä tapahtumista, joiden rakenteistaminen on huomattavasti helpompaa kuin vapaamuotoisen tekstin.² Esimerkiksi Akseli Gallen-Kallelan elämäkerran lopussa lukee ensin:

¹ Työtä on kuvattu tarkemmin tutkimuksessa (Tamper, 2016).

² Semanttinen kansallisbiografia -hanketta on esitelty tarkemmin sivulla <http://seco.cs.aalto.fi/projects/biographies/>, josta löytyy myös aiheeseen liittyviä julkaisuja.

Axel Waldemar Gallén vsta 1907 Akseli Gallen-Kallela S 26.4.1865 Pori, K 7.3.1931 Tukholma. V pankin kassanhoitaja, asianajaja Peter Wilhelm Gallén ja Anna Mathilda Wahlroos. P 1890 – Mary Helena Slöör S 1868, K 1947, PV hovineuvos Kaarlo Aleksanteri (Karl Alexander) Slöör ja Aina Emilia Ehrström. Lapset: Marjatta S 1891, K 1895; Kirsti S 1896, K 1980; Jorma S 1898, K 1939.

Tämän jälkeen luetellaan museoita, joissa on hänen maalaamiaan teoksia:

TEOKSET. Ateneumin taidemuseo, Helsinki; Turun taidemuseo; Gallen-Kallelan museo, Espoo; Gösta Serlachiuksen taidemuseo, Mänttä; Helsingin kaupungin taidemuseo; Riihimäen museo; Pohjanmaan museo, Vaasa; Lahden taidemuseo; Hämeenlinnan taidemuseo; Tampereen taidemuseo; Hiekan taidemuseo, Tampere; Mannerheim-museo, ...

Dataa rakenteistettaessa pitää tunnistaa tekstissä esiintyvät nimet, ammatit, ajat ynnä muut tiedot niiden rakenteen ja ympäristön avulla. Sitten pitää päätellä, mihin käsitteisiin niillä ontologiassa viitataan ja mitä semanttisia suhteita, kuten sukulaisuussuhteita, käsitteiden välille voidaan muodostaa.

automaattinen
annotointi

tietämyksen irrotus

Jos teksti on vapaamuotoista ja rakenteetonta, siitä voidaan kuitenkin yrittää tuottaa merkityksiä *automaattisen annotoinnin* (automatic annotation) menetelmillä. Automaattisen annotoinnin tutkimus kuuluu *tietämyksen irrottamisen* (knowledge extraction) alaan, jossa tutkitaan sekä rakenteettoman että rakenteisen datan muuntamista semanttiseen muotoon.¹ Annotoinnin tavoitteena on löytää tekstistä sen merkitystä kuvaavia termejä (esim. henkilöitä, paikkoja, teoksia jne.), mahdollisesti näiden välisiä suhteita (esim. että tietty henkilö on tietyn teoksen kirjoittaja) ja laajempia tapahtumia (esim. että henkilö sävelsi teoksen tiettyyn aikaan tietyssä paikassa).

NER

Automaattinen annotointi alkaa taivutusmuodot normalisoivalla morfologisella analyysillä, jonka avulla teksti perusmuotoistetaan. Tämän jälkeen voidaan tunnistaa tekstissä sitä kuvailevia, siinä esiintyviä nimiä (Named Entity Recognition, NER) ja avainsanoja. Jos tekstistä tunnistetut käsitteet pystytään yhdistämään järjestelmän taustalla oleviin ontologioihin, voidaan tekstiä ja dokumentin metatietoja alkaa linkittää

¹ Alalla kehitettyjä lukuisia työkaluja on lueteltu mm. Wikipediassa https://en.wikipedia.org/wiki/Knowledge_extraction.

semanttisesti ja rikastaa niitä muiden aineistojen avulla.

15.3 Merkitysten erottelu

Haasteena kuitenkin on taivutusmuotojen ja sanojen merkitysten homonymia, esimerkiksi se, onko sanan ”alusta” perusmuoto ”alku”, ”alusta”, ”alustaa” vai ”alus”. Lisäksi sama termi voi merkitä erikseen tarkasteltuna 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 yleensä oletusarvoisesti Ranskan pääkaupunkia. Jos puheena kuitenkin on Wim Wendersin elämä, tilanne on toinen ohjaajan vuonna 1984 valmistuneen elokuvan *Paris, Texas* vuoksi. Lisäksi paikannimi Paris voi mennä sekaisin kuuluisan miljoonaperijättären Paris Hiltonin nimen kanssa. Kauhuskenaaariona on teksti, jossa Paris Hilton katsoo Pariisin Hiltonissa elokuvaa *Paris, Texas*.

Merkitysten erotteluun eli *semanttiseen disambigointiin* (semantic disambiguation) on kehitetty paljon menetelmiä ja työkaluja erityisesti englannin kielelle, kuten vaikkapa DBpedia Spotlight¹ DBpedian käsitteiden tunnistamiseen tekstistä. Suomenkielisiin teksteihin ne eivät yleensä sovellu.²

semanttinen
disambigointi

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 on ”kapteeni Korhonen” tai että kuvan on ottanut ”Ojanen”. Sitä, kenestä rintamalla olleista sadoista korhosista tai ojasista on kyse, ei välttämättä ole helppoa selvittää.

Toinen automaattiseen annotointiin 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 kertoa, että kyse on jääkiekosta. Tällaisten impli-

¹ <http://www.dbpedia-spotlight.org/>

² Esimerkiksi Sotasammonsa käytettiin entiteettien linkitykseen tutkimusryhmässä itse kehitettyä työkalua ARPA (Mäkelä, 2014) ja disambigointiin erilaisia alakohdaisia heuristiikkoja ja sääntöjä (Heino ym., 2017).

siittisten käsitteiden löytämiseen voidaan käyttää taustalla olevien ontologioiden yhteyksiä, jolloin puhutaan dokumenttien laajentamisesta. Erimerkiksi Suomesta kertova artikkeli voi kertoa jotain myös Pohjoismaista. Laajennus voi parantaa tiedon löydettävyyttä, mutta ei välttämättä. Esimerkiksi haettaessa tietoa hakusanalla ”Pohjoismaat” kiinnostuksen kohteena voi olla vain Pohjoismaat kokonaisuutena, eivätkä Pohjoismaihin kuuluvat valtiot erikseen, jolloin haun tarkkuus huononee, vaikka saanti paranisikin.

aiheiden tunnistus

Tekstien aiheiden löytämiseen voidaan käyttää myös tilastollista tiedonlouhintaa, jossa samantyyppisten tekstien voidaan olettaa puhuvan samoista asioista, vaikka joissakin niistä eivät kaikki yhteiset termit esiintyisikään. Esimerkki tällaisesta tekniikasta on *aiheiden tunnistus* (topic modeling).¹

RDF-datan tuotannon kannalta ongelmattominta tietoa on data, jonka kuvailussa on käytetty standardoituja muotoja tai jopa yksilöiviä tunnisteita, esimerkiksi siten, 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 erilaisia käytäntöjä vaikkapa päiväysten esittämisessä harmonisoidaan. Historiallisten päiväysten osalta lisäharmia tuottavat eri maissa eri aikoina käytetyt erilaiset kalenterijärjestelmät. Esimerkiksi Englannissa ja Venäjällä luovuttiin juliaanista ajanlaskusta myöhemmin kuin Italiassa tai Suomessa. Erityisesti XML- ja taulukkomuotoisen datan puhdistamista ja muunnoksia varten on kehitetty monia näppäriä työkaluja, kuten Open Refine² ja Karma.³

15.4 Datan semanttinen validointi

datan validointi

Kun data on saatu teknisesti haluttuun muotoon, on vielä syytä tarkastaa, ettei datassa ole virheitä tai ettei niitä ole syntynyt sen muunnoksessa, eli data pitää *validoida* (validate). Metadataskeemoissa esimerkiksi usein määritellään jokin tietty elementti pakolliseksi tai rajoitteita kentän arvoille. Datan automaattista validointia varten metadatatamalli

¹ Johdatus aiheiden automaattiseen tunnistamiseen on esim. (Brett, 2012).

² <http://openrefine.org/>

³ <http://usc-isi-i2.github.io/karma/>

voidaan määritellä esimerkiksi käyttämällä W3C:n SHACL-suositusta.¹

Teknisen validoinnin ohella dataa on mahdollista validoida myös *tietämysperustaisesti* sovellusalueen tietämyksen avulla (knowledge-based validation). Esimerkiksi Sotasammossa havaittiin, että järjestelmään sisältyneessä, viime sodissa kaatuneiden tietokannassa Menehtyneet 1939–1945 oli henkilöitä, jotka oli merkitty haavoittuneiksi kuolemansa jälkeen alkuperäisessä datassa olevien kirjoitus- tai muiden virheiden vuoksi.

tietämysperustainen
validointi

Isoista aineistoista ei ole mahdollista tarkistaa kaikkia automaattisesti tuotettuja linkityksiä ja rakenteita. Voidaan kuitenkin tehdä pistokokeita ja arvioida tällä tavalla tilastollisesti linkityksen onnistumista laajemminkin.

¹ <https://www.w3.org/TR/shacl/>

16

Webin tietoinfrastruktuurit

Intellektuellit ratkovat
ongelmia, nerot estävät niiden
syntymisen.

Albert Einstein (1879–1955)

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.

16.1 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 yhteyksiä hieman vastaavaan tapaan kuin vaikkapa tie- tai puhelinverkko.¹ Viona on, että jos yhteisen infrastruktuurin pelisäännöillä tuotettu sisältö julkaistaan verkossa, se voidaan kytkeä automaattisesti muiden toimijoiden semanttisiin verkoihin, jolloin sitä voidaan hyödyntää ikään kuin uutta maantienpätkää tieverkostossa. Etuna on, että verkon muu sisältö lisää uuden sisällön arvoa ”ilmaiseksi” vastaavasti kuin uuden tieosuuden arvo syntyy sen kytkeytymisestä muihin teihin. Samalla myös muiden verkon julkaisijoiden sisältöjen arvo rikastuu uuden tiedonsirpaleen avulla samalla tavoin kuin uusi tie parantaa muiden teiden keskinäistä saavutettavuutta. Sekä itse infrastruktuuri että siinä jo oleva tieto voidaan hyödyntää toisissa soveluksissa, mikä säästää merkittävästi järjestelmien kehityskustannuksia.

Tietoinfrastruktuurissa voidaan erottaa seuraavat osat:

1. **Sovellusriippumattomat W3C:n semanttisen webin standardit** (RDF, RDFS, SPARQL, SKOS, OWL, RIF ym.), joita voi käyttää eri alojen tietämyksen esittämiseen.
2. **Metadatatmallit** ja niiden yhteentoimiva kokonaisuus. Käytössä ovat esimerkiksi tässä teoksessa edellä 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

aiheontologia

aiheontologia. Kansalliseen käsitteistöön perustuvia ontologioita, joilla on linkkejä myös kansainvälisiin sanastoihin ja tietosisältöihin. Aiheontologioiden resursseja käytetään metadatatmalleissa tietojen kuvailussa.

Semanttisen webin standardeja on esitelty jo tämän teoksen muissa luvuissa. Työtä koordinoi kansainvälisesti ja aktiivisesti W3C. 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

¹ Tätä ajatusta on kehitelty artikkelissa (Hyvönen ym., 2007).

Institutions). Lisäksi lukuisilla muillakin aloilla ja yhteisöillä on omia vastaavantyyppisiä datamalleja ja käytäntöjä vaikkapa potilastietoja tai luontohavaintoja varten.

Myös työ aiheontologioiden kehittämiseksi edistyy. Jo Carl von Linnén (1707–1778) ajoista lähtien esimerkiksi biologit ovat ryhmitelleet kasveja ja eläimiä hierarkkiseksi taksonomioiksi, elämän puuksi, ja lajien kehitystä aika-akselilla kuvaaviksi fylogeneettisiksi evoluutiopuiksi. Lääketieteen saralla on kehitetty laajoja ontologioita muun muassa anatomiaan, lääkeaineisiin ja tautiluokitukseen liittyen. Esimerkkejä erilaisista luokituksista ja sanastoista löytyy kaikilta aloilta. Seuraavaksi kuvataan hie- man tarkemmin kulttuurisisältöjen esittämisessä tarvittavia keskeisimpiä ontologioita. Näitä voidaan ryhmitellä seuraavasti:

1. **Yleiskäsiteontologiat.** Nämä ontologiat vastaavat käsitteistöltään karkeasti nykyisiä asiasanastoja ja tesauroksia (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 niin sanottuja auktoriteettitietokantoja. Toimijaontologian avulla samannimiset toimijat voidaan yksilöidä ja erottaa toisistaan eri tunnisteiden ja lisätietojen avulla, kuten henkilökaimat syntymävuoden ja -paikan avulla. Esimerkiksi Getty-säätiön Union List of Artist Names (ULAN) -sanastossa on noin 120 000 toimijaa (kuten Akseli Gallen-Kallela), joilla on noin 293 000 erilaista nimeä. Semanttisen kuvailun osalta toimijoita on luokiteltu muun muassa 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 tms.). Historiallisten paikkojen ontologioiden osalta erityisenä haasteena ovat paikoissa ja kartoilla tapahtuvat muutokset.¹
4. **Aikaontologiat.** Monella alalla ajan esittäminen on tärkeässä roolissa. Lineaarisen kalenteriajan ohella voidaan viitata myös päivän

¹ Lisätietoa Suomen historiallisten paikkojen ja karttojen kuvaamisesta ontologiana on mm. projektisivulla <http://seco.cs.aalto.fi/projects/histoplaces/>.

ja vuodenvaihteen aikoihin (esim. aamu tai kevät) sekä nimettyihin aika- ja tyylikausiin (esim. rautakausi, valistuksen aika tai art deco). Aikakausien, esimerkiksi rautakauden, merkitys voi olla riippuvainen paikasta.

5. **Tapahtumaontologiat.** Tapahtumat, kuten Porvoon valtiopäivät tai Napoleonin kruunajaiset, liittävät toisiinsa henkilöitä, paikkoja, aikoja ja toisia tapahtumia ja mahdollistavat kulttuuristen ilmiöiden kuvaamisen semanttisesti yhteentoimivalla tavalla. Myös kokoelmien esineiden ja muiden kohteiden valmistus-, käyttö- ja omistushistoriaa kuvaavassa *provenienssiedossa* (provenance) ja kokoelmien hallintaan liittyvässä metadatatassa tapahtumat, kuten hankinnat, näyttelyt tai konservointi, ovat tärkeässä asemassa.¹ Tämän vuoksi muun muassa 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, lääkeaineet sekä kielitieteen kielet. Esimerkiksi FinnONTO:n YSO-ontologian yleiskäsité ”linnut” laajenee kattavaksi, yli 11 000 maailman lintulajin taksonomiaksi (lajilistaksi) AVIO-ontologian² kautta.

proveniensi

Lisäksi tarvitaan verkkopalveluita, joiden avulla tietoinfrastruktuurin osat voidaan julkaista ja tukea niiden käyttöä asiakkaiden järjestelmissä:

- **Ontologiapalvelut.** Ontologiapalveluiden kautta ajantasaiset ontologiat voidaan ottaa kustannustehokkaasti käyttöön eri organisaatioissa verkkopalveluina esimerkiksi kokoelmien sisältöjä kuvailemalla. Ontologiapalvelu voi olla tarkoitettu ihmisen käytettäväksi sähköisenä tesauruksena mutta myös sovellusten, kuten luetteloin-

¹ Suomessa puhutaan provenienssiedosta, mutta englannissa käytössä on termi *provenance*, ja *provenience*-termi viittaa yleensä rajatun arkeologiassa ja paleontologiassa kohteen löytöpaikan tietoihin.

² FinnONTO-hankkeessa kehitettiin Luonnontieteellisen keskusmuseon erilaisista taksonomioista RDF-muotoisia ontologioita, kuten lintulajien AVIO-ontologia (Tuominen ym., 2013), nisäkkäiden MAMO-ontologia ja suomalaisten kasvien KASSU. Tätä työtä ja siihen liittyviä sovelluksia on esitelty sivulla <http://seco.cs.aalto.fi/ontologies/biology/>.

tijärjestelmien, hyödynnettäväksi rajapintapalvelun (API) kautta. Palveluun voi sisältyä myös yhteisöllinen komponentti, jolloin palvelun käyttäjät voivat tehdä korjaus- ja lisäysehdoituksia uusista ontologisista käsitteistä ontologian kehittämiseksi.¹

- **Datapalvelut.** Linkitetyn datan palveluiden kautta voidaan julkaista ajantasaista linkitettyä dataa ja ottaa sitä kustannustehokkaasti käyttöön rajapintojen avulla.
- **Infrastruktuurin ylläpitoprosessi.** Lisäksi tarvitaan systemaattinen ja koordinoitu organisaatio ja mekanismi, joka kantaa vastuun eri aloilla tarvittavien sanastojen ja datan yhteisöllisestä kehittämisestä.

16.2 Ontologiapalvelut

Suomessa semanttisen webin ontologioiden ja ontologiapalveluiden systemaattinen kehittäminen aloitettiin vuonna 2003 kansallisessa FinnONTO-hankesarjassa (2003–2012).² Työn tuloksena syntyi muun muassa Kansalliskirjaston Yleisestä suomalaisesta asiasanastosta (YSA) ja 14 muusta eri aloilla käytetystä asiasanastosta muokattu KOKO-ontologia, joka sisältää tätä kirjoitettaessa 53 000 käsitettä.³ Ontologiat julkaistiin sekä avoimena datana että erityisenä ontologiapalveluna ONKI.fi, joka voitiin ottaa rajapintojen avulla käyttöön muun muassa museoiden luettelointijärjestelmissä.⁴ ONKI.fi-palvelu julkaistiin verkossa vuonna 2008, ja sillä oli pian noin 14 000 ihmiskäyttäjää kuukaudessa. Palvelun rajapintoja rekisteröityi käyttämään yli 400 eri tahoa. Tärkeitä käyttötapauksia olivat luetteloinnin ja asiasanoituksen tukeminen ontologiaperustaisen metadatan tuotannossa sekä kyselyiden laajentaminen museoiden kokoelmien tietojärjestelmissä. Järjestelmän ytimessä ovat yleiskäsiteontologiat, joita muut ontologiat eri tavoin tarkentavat. Kansalliskirjasto tuotti vuonna 2014 ONKI:n keskeisiä osia Finto-

¹ Maailmalla kehitettyjä erilaisia ontologiapalveluita esitellään ja vertaillaan artikkelissa (Noy & d'Aquin, 2012).

² FinnONTO:n kansallinen ontologinen visio on julkaistu ja selostettu tarkemmin artikkelissa (Hyvönen ym., 2008).

³ Asiasanastojen muuttamista ontologiaksi on käsitelty suomeksi julkaisussa (Seppälä & Hyvönen, 2014): <https://www.doria.fi/handle/10024/96825>.

⁴ ONKI-konseptin ideaa ja sen toteutusta on esitelty tarkemmin kotisivulla <https://seco.cs.aalto.fi/services/onki/>. Itse palvelu on käytettävissä osoitteessa <http://onki.fi>.

palveluksi, josta on tullut pysyvästi ylläpidetty kansallinen sanastopalvelu.¹ Seuraavaksi esitellään lyhyesti FinnONTO:n ontologiatyön tuloksia.

Kehitystyön lähtökohdaksi otettiin maassamme jo käytössä olevat asiasanastot. Niiden käytöllä voidaan parantaa ja yhdenmukaistaa tiedon indeksointia, jolloin tiedon haussa päästään parempaan tarkkuuteen (precision) ja saantiin (recall). Perinteisiin asiasanoihin perustuvaan tekstihakuun liittyy kuitenkin suuria hankaluuksia, kuten luvussa 3.2 osoitettiin. Esimerkiksi ravintoloita haettaessa ei löydy pizzerioita. Myös kielimuurit ovat perinteisen merkkijonohaun haasteena.

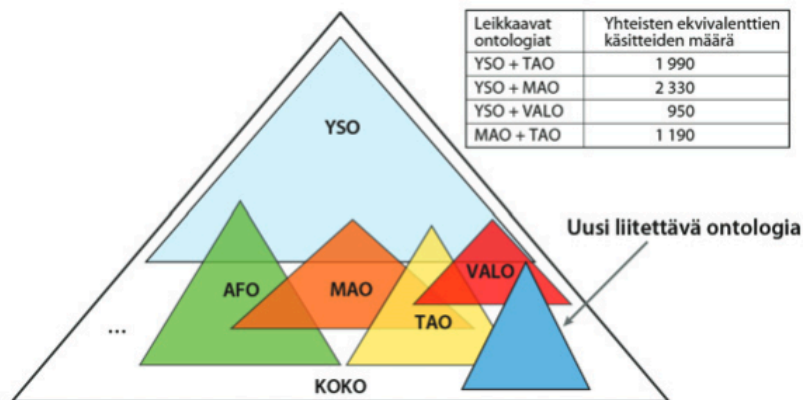
Ontologiaperustainen hakupalvelu sen sijaan ymmärtää paremmin käyttäjiäänsä, koska haun apuna käytetään maailman käsitteitä kuvaavia ontologioita. Ne voivat kertoa, että esimerkiksi ”Kallio” on osa Helsinkiä ja että ”Kallio” tässä yhteydessä ei tarkoita luonnonmuodostelmaa tai presidentti Kyösti Kalliota. Ontologisten viittausten (URI) 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 yleiskäsitettyä eri aloilta. Sanaston laajentaminen oli edelleen tarpeen, mutta kaikkien eri alojen asiantuntemuksen saaminen sanastotyöhön oli haasteellista, vaikka ylläpitoa varten oli perustettu monialainen työryhmä. Samaan aikaan maassamme oli käytössä ja kehitettiin lukuisia erityis-sanastoja, jotka olivat YSA:n kanssa erityisesti yleisempien käsitteiden osalta päällekkäisiä, mikä tuntui usein tarpeettomalta moninkertaiselta työltä. Lisähaasteena oli, että samannimisillä käsitteillä saattoi olla eri sanastoissa erilainen merkitys, mikä aiheutti sekaannuksia tietoja yhdistettäessä. Merkitseekö esimerkiksi ”johtaminen” organisaation johtamista, sähkövirran kulkua, matemaattista menetelmää vai reittiä johonkin paikkaan?

FinnONTO tarttui näihin sisällöllisiin ja organisatorisiin haasteisiin ehdottamalla kansallisesti koordinoitua mallia, jossa sanastotyö voitaisiin yhteistuumiin jakaa eri alojen sanastojen kehittäjäryhmien kesken ja

¹ Tuotteistusprosessi on kuvattu julkaisussa (Suominen ym., 2017), ja Finto-palvelu on käytettävissä osoitteessa <http://finto.fi>. Palvelun nimi on johdettu FinnONTO-projektista.

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.¹



Kuva 16.1. Yhteisöllinen kokonaisontologia KOKO koostuu yläontologiasta YSO ja sitä tarkentavista alaontologioista. Ontologioiden keskinäiset leikkaukset kuvassa ovat vain viitteellisiä ja tarkoitettu yleiskuvan havainnollistukseksi. Järjestelmään voidaan liittää hankkeessa kehitetyllä menetelmällä uusia asiakaskohtaisia ontologioita.

Työn tuloksena syntyi prototyyppi KOKO-ontologiasta, joka on eräänlainen eri alojen ontologioiden muodostama linkitetyn datan ontologiapilvi. KOKO:a on havainnollistettu kuvassa 16.1. Sen ytimenä on YSA:sta kehitetty Yleinen suomalainen ontologia YSO, joka muodostaa KOKO:n *yläontologian* (upper ontology) sisältäen merkitysalaltaan laajimmat käsittehierarkian käsitteet. Erikoisalojen tarkempi käsitteistö ikään kuin ripustuu YSO:n eri haaroihin hierarkioita syventäen.

yläontologia

KOKO-ontologiapilveen kuului alkuvaiheessa (vuonna 2013) taulukossa 16.1 luetellut 15 ontologiaa. Ontologiat linkitettiin toisiinsa ensin sellaisinaan, koska ne perustuivat eri aloilla jo käytössä oleviin asiasanastoihin. Seuraavaksi vuorossa on KOKO-ontologiapilven rakenteen muokaus tarkoituksenmukaisempaan muotoon muun muassa tarpeettomia päällekkäisyyksiä poistamalla, pilven eri osien ylläpitoon liittyvästä työjaosta sopiminen ja rakenteen laajentaminen ja linkittäminen muihin ontologioihin. Työ on jatkunut Kansalliskirjaston koordinoimana Finto-

¹ Tätä FinnONTO-hankkeessa tehtyä laajaa työtä eri alojen asiasanastojen muuttamiseksi ontologioiksi on käsitelty tarkemmin raportissa (Seppälä & Hyvönen, 2014).

Taulukko 16.1. KOKO-ontologiapilven alkuperäiset osaontologiat.

Ontologia	Käsitteitä	Asiasanasto
Yleinen suomalainen ontologia (YSO)	n. 26 000	Yleinen suomalainen asiasanas- to
Julkishallinnon ontologia (JUHO)	n. 6 350	Valtioneuvoston asiasanasto
Kaunokirjallisuuden ontologia (KAUNO)	n. 5 100	Kaunokki-asiasanasto
Kielitieteen ontologia (KTO)	n. 950	Uralistiikan tutkimuksen biblio- grafian asiasanaluettelo
Kirjallisuudentutkimuksen ontologia (KITO)	n. 850	Kirjallisuudentutkimuksen asia- sanasto
Kulttuurien tutkimuksen ontologia (KULO)	n. 1 500	Kulttuurien tutkimuksen asia- sanasto
Liiketoimintaontologia (LIITO)	n. 3 400	Yrityssuomi.fi-portaalin käsit- teistö
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 asiasa- nasto ja Museoalan asiasanasto
Terveyden ja hyvinvoinnin ontologia (TERO)	n. 6 500	TESA, Stameta-asiasanasto, ja n. 2500 MeSH-käsitettä (Me- dical Subject Headings)
Valokuvausalan ontologia (VALO)	n. 2 000	Valokuvan asiasanasto
Viikin tiedekirjaston ontologia (AFO)	n. 6 000	Agriforest-asiasanasto

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 kehittäjien vastuulle.

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 linkittämisen 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 muun muassa päättelyn.
- Laajempi/suppeampi termi -suhteiden tarkentaminen. Tällä tarkoitetaan asiasanastoissa käytettyjen laajempi termi/suppeampi termi (LT/ST) -suhteiden tarkentamista yläluokkasuhteiksi (`rdfs:subClassOf`), 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 periyty-

misessä ja kyselyn laajentamisessa.

- Monimerkityksisten asiasanojen merkitysten erottelu. Tämä tarkoittaa asiasanastojen monimerkityksisten käsitteiden merkitysten jakamista 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 ovat peilejä ja toisaalta peilit ovat huonekaluja, mikä kuulostaa järkevältä. Transitiivisuus molempien suhteiden ylitse ei kuitenkaan toimi, sillä taskupeilit eivät ole huonekaluja. Jos LT-suhteet muutetaan mekaanisesti `rdfs:subClassOf`-luokkasuhteiksi, seurauksena on, että haettaessa huonekaluja hakutulokseen voi tulla virheellisesti mukaan taskupeilejä. Transitiivisuuden tarkistaminen on erityisen haastavaa KOKO-ontologiapilvessä, kun luokkahierarkiat kulkevat eri ontologioiden kautta. Asiasana ”taskupeilit” löytyy Museoalan asiasanastosta MASA muttei YSA:sta.

Esimerkiksi asiasanojen monimerkityksisyyden haasteista sopii YSA:n asiasana ”lapset”, joka voi tarkoittaa muun muassa ikäluokkaa tai perhesuhdetta. Haasteena on, että aikuisetkin ovat omien vanhempiansa lapsia. Siksi YSO-ontologiassa lapset-käsitteestä erotettiin merkitykset ”lapset (ikään liittyvä rooli)” ja ”lapset (perheenjäsenet)”. Mikäli erotte-
lua ei tehtäisi, saattaisi tietojärjestelmä esimerkiksi suositella aikuisille tarpeettomasti lastenkirjoja luettavaksi, koska nämä ovat vanhempiansa lapsia.

FinnONTO-hankkeessa neuvoteltiin lupa julkaista kaikki KOKO-ontologiat avoimena datana. Tulosten hyödyntäminen myös kaupallisiin tarkoituksiin on vapaata, ja ainoastaan aineiston kehittäjätaho on mainittava (CC-BY-lisenssi). Edellytykset työn jatkamiselle ja tulosten hyödyntämiselle ovat tältä osalta paremmat kuin hankkeen alkaessa.

16.3 Työn arviointia

Ontologiatyö tai jo pelkästään insinöörien filosofiasta uusiokäyttöön otama termi ontologia, on herättänyt myös vastustusta. Kritiikkiä on synnyttänyt ontologisoinnin yhteydessä syntyvä yhä tarkempi jaottelu merkityksiin, mikä johtaa yhä isompaan ja mutkikkaampaan käsitteierariaan. 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 kaikki lapsiin liittyvät merkitykset?

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*¹ päätyi luomaan yhden laajan palveluontologian sijasta yleisemmän metatietomallin, jonka metatietokentille voidaan valita arvoja eri ontologioista ilman, että näiden eri kombinaatioille tarvitsee luoda suuria määriä omia käsitteitä.

Ehkä keskeisin periaatteellinen kysymys ontologiatyössä on, miten tarkasti maailmaa voidaan ja on tarkoituksenmukaista kuvata loogisena rakenteena. Maailma ja siihen liittyvä tieto on esimerkiksi monin tavoin epävarmaa (uncertainty), puutteellista (incomplete) ja sumeaa (fuzzy). Tietämyksen loogisen kuvaamisen rajoihin törmättiin jo kerran 1980-luvulla tekoälytutkimuksessa, kun suurta innostusta herättäneiden, ihmiselle vaikeita ongelmia ratkovien *asiantuntijajärjestelmien* (expert system) kehittäminen osoittautui lopulta arvioitua hankalammaksi ja päädyttiin niin sanottuun ”tekoälyn talveen” (AI winter). Semanttisen webin

epävarmuus
puutteellisuus
sumeus

asiantuntijajärjestelmä

¹ <http://www.jhs-suositukset.fi/suomi/jhs183>

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 ylipäättään ole virheetön, mutta se on silti hyödyllinen. On kuitenkin selvää, että täsmälliseen tietoon perustuva logiikka asettaa rajoitteita monimuotoisen ja epätäsmällisen maailman kuvaamiselle ja että yhä automaattisemmin menetelmin ja yhteisöllisemmin tuotetun yhdistetyn datan laatu on tulevaisuudessa semanttisen webin keskeinen haaste.

Hankaluuksista 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 Terveystietokeskuksen ja hyvinvoinnin laitoksen TerveSuomi-portaalissa, Kulttuurisampo-palvelussa, työ- ja elinkeinoministeriön YritysSuomi-palvelussa, oikeusministeriön Semanttisessa Finlexissä, monissa maamme taide- ja kulttuurihistoriallisissa museoissa, yleisten kirjastojen Kirjasampo-portaalissa ja Sotasammossa.

16.4 Linkitetyn datan palvelut

Käytännön sovellusten kannalta merkittävä rooli semanttisessa webissä on linkitetyn datan palveluilla, joiden rajapintojen varaan linkitetyn datan sovellukset kehitetään. Edellä on jo esitelty linkitetyn datan julkaisuperiaatteita, kuten URI-tunnisteiden ratkointia sekä SPARQL-palvelupisteen käyttöä. Datapalvelu tarjoaa asiakkaalleen yleensä 1) mahdollisuuden saada datasta tarkempaa tietoa, 2) rajapinnat datan käyttämiseen ja 3) mahdollisuuden datan lataamiseen kopiona omaan käyttöön.

Verkossa olevia hyödyllisiä SPARQL-palvelupisteitä on listattu ja esitelty muun muassa W3C:n sivuilla.¹ Suosittuja palveluita ovat muun muassa Wikipedian dataa tarjoavat Wikidata ja DBpedia. Avoimet palvelut eivät kuitenkaan välttämättä aina ole toiminnassa, ja niiden käyt-

¹ <https://www.w3.org/wiki/SparqlEndpoints>

töön 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.



Linked Data Finland

Living Laboratory Data Service for the Semantic Web

[Home](#)
[Project](#)
[Datasets](#)
[Search Data](#)
[Schemas](#)
[Services](#)
[Policies](#)
[Documentation](#)
[Validation](#)
[Linked Data Science](#)
[Applications](#)
[Your Data?](#)
[Linked Data School](#)

This site is the Living Laboratory of the [Linked Data Finland](#) research initiative, conducted by the [Semantic Computing Research Group](#) at [Aalto University](#) in collaboration with University of Helsinki and a large consortium of Finnish public organizations and companies.

Our goal is to make life easier for both publishers as well as consumers of structured data on the Web. We base our work on the [Linked Data](#) paradigm and stack of standards, which combines an expressive, semantic data model ([RDF](#)) with standardized access mechanisms ([SPARQL](#) and [live HTTP URIs](#)).

5-star Linked Data

The baseline of our work is the [5-star Linked Data model](#), proposed [originally](#) by Tim Berners-Lee.

- ★ Make data available on the Web in whatever format.
- ★★ Make data available as structured data (e.g., Excel instead of an image scan of a table).
- ★★★ Use non-proprietary formats (e.g., CSV instead of Excel format).
- ★★★★ Use URIs to denote things, so that people can point at your data.
- ★★★★★ Link your data to other data to provide context.

7-star Linked Data Service

However, in our opinion, providing 5-star Linked Data is just the beginning. To actually make use of the datasets, consumers need more support in getting to know and access them, as well as a better grasp of their quality and provenance. To this end, we extend the model with two additional stars:

- ★★★★★ Provide your data with a schema and documentation so that people can *understand and re-use* your data easily.
- ★★★★★ Validate your data and denote its provenance so that people can *trust the quality* of your data.

This added support should come with as little extra work as possible to the data publisher. Our hypothesis is that

Kuva 16.2. Linked Data Finland -palvelun etusivu. Seitsemän tähden julkaisumalli kannustaa linkitetyn datan uusiokäyttöön. Datajulkaisut ja palvelut löytyvät vasemman sarakkeen linkkien kautta. (<http://ldf.fi>.)

Suomessa linkitetyn datan palveluita on pilotoitu erityisesti Aalto-yliopiston ja Helsingin yliopiston kehittämässä Linked Data Finland -palvelussa LDF.fi (ks. kuva 16.2). 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 uudelleenkäyttöä sovelluksissa. Tätä kannustaakseen LDF.fi laajentaa Tim Berners-Leen viiden tähden mallin (ks. kappale 7.4 sivulla 115) ”seitsemän tähden linkitetyn datan malliksi”. Kun viiden tähden hotellien lisäksi markkinoille on ilmestynyt seitsemänkin tähden hotelleja, voisi samaan pyrkiä datahotelleissakin:

Kuudennen tähden ideana on kannustaa datan julkaisijaa julkaisemaan datan ohella myös siinä käytetyt metadataskeemat ja niiden dokumentaatio, mikä helpottaa huomattavasti datan uudelleenkäyttöä ja validointia.

Seitsemäs tähti tulee, jos data on validoitua, eli jos julkaisija antaa takeita sille, että data on käytettyjen metadataskeemojen mukaista. Ajatuksena on kannustaa julkaisemaan testatusti laadukasta dataa.

LDF.fi-palveluun on liitetty lukuisia lisäpalveluita helpottamaan datan tuotantoa, julkaisua ja dataan tutustumista. Kun julkaistavaksi tarjotusta datasta luo standardin mukaisen metadatakuvauksen, LDF.fi-palvelu generoi sen perusteella datajulkaisulle automaattisesti oman kotisivun, jossa on paitsi kuvaus itse datasta ja sen lisenssiehdoista myös valmiita työkaluja linkitetyn 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*¹ linkitetyn datan tuottamista, julkaisemista ja hyödyntämistä varten.

LDF.fi-palvelun toteutukseen kuuluu Varnish-edustapalvelin, joka hoitaa HTTP-URI-pyyntöjen käsittelyn.² Taustalla olevat SPARQL-palvelupisteet on toteutettu Apache Jena Fuseki -alustan avulla.³

¹ <http://linda.seco.cs.aalto.fi>

² <https://varnish-cache.org/>

³ <https://jena.apache.org/documentation/fuseki2/>

17

Sovellusesimerkki: Sotasampo

Historiasta opimme, että
emme opi historiasta mitään.

Georg Wilhelm Friedrich Hegel
(1770–1831)

Tässä luvussa esitellään edellä 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. Sovellus sisältää sotahistorian sovellusalueelle luodun, linkitettyyn avoimeen dataan perustuvan tietoinfrastruktuurin, jonka varaan voidaan kehittää uusia sovelluksia. Sotasampo-järjestelmän ensimmäinen versio julkistettiin 27.11.2015 Kansallisarkistossa pidetyssä tilaisuudessa ja sai verkossa nopeasti kymmeniä tuhansia käyttäjiä.²

¹ Sotasampo-hankkeen kotisivuilta löytyy runsaasti aiheeseen liittyvää tietoa ja julkaisuja: <https://seco.cs.aalto.fi/projects/sotasampo/>.

² <http://seco.cs.aalto.fi/events/2015/2015-11-27-sotasampo/>

17.1 Tavoitteet ja käyttötapaukset

Itsenäisyytemme ajan sotahistoria, erityisesti talvi- ja jatkosota, on Suomen kansallisen identiteetin ja maanpuolustustahdon tärkeä tukipilari ja laajaa kiinnostusta herättävä historian tutkimuskohde niin tutkijoille kuin kansalaisillekin. Useimpien suomalaisten vanhemmat tai isovanhemmat osallistuivat tavalla tai toisella sodan tapahtumiin, ja monet menettivät henkensä. Sotasampo-hanke rakentaa suomalaista identiteettiä lisäämällä ymmärrystämme viime sodistamme ja pyrkii näin edistämään rauhaa. Mitä paremmin sodan inhimillinen hinta ymmärretään, sitä vähemmän sotia toivon mukaan tulevaisuudessa syttyy.

Sotasampo julkaisee linkitettyinä avoimena datana ja datapalveluna talvi- ja jatkosotaan liittyviä keskeisiä kansallisia tietoaaineistoja yhteistyössä Kansallisarkiston, Puolustusvoimien, Maanmittauslaitoksen, Suomen Sotahistoriallisen Seuran, Yleisradion ja muun yhteistyöverkoston kanssa. Tavoitteena on edistää tiedon saantia sekä tukea alan Digital Humanities -tutkimusta uusimman semanttisen webin ja kieliteknologian avulla. Kehitettäviä uusia teknisiä ratkaisuja voidaan soveltaa muihinkin Digital Humanities -alueen aineistoihin.

Datapalvelun hyödyntämistä pilotoidaan Sotasampo.fi-verkkopalvelussa, joka koostuu erilaisista näkymäsovelluksista aineistoihin, kuten sotien tapahtumat aikajanalla ja kartalla, henkilöt, joukko-osastot, historialliset paikat, veteraanien muistelmat, sotaurmat, valokuvat ja sankarihautausmaat. Hankkeen tavoitteena on tarjota semanttisesti rikas kokonaisuus sekä avoimena linkitettyinä datana sovelluksille että verkkosivustona ihmisille. Eri datasiilojen aineistoja rikastetaan toistensa avulla linkittämällä dataa yhteisen tapahtumaperustaisen ontologiainfrastruktuurin kautta.

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.¹
2. **Semanttinen portaali.** Sotasampo.fi on Sotasampo-datapalvelun perustalle kehitetty sovellusjoukko tutkijoille ja kansalaisille.

¹ Sotasammon (WarSampo) datapalvelu on käytettävissä Linked Data Finland -alustalla osoitteessa <http://www.ldf.fi/dataset/warsa> ja on myös osa kansainvälistä Linked Open Data -pilveä: <http://linkeddata.org/>.

17.2 Aineistot ja tiedon tuottajayhteisö

Vaikka Suomen sodista on saatavilla runsaasti tietoa sekä verkossa että sotahistoriallisena kirjallisuutena, sitä ei ole ollut saatavilla koneluettavassa muodossa avoimena datana. Tämä on vaikeuttanut 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 varustettuina. Kuvat ovat katseltavissa verkossa. Sotamuseolla on myös arkistoissaan muun muassa noin 1300 piirrosta kattava jatkosodan ajan TK-piirroskokoelma, sotasaalislippukokoelma, ilmavoimien 60 000 kuvan kokoelma vuosilta 1918–1960 ja yksityisten henkilöiden kuvakokoelmia. SA-kuva-arkiston verkkojulkaisulla oli ensimmäisen vuoden aikana noin miljoona vierailijaa.
- **Sotasurmatietokannat.** Kansallisarkisto on julkaissut verkkopalveluna useita sotahistoriallisia tietokantoja, kuten Suomen sotasurmat vuosina 1914–1922, Suomen sodissa vuosina 1939–1945 menehtyneet ja Suomi, sotavangit ja ihmishuovutukset vuosina 1939–1955. Sotasurma-aineistot vuosilta 1939–1945 ovat olleet ahkerassa käytössä verkossa: ensimmäisenä julkaisuvuotena 2013 aineistoihin tutustui noin 83 000 kävijää. Muita kiinnostavia aineistoja ovat noin 30 000 sotapäiväkirjaa, joiden avulla voi muun muassa tutustua yksittäisen sotilaan sotahistoriaan hänen joukko-osastonsa päiväkirjojen kautta. Yksittäisten rintamamiesten sota-ajan vaiheet selviävät sotilaiden kantanorteista, mutta niitä ei ole vielä digitoitu.
- ***Kansa Taisteli – miehet kertovat* -lehdet (1957—1986).** Lehden artikkelit, joita on yhteensä noin 3360 kappaletta, ovat paljolti rintamalla palvelleiden veteraanien itsensä kirjoittamia muistoja sota-ajan tapahtumista. Suomen Sotahistoriallinen Seura ja Bonnier julkaisivat ne verkossa kesällä 2014.
- **Sota-ajan filmi- ja ääniteaineistot.** Yleisradiolla ja Kansalli-

sella audiovisuaalisella arkistolla KAVA on arkistoissaan runsaasti sota-aikaan liittyvää kuva-, radio-, filmi- ynnä muuta materiaalia, josta osa löytyy jo esimerkiksi Ylen Elävästä arkistosta.

- **Sotahistorialliset kokoelmat.** Sotamuseossa ja eräissä muissakin maamme museoissa on runsaasti sota-aikaan liittyvää esineistöä ja dokumentteja, joita on osin julkaistu muun muassa Kulttuurisampo- (kulttuurisampo.fi) ja Finna-portaaleissa (finna.fi).
- **Kartat ja paikkatieto.** Maanmittauslaitos on avannut paikkatietodataansa. Saatavilla on digitoituna avoimena datana muun muassa luovutetun Karjalan alueen vanhat kartat ja niihin liittyvä noin 35 000 paikannimen geokoodattu paikannimirekisteri.¹
- **Sota-ajan muistot.** Suomalaisen Kirjallisuuden Seuran arkistossa on sota-ajalla kerättyä tietoa ja muistoja sota-ajan elämästä. Myös Svenska Litteratur Sällskapetilla on aineistoja muun muassa Ruotsiin lähetetyistä sotalapsista.

Sotasampo-hanke kokoaa edellä mainittuja datan julkaisijoita yhteen luomaan eri organisaatioiden erillisten datasiilojen aineistoista suuri, avoin linkitetyn datan semanttinen verkko ja sovelluksia. Sotasampo.fi:n ensimmäiseen versioon valittiin kuvan 17.1 taulukossa lueteltuja aineistoja. 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 on myös yksi Suomen itsenäisyyden satavuotisjuhlavuoden ohjelmahankkeista.

Seuraavaksi esitellään lyhyesti Sotasammon tietomallia ja sisällöntuotantoa, sitten portaalin sovelluksia ja lopuksi palvelun perustana oleva datapalvelu ja infrastruktuuri.

17.3 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 aktiivi-*

¹ Alkuperäisen rekisterin datan on käsityönä geokoodannut Jyrki Tiittanen.

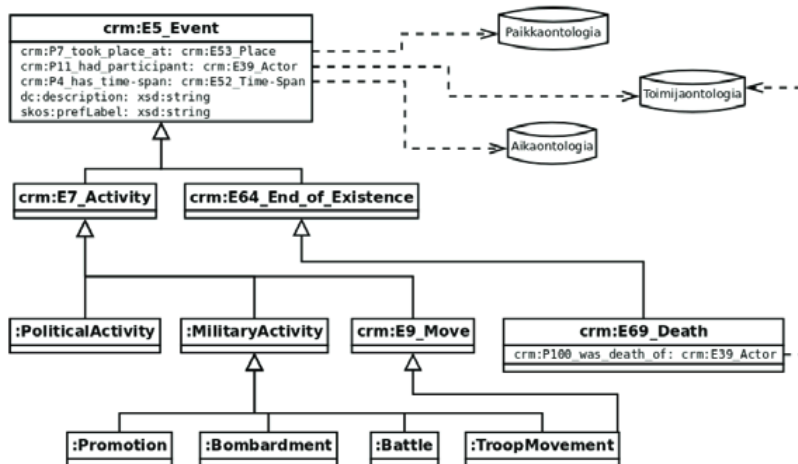
Datan luovuttaja	Aineisto (25.11.2015)	Määrä	Datan lähteet ja tekijänoikeus	Lisenssi linkitetty
1 Puolustusvoimien SA-	TK-kuvat ja videot	160 000	Puolustusvoimat & Aalto/SeCo	CC BY 4.0
2 Kansallisarkisto	Menehtyneet 39-45	95 000	Kansallisarkisto & Aalto/SeCo	CC BY 4.0
3 Kansallisarkisto	Organisaatiokortisto	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	Kansa taisteli -lehtien	3 360	Metadatat: Timo Hakala & Aalto/SeCo. Artikkelit © Suomen Sotahistoriallinen Seura ja Bonnier, http://sshs.fi .	CC BY 4.0 (metadatat)
7 Maanmittauslaitos	Paikannimirekisteri	800 000	Maanmittauslaitos (MML) & Aalto/SeCo	CC BY 4.0
8 Maanmittauslaitos	Karjalan topograf. kartat	47	MML & Aalto/SeCo	Ei-kaupallisiin tark.
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 pikkujähtiläiset, 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, organisaatiokortisto	CC BY 4.0
YHTEENSÄ		1 142 616		

Kuva 17.1. Sotasammon ensimmäisen version (2015) aineistot, datan määrät, lähteet, tekijänoikeudet ja lisenssiehdot. (<https://seco.cs.aalto.fi/projects/sotasampo/>.)

teetti (:PoliticalActivity). Tietomallin ydin on esitetty kuvassa 17.2. Ideana on kuvata sotahistoriallinen tieto tapahtumina, joita luonnehtivat erityisesti tapahtuman paikka (Missä?), aika (Milloin?) ja tapahtumaan liittyvät toimijat (Kuka?). Näiden kuvaukset löytyvät URI-tunnisteilla identifioituina omista aiheontologioistaan. CIDOC CRM valittiin, koska sen tapahtumamalli tukee luontevasti tapahtumakeskeistä sotahistorian esittämistä ja koska malli on jo laajemmassa käytössä oleva kehittynyt standardi.

Kuvan 17.1 aineistoista suurimmat, esimerkiksi SA-kuva-arkiston valokuvat, Menehtyneet-tietokannan tiedot, sotapäiväkirjat ja *Kansa Taisteli* -lehdet oli jo aiemmin digitoitu ja julkaistukin verkossa ainakin osittain. Niistä oli saatavilla metadatat digitaalisessa muodossa CSV-muotoisina tietokantakopioina ja Excel-taulukkoina. Taulukkomuotoon digitoitiin lisäksi käsityönä monista lähdeeteoksista tietoa muun muassa sota-ajan tapahtumista, Mannerheim-ristin ritareista, Puolustusvoimien organisaatiokorteista ja paikannimilistoista. Lisäksi näistä tietoa-aineistoista 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 suoraviivaista: taulukon rivit vastaavat usein luokkien yksilöitä



Kuva 17.2. 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. (Hyvönen ym., 2016)

ja sarakkeiden arvot ominaisuuksien arvoja. Esimerkiksi valokuvataulukossa jokainen rivi vastaa valokuvaa, johon voi liittyä tieto kuvaajasta, kuvausajasta, kuvauspaikasta ja valokuvan sisällön kuvaus tekstinä. Tilanteesta ja mallinnuksesta riippuen taulukon rivin muunnos voi kuitenkin olla mutkikkaampikin. Esimerkiksi Menehtyneet-tietokannassa kerrotaan vainajasta muun muassa sekä syntymä- että kuolinaika ja -paikka ja lisäksi mahdollisesta haavoittumisesta. CIDOC CRM -mallin mukaan näistä pitää luoda erilliset tapahtumaluokan yksilöt (esimerkiksi kuoleminen tiettyssä paikassa tiettyyn aikaan), jotka sitten voivat rikastaa muuta henkilöön liittyvää tapahtumatietoa vaikkapa Wikipediasta tai organisaatiokorteista.

Taulukkomuotoisen datan muunnoksia ja siivoamista (cleaning) varten on olemassa valmiita työkaluja, kuten OpenRefine. Sotasampo-hankkeessa jouduttiin kuitenkin kehittämään tähän työhön uusia välineitä ja muunnosohjelmia, jotka soveltuivat suomen kielelle ja 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, jota tarvittiin karttapohjaisissa visualisoin-

neissa. 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 karttojen päälle nykyisin käytössä olevaan koordinaattijärjestelmään WGS 84.¹ Metatieto kartoista talletettiin omaan RDF-graafiinsa, jonka kautta niitä voidaan hakea ja visualisoida Google-karttojen päällä paikkatiedon perusteella. Tässä hyödynnettiin Aalto-yliopiston historiallisten paikkojen ja karttojen ontologiapalvelua Hipla.fi.

17.4 Entiteettien automaattinen linkitys

Sotasampo linkittää myös tekstidokumentteja, kuten *Kansa Taisteli* -lehtien kaikkien vuosikertojen (1957–1986) 3360 artikkelia, jotka on julkaistu Sotahistoriallisen seuran kotisivuilla.² Näiden linkitystä varten oli käytettävissä hakemistomuotoinen metadata Excel-taulukkona.³ Tämän lisäksi teksteistä tunnistettiin kieliteknologisin välinein metatietoa perustuen muun muassa mainintoihin henkilöistä, joukko-osastoista 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).⁴ CORE kykenee etsimään dokumentissa olevia viittauksia ontologioissa oleviin käsitteisiin ja tarjoamaan niihin liittyvää lisätietoa verkossa olevien datapalveluiden, kuten DBpedia tai Sotasammon, kautta.

Keskeinen haaste linkitetyn datan tuotannossa on tekstuaalisia ilmauksia vastaavien linkkien muodostaminen. Esimerkiksi henkilöiden nimille, joukko-osastojen nimille ja lyhenteille, paikannimille ja sisällönkuvaailussa käytetyille asiasanoille pitää löytää vastaavat yksikäsitteiset käsitteet ontologioissa ja datassa. Jos data tulee tietokannasta tai on muuten määrämuotoista, tiedon rakenne auttaa linkityksessä. Esimerkiksi sa-

¹ WapWarper ohjelmiston kotisivu on <https://github.com/timwaters/mapwarper>. Ohjelmistosta asennettiin Sotasampoa varten ONKI-palveluun oma versio: <http://mapwarper.onki.fi>.

² <http://kansataisteli.sshs.fi/>

³ Hakemiston on laatinut Timo Hakala:

[urlhttp://kansataisteli.sshs.fi/files/Kansa_Taisteli_hakemisto_57.86.pdf](http://kansataisteli.sshs.fi/files/Kansa_Taisteli_hakemisto_57.86.pdf).

⁴ CORE kehitettiin ja sitä sovellettiin alunperin aiemmassa tutkimusprojektissa ensimmäisen maailmansodan aineistoihin (Mäkelä ym., 2017).

na *Taipale* tietokentässä, jossa on tietokantaskeeman mukaan henkilön sukunimi, ei voi viitata Kannaksella olevaan saman nimiseen paikkaan *Taipale*, jolla on luonnollisesti eri URI-tunnus.

Hankalinta linkityksen muodostamisessa on vapaan tekstin automaattinen annotointi, jossa tietokone muodostaa tekstistä semanttisen kuvauksen. Tähän ei riitä perinteinen nimettyjen entiteettien tunnistaminen (Named Entity Recognition, NER), jossa tunnistetaan esimerkiksi henkilöihin, organisaatioihin ja paikkoihin viittaavat sanat ja ilmaukset. 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, että tiedetään *Korhosen* olevan 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 linkitykseen tuovat aineistoon mahdollisesti liittyvät OCR-prosessin virheet, kuten esimerkiksi *Kansa Taisteli* -lehtien tapauksessa. 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

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

Säännölliset lausekkeet tarjoavat yhden lähestymistavan myös tällaisten muotojen tunnistamiseen, mutta tuloksena voi olla hyvinkin mutkikkaita muotoiluja. Esimerkiksi viimeinen listan muodoista voidaan tunnistaa seuraavalla lausekkeella:

¹ Ks. (Heino, 2017).

```
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. Kun niiden ja aineiston määrä kasvaa, käy helposti niin, että jokin sääntö laukeaa väärässä kontekstissa ja aiheuttaa virheen.

Sotasammossa linkitettiin lehtiaineistojen yli 3300 artikkelia henkilöihin, joukko-osastoihin ja paikkoihin sekä Wikipedian käsitteisiin. Samoin analysoitiin portaalin yli 1000 merkittävän historiallisen tapahtuman ja 160 000 valokuvan kuvailutekstit ja linkitettiin ne henkilöihin, joukko-osastoihin 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 (URI)	Ontologia
<i>Ylipäällikkö</i>	C. G. Mannerheim	Henkilöt
<i>eversti P. Talvelan</i>	kenraali Paavo Talvela	Henkilöt
<i>Tolvajärven</i>	Tolvajärvi	Paikat
<i>Ilomantsin</i>	Ilomantsi	Paikat
<i>Osasto Talvelan</i>	Osasto Talvela	Joukko-osastot

Taulukko 17.1. Tapahtuman tekstikuvauksesta tunnistetut entiteetit ja niiden linkitys.

Tarkastellaan toisena esimerkkinä kuvan 17.3 valokuvaa ja sen kuvatekstiä. Tavoitteena on selvittää, viittaako tekstissä mainittu *kersantti Leskinen* johonkuhun Sotasammon tuntemaan henkilöön ja jos, niin keeneen. Ongelman ratkaiseminen vaatii sovellusaluekohtaista tietämystä, jota Sotasampoon on ohjelmoitu jonkin verran. Tässä tapauksessa Sotasammon annotointialgoritmi löytää semanttisesta verkosta kolme mahdollista kersantti Leskistä. Näistä yksi, Reino Leskinen, palveli tekstissä mainitussa Sissipataljoona 2:ssa, joten tämä tulkinta näyttäisi toden-



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

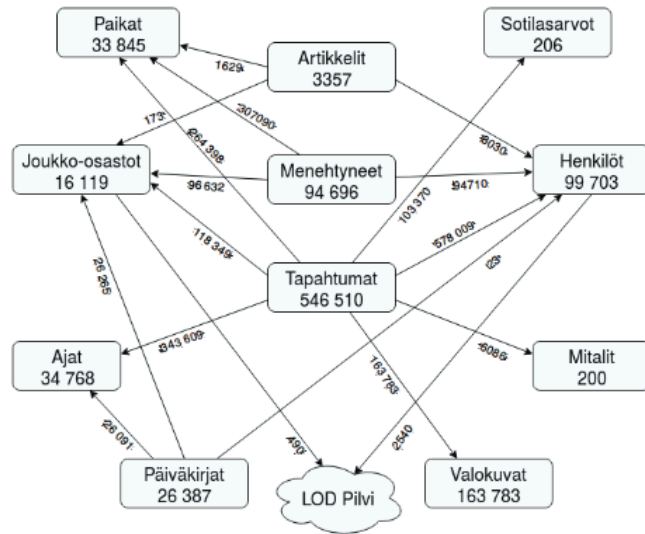
Kuva 17.3. Sotasammossa oleva SA-valokuva 7.2.1942 ja sen kuvateksti.

näköiseltä. 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 vain 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 luodun linkityksen laadun testaamiseksi valokuvien ja tapahtumien linkityksiä on tarkastettu käsityönä pienillä, 50–100 kohteen satunnaisotoksilla. Tarkkuus ja saanti näyttäisivät niiden perusteella olevan kohtuullista 70 prosentin luokkaa tehtävän haasteellisuudesta huolimatta. Virheitäkin siis kuitenkin esiintyy, ja koska tieto esimerkiksi eri henkilöistä ei ole täydellistä, jää tulkintoihin epävarmuutta. Lähdekriittisyys on paikallaan tällaisissa semanttisen webin sovelluksissa, joissa aineistot ovat laajoja ja joissa joudutaan luottamaan koneen automaattisesti tekemiin päätelmiin ja valintoihin ilman mahdollisuutta katavaan tarkastukseen ja korjailuun käsityönä. Osittainkin oikea tulos voi kuitenkin olla parempi kuin ei lainkaan tulosta.

Kuva 17.4 esittää datan tuotannon ja linkityksen lopputuloksena syntyntä Sotasammon linkitettyjen datajoukkojen pilveä vuoden 2017 alussa. 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. Data linkittyy myös Sotasammon ulkopuolelle muun muassa Wikipediaan ja Kansallisbiografiaan.

Sotasammon datapilveen voidaan linkittää uusia datajoukkoja ja näin



Kuva 17.4. Sotasammon datajoukot vuoden 2017 alussa, niiden luokkien yksilöiden lukumäärät ja joukkojen yksilöiden välisten linkkien määrät ja suunta. (Kuva: Petri Leskinen)

hyödyntää aiemmin tehtyä työtä yhä uudestaan. Esimerkiksi vuoden 2017 lopulla pilveen lisättiin Suomi 100 -juhlavuoden kunniaksi uusi datajoukko, jossa on tietoa nykyisen Suomen alueella olevasta yli 600 sankarihautausmaasta ja niiden noin 2400 valokuvasta, ja portaaliin kehitettiin tähän dataan perustuva uusi sovellusnäkökulma Sankarihautausmaat.¹ Vuonna 2018 järjestelmään ollaan linkittämässä uusi, tuhansien sotavankien tietoja sisältävä tietokanta, ja sille julkaistaan oma sovellusnäkökulma Sotavangit.

17.5 Portaalisovellus Sotasampo.fi

Sotasampo-portaali² koostuu joukosta toisiinsa linkitettyjä *näkymäsovelluksia* (application perspective), jotka tarjoavat erilaisia näkökulmia sodan aineistoihin. Kuvassa 17.5 näkyvät palvelun etusivulla olevat linkit eri näkymiin, jotka ovat Tapahtumat, Henkilöt, Joukko-osastot, Paikat, Kansa taisteli, Menehtyneet, Valokuvat ja Sankarihautausmaat. Seuraa-

¹ Tämä sovellus (Ikkala ym., 2017) sai viikon kuluessa n. 57 000 käyttäjää.

² <http://sotasampo.fi>

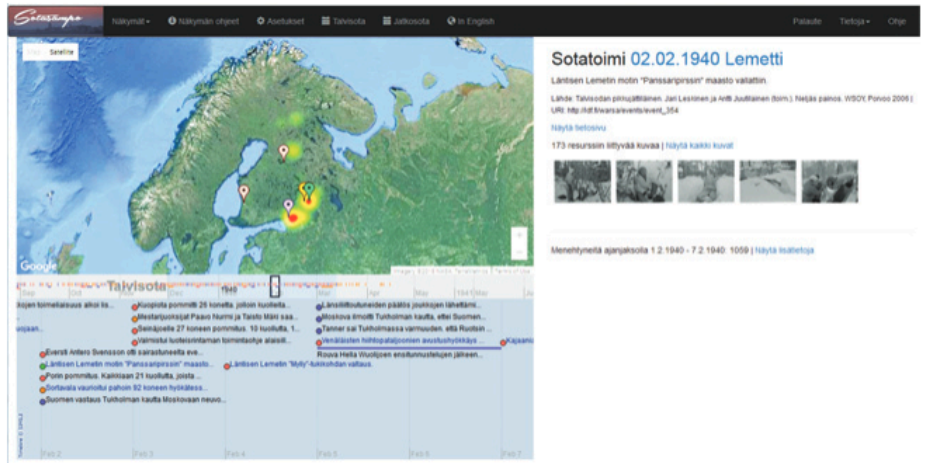
Esimerkkisivuja

Kuva 17.5. Sotasammon sovellusnäkymät, joista jokainen on oma datapalvelun SPARQL-rajapintaan perustuva sovelluksensa. (<http://sotasampo.fi>.)

vaksi tarkastellaan lyhyesti näiden näkymien toimintaa.

Tapahtumat-näkymän pääsivu (ks. kuva 17.6) esittää talvi- ja jatkosodan poliittisia ja sotilaallisia tapahtumia aikajanalla ja viiden päivän aikaikkunaan sisältyvät tapahtumat kartalla. Taustalla näkyvä lämpökartta lasketaan valitun aikaikkunan aikana menehtyneistä sotilaista sotaurmadatan perusteella. Ikkunaa ajassa siirtämällä muuttuvat tapahtumat ja sotaurmien määrää ilmaisevan lämpökartan väritys (punainen–keltainen–vihreä), mikä antaa yleiskuvan tapahtumien kehitymisestä ajan ja paikan suhteen.

Käyttäjä voi klikkaamalla valita tapahtuman aikajanalta tai kartalta. Kuvassa on valittu *Talvisodan pikkujättiläisessä* mainittu sotatoimi 2.2.1940, jossa ”Läntisen Lemetin motin ’Panssaripirssin’ maasto vallattiin”. Tapahtuman aikaan ja paikkaan liittyvä data ja semanttiset linkit



Kuva 17.6. Sotasammon Tapahtumat-sovellusnäkymä. (<http://sotasampo.fi>.)

muihin aineistoihin näytetään oikealla. Kuvaukseen sisältyvät seuraavat toiminnallisuudet: 1) Linkki ”02.02.1940” johtaa päivämäärän ”kotisivulle”, johon on koottu päiväkohtaisesti tietoa tapahtumista sekä menehtyneistä sotilaista ja otetuista valokuvista. 2) ”Lemetti” on linkki Sotasammon Paikat-näkymän tietosivulle Lemetistä, jolla voi tarkastella Lemettiä historiallisilla kartoilla sekä paikkaan eri tavoin linkittyvää tietoa. Sekä historialliset kartat että muu tieto tulevat jälleen toisista datajoukoista. 3) Linkki ”Näytä tietosivu” vie tapahtuman omalle kotisivulle, jossa esitellään tapahtumaan loogisesti liittyviä muita tietoja, esimerkiksi kootusti eri aikoina samalla paikalla käytyjä taisteluja ja muita tapahtumia tai henkilöitä, jotka osallistuivat kyseiseen tapahtumaan. Tällaisia semanttisia suosittehuja voidaan luoda joustavasti SPARQL-rajapinnan kyselyiden avulla. 4) Tapahtumatietoon on valmiiksi linkitetty oikealla näkyviä Puolustusvoimien SA-kuva-arkiston kuvia, jotka on otettu samalla alueella ja samaan aikaan kuin valitun tapahtuman tiedetään tapahtuneen. 5) Linkin ”Näytä lisätietoja” takaa löytyy aina yhteenveto valitun ajanjakson kuluessa menehtyneistä sotilaista.

Yhteisen datajulkaisun luomiselle on selkeä motivaatio: yhteistyössä eri datajoukkojen sisällöt rikastavat toisiaan. Esimerkiksi Henkilöt-näkymän kautta voidaan tieto sotilaan joukko-osastosta yhdistää organisaatiokorteista saatavaan tietoon joukko-osaston tapahtumista sekä joukko-osaston sotapäiväkirjoihin, jolloin voidaan rakentaa ja visualisoi-

da kartalla paitsi joukko-osastojen 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://www.sotasampo.fi/places/page?uri=http://ldf.fi/warsa/places/municipalities/m_place_87

Kotisivut on luotu systemaattisesti muillekin Sotasammon keskeisten aineistoluokkien yksilöille, kuten henkilöille ja joukko-osastoille. Tämä mekanismi mahdollistaa tietosivujen uudelleenkäytön paitsi Sotasampo.fi:ssä myös missä tahansa toisissa sovelluksissa HTML-muodossa, joka voidaan näyttää selaimessa.

Henkilöt

Hae tunnettuja viime sodan henkilöitä kirjoittamalla heidän nimen alkuosat tydensivustoon josta valitsee henkilön, josta halutaan tietoa. Henkilön tiedot ja kuvasivut löytyvät muuttamalla nimistä. Jos et löydä henkilöä, voit lisätä uuden henkilön joukko-osastotietojen kautta.

Paavo Juho Talvela

19.01.1901 Heinämaa - 30.09.1973 Helsinki

Henkilön tiedot

Paavo Juho Talvela (kunperi Toivonen) 19. tammikuuta 1901 Heinämaan maalaiskunta - 30. syyskuuta 1973 Helsinki oli suomalainen upseeri, joka osasi Saksassa jäännöskutauksen. Hän osallistui sisällissotaan, haavoittui sekä talvi- ja jatkosotien aikana. Hän toimi Suomessa 1930-luvulta sekä jatkosodan jälkeen muuten muutamissa sotamuseoiden palveluksissa. Lähde: Wikipedia, Suomen sotajärjestelmän kuva, Biografias, www.kansalliskirjasto.fi, Suomen Sotaseura, 1930-luvun sotajärjestelmän kuva, Paavo 1901, Komentajien tiedot, http://www.kansalliskirjasto.fi/kotisivu

Talvela, Paavo

Syntynyt 1901 paikkaa Helsinki. Kuoli 1973 paikkaa Helsinki.

Tapahtumat (7)

Ylipäätökä muuttanut Paavo Talvelan joukko-osastot ja komentajat Suomessa...

Ylipäätökä muuttanut Paavo Talvelan joukko-osastot ja komentajat Suomessa...

Ylipäätökä muuttanut Paavo Talvelan joukko-osastot ja komentajat Suomessa...

Ylipäätökä muuttanut Paavo Talvelan joukko-osastot ja komentajat Suomessa...

Ylipäätökä muuttanut Paavo Talvelan joukko-osastot ja komentajat Suomessa...

Ylipäätökä muuttanut Paavo Talvelan joukko-osastot ja komentajat Suomessa...

Ylipäätökä muuttanut Paavo Talvelan joukko-osastot ja komentajat Suomessa...

Joukko-osastot (2)

Aikuisuus (1901-1902)

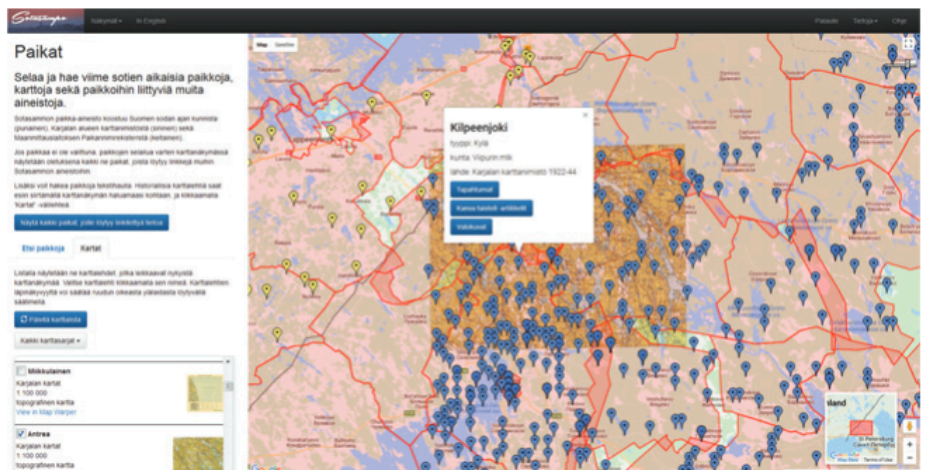
Vt armeijakuori (1901-1902)

Kuva 17.7. Sotasammon Henkilöt-näkymän kautta löytynyt Paavo Talvelan kotisivu. (<http://sotasampo.fi>.)

Henkilöt-näkymän kautta voi tutustua Sotasammon aineistoihin haakemalla henkilöiden kotisivuja ja heihin eri tavoin liittyviä aineistoja. Kuvassa 17.7 käyttäjä on kirjoittanut vasemmalla olevaan hakukenttään *Talvela* ja valinnut tulostulistasta *Paavo Juhani Talvelan*, jonka kotisivu näkyy oikealla *Henkilön tiedot* -välilehdellä. Sivulle linkittyvät Paavo

Talvelan valokuvat (yhteensä 193) ja linkit historiallisiin tapahtumiin, joissa hän oli mukana. Sivun oikeassa laidassa on elämäkerrallista tietoa, joka on peräisin SKS:n Kansallisbiografiasta. Kuvakaappauksen ulkopuolelle jäävät linkit Talvelan joukko-osastojen kotisivuille, ylennykset, joista viimeinen on ylennys jalkaväen kenraaliksi 6.12.1966, tieto hänelle myönnetyistä kunniamerkeistä, linkkejä häneen liittyviin paikkoihin sekä 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 linkitettyinä myös joukko-osastoihin liittyvät autenttiset sotapäiväkirjat. Joukko-osastojen historian avulla voi selvittää niihin kuuluneiden henkilöiden liikkeitä sodassa, vaikka näistä henkilöistä 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 eri lähteistä mainitulla noin 5000 tunnetulla sotilaalla. Elossa selvinneistä sotilaista ei ainakaan vielä ole käytettävissä tarkempia tietoja kuin noissa tapauksissa.



Kuva 17.8. Sotasammon paikka-sovelluksen kuvakaappaus Viipurin pohjoispuolisen alueen paikoista, joihin liittyy tapahtumia, *Kansa Taisteli* -lehden artikkeleita tai valokuvia. (<http://sotasampo.fi>.)

Paikat-näkyvässä sodan tapahtumat ja aineistot on jäsennetty karttojen avulla (ks. kuva 17.8). Kartoilla näkyviä merkkejä klikkaamalla avautuu ikkuna valitun paikan tietosisältöihin. Tässä tapauksessa tarkasteltavaksi on otettu ”Kilpeenjoen” kylä, josta on linkkejä eteenpäin Sotasammon tapahtumiin, lehtiartikkeleihin sekä valokuviiin. Modernien Googlen karttojen ohella käytettävissä ovat näiden päälle läpinäkyvästi asemoidut Hipla.fi-palvelun historialliset Karjalan kartat, joista löytyvät muun muassa vanhat sotien aikaiset paikannimet. Kuvassa keskellä näkyy näistä yksi, Antrean kartta, joka on valittu vasemmalla näkyvästä karttaluettelosta.

Kansa taisteli -näkyvä tarjoaa semanttiseen fasettihakuun perustuvan hakukäyttöliittymän *Kansa Taisteli* -lehtien artikkeleihin. Kun kiinnostava artikkeli löytyy, siihen voi tutustua paitsi lukemalla myös lisälinkkien kautta, joita luodaan automaattisesti tekstissä esiintyvistä henkilönnimistä, paikannimistä ja muista käsitteistä. Linkit johtavat Sotasammon vastaaville kotisivuille ja Wikipediaan.

Menehtyneet-näkyvän 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.

Valokuvat-näkyvässä voi hakea fasettivalinnoilla vastaavaan tapaan sodanaikaisia valokuvia esimerkiksi henkilöiden ja paikkojen avulla aineistoa suodattaen.

Sankarihautausmaat-näkyvä sisältää hakukoneen ja tiedot yli 600 Suomen sankarihautausmaasta ja niiden kotisivut valokuvineen. Kotisivuilla voi luoda ja tutkia tilastoja ja visualisointeja kunkin hautausmaan vainajista, esimerkiksi nähdä heidän kuolinpaikkansa kartalla, iät histogrammina tai sotilasarvojen jakauman. Sovellus tarjoaa paikallishistoriallisia näkymiä sodan uhreihin. Suomessa, päinvastoin kuin useimmissa muissa maissa, sankarivainajia ei haudattu rintamille, vaan heidät pyrittiin tuomaan kotiin haudattaviksi. Tämä vainajen muistoa kunnioittava työ jatkuu edelleenkin.

17.6 Datapalvelu Sotasampo

Kaikki Sotasammon kuvan 17.4 datajoukot ovat käytettävissä avoimella Linked Data Finland -palvelualustalla (LDF.fi), jonka SPARQL-rajapinnan käyttöön perustuvat suoraan kaikki Sotasampo.fi-portaalin eri näkymäsovellukset.

Sotasammon datajoukkojen julkaisu alkoi Kansallisarkiston kolmen sotasurmatietokannan avaamisella. Näistä 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 aina osoitteesta, 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 YASGUI:n 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.

17.7 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. 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 tärkeässä asemassa monissa Digital Humanities -alueen hankkeissa, jollainen Sotasampokin on.

Sotasampo-pilotin pidemmän aikavälin tavoitteena on käynnistää sotahistoriallisen tiedon pysyvä tuotanto ja ylläpito linkitettynä datana. Ajatuksena on, että työ siirtyisi jatkossa jonkin muistiorganisaation tehtäväksi. 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 Sotasampon, vaan välissä tarvitaan muunnoksia ja mahdollisesti käsityötä. Lisäksi jotkin muutokset, kuten vaikka uuden historiallisen paikan tai henkilön lisääminen, voivat olla kriittisiä myös muiden organisaatioiden tietokantojen ylläpidon kannalta, ja tieto muutoksesta ja uusista tunnistuksista 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.

Sotasampo toteutettiin Aalto-yliopistossa joukkona toisiinsa liittyviä opinnäytetöitä, joihin tämä lukukin perustuu. Työt käsittelevät järjestelmän perustana olevaa tapahtumaperustaista mallia ja aineistojen linkitystä,¹ sotilashenkilöiden ja joukko-osastojen ontologista mallintamista, hakua ja suosittelua,² *Kansa Taisteli* -lehtien automaattista linkitystä

¹ Ks. (Heino, 2017).

² Ks. (Leskinen, 2016).

ja hakua¹ sekä historiallisten paikkojen ontologioita ja karttapalvelua.² Tuloksista on raportoitu kansainvälisissä tieteellisissä konferensseissa.³ Kesällä 2017 Sotasampo sai arvostetun LODLAM Challenge Open Data Prize -palkinnon Venetsiassa.

¹ Ks. (Tamper, 2016).

² Ks. (Ikkala, 2016).

³ Ks. (Hyvönen ym., 2016; Koho ym., 2017; Heino ym., 2017; Tamper ym., 2017; Leskinen ym., 2017; Ikkala ym., 2017).

Liite: Esimerkki OWL-ontologiasta

Ohessa on W3C:n suosituksessa¹ esitetty laajempi esimerkki ontologia-dokumentista, jossa on määritelty esimerkiksi erilaisia sukulaisuussuh-teita.

```
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 "Repre-
sents 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 ) )
```

¹ <http://www.w3.org/TR/2012/REC-owl2-primer-20121211/>

```

Declaration( Class( :MyBirthdayGuests ) )
Declaration( Class( :Dead ) )
Declaration( Class( :Orphan ) )
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 per-
son." )
  :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

- David Allemang & Jim Hendler. *Semantic Web for the Working Ontologist*. Morgan Kaufmann, San Francisco, 2008.
- Gustavo Alonso, Fabio Casati, Harumi Kuno & Vijay Machiraju. *Web Services: Concepts, Architectures and Applications*. Springer-Verlag, 2003.
- Paul Anderson. *Web 2.0 and Beyond: Principles and Technologies*. Chapman and Hall/CR, USA, 2012.
- Grigoris Antoniu, Frank van Harmelen & Rinke Hoekstra. *Semantic Web Primer (3rd Edition)*. The MIT Press, Cambridge, Massachusetts, 2012.
- Murtha Baca, toim. *Introduction to Metadata*. Getty Publications, Los Angeles, 2008.
- Tim Berners-Lee, Mark Fischetti & M. Dertouzos. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*. Harper Business, 2000.
- Ronald Brachman & Hector Levesque. *Knowledge Representation and Reasoning*. Morgan Kaufmann, San Francisco, 2004.
- Ivan Bratko. *Prolog Programming for Artificial Intelligence (4th edition)*. Pearson, Harlow, U.K., 2012.
- Megan R. Brett. Topic modeling: A basic introduction. *Journal of Digital Humanities*, 2(1), 2012. URL <http://journalofdigitalhumanities.org/2-1/topic-modeling-a-basic-introduction-by-megan-r-brett/>.
- William F. Clocksin & Christopher S. Mellish. *Programming in Prolog*. Springer-Verlag, 2003.

- Nick Crofts, Martin Doerr, Tony Gill, Stephen Stead & Matthew Stiff, toim. *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.
- Martin Doerr. The CIDOC CRM—an ontological approach to semantic interoperability of metadata. *AI Magazine*, 24(3):75–92, 2003.
- Martin Doerr. Ontologies for cultural heritage. Teoksessa Staab & Studer (2009), sivut 463–486.
- Deborah Edwards & Lynda Hardman. Lost in hyperspace: Cognitive mapping and navigation in a hypertext environment. Teoksessa R. McAleese, toim, *Hypertext: Theory into Practice*, sivut 105–125. Blackwell Scientific Publications Ltd, 1989.
- Jerome Euzenat & Pavel Shvaiko. *Ontology Matching*. Springer–Verlag, 2007.
- Andy Field, Jeremy Miles & Zoe Field. *Discovering Statistics Using R*. SAGE Publications Inc., USA, 2015.
- Richard Gartner. *Metadata. Shaping Knowledge from Antiquity to the Semantic Web*. Springer–Verlag, 2016.
- Anne J. Gilliland. Setting the stage. Teoksessa Baca (2008), sivut 1–19.
- Nicola Guarino, Daniel Oberle & Steffen Staab. What is an ontology? Teoksessa Staab & Studer (2009), sivut 1–17.
- Erkki Heino. Sotahistorian kuvaaminen ja rikastaminen linkitettyinä datana. Master’s thesis, University of Helsinki, Dept. of Computer Science, Finland, 2017. URL <https://helda.helsinki.fi/handle/10138/229154>.
- Erkki Heino, Minna Tamper, Eetu Mäkelä, Petri Leskinen, Esko Ikkala, Jouni Tuominen, Mikko Koho & Eero Hyvönen. Named entity linking in a complex domain: Case second world war history. Teoksessa *Proceedings, Language, Technology and Knowledge (LDK 2017)*, sivut 120–133. Springer–Verlag, 2017. URL <http://seco.cs.aalto.fi/publications/2017/heino-et-al-nel-2017.pdf>.
- Pascal Hitzler, Markus Krötzsch & Sebastian Rudolph. *Foundations of Semantic Web technologies*. Springer–Verlag, 2010.
- Eero Hyvönen, toim. *Semantic Web Kick-Off in Finland*. HIIT Publica-

tions, 2002-001. Helsinki Institute for Information Technology (HIIT), Helsinki, Finland, 2002. <http://seco.cs.aalto.fi/publications/2002/hyvonen-semantic-web-kick-off-2002.pdf>.

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. URL <http://www.morganclaypool.com/doi/abs/10.2200/S00452ED1V01Y201210WBE003>.

Eero Hyvönen, Ilkka Karanta & Markku Syrjänen, toim. *Tekoälyn ensyklopedia*. Gaudeamus, 1993.

Eero Hyvönen, Eetu Mäkelä, Mirva Salminen, Arttu Valo, Kim Viljanen, Samppa Saarela, Miika Junnila & Suvi Kettula. MuseumFinland—Finnish museums on the semantic web. *Journal of Web Semantics*, 3(2):224–241, 2005. URL <http://seco.cs.aalto.fi/publications/2005/hyvonen-makela-et-al-museumfinland-finnish-2005.pdf>.

Eero Hyvönen, Kim Viljanen, Eetu Mäkelä, Tomi Kauppinen, Tuukka Ruotsalo, Onni Valkeapää, Katri Seppälä, Osma Suominen, Olli Alm, Robin Lindroos, Teppo Käsälä, Riikka Henriksson, Matias Frosterus, Jouni Tuominen, Reetta Sinkkilä & Jussi Kurki. Elements of a national semantic web infrastructure—case study Finland on the semantic web (invited paper). Teoksessa *Proceedings of the First International Semantic Computing Conference (IEEE ICSC 2007)*, Irvine, California, sivut 216–223. IEEE Press, Sept 2007. URL <http://seco.cs.aalto.fi/publications/2007/hyvonen-et-al-elements-2007.pdf>.

Eero Hyvönen, Kim Viljanen, Jouni Tuominen & Katri Seppälä. Building a national semantic web ontology and ontology service infrastructure—the FinnONTO approach. Teoksessa *Proceedings of the 5th European Semantic Web Conference (ESWC 2008)*. Springer-Verlag, 2008.

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 Palonen, Matias Frosterus, Reetta Sinkkilä, Panu Paakkari, Joonas Laitio & Katariina Nyberg. CultureSampo – Finnish culture on the Semantic Web 2.0. Thematic perspectives for the end-user. Teoksessa *Museums and the Web 2009, Proceedings*. Archives and Museum Informatics,

- Toronto, 2009. URL <http://seco.cs.aalto.fi/publications/2009/hyvonen-et-al-culsa-mw-2009.pdf>.
- Eero Hyvönen, Erkki Heino, Petri Leskinen, Esko Ikkala, Mikko Koho, Minna Tamper, Jouni Tuominen & Eetu Mäkelä. WarSampo data service and semantic portal for publishing linked open data about the Second World War history. Teoksessa *The Semantic Web – Latest Advances and New Domains (ESWC 2016)*, sivut 758–773. Springer–Verlag, 2016. URL <http://seco.cs.aalto.fi/publications/2016/hyvonen-et-al-warsa.pdf>.
- Eero Hyvönen, Petri Leskinen, Minna Tamper, Jouni Tuominen & Kirsi Keravuori. Semantic National Biography of Finland. Teoksessa *Digital Humanities in Nordic Countries, 3rd Conference, Proceedings*, 2018. URL <http://seco.cs.aalto.fi/publications/2018/hyvonen-et-al-snb-2018.pdf>.
- Esko Ikkala. Suomalainen historiallisten paikkojen ja karttojen ontologiapalvelu. Master’s thesis, Aalto University, Dept. of Computer Science, Finland, 2016. URL <http://seco.cs.aalto.fi/publications/2016/ikkala-msc-thesis-2016.pdf>.
- Esko Ikkala, Mikko Koho, Erkki Heino, Petri Leskinen, Eero Hyvönen & Tomi Ahoranta. Prosopographical views to Finnish WW2 casualties through cemeteries and Linked Open Data. Teoksessa Alessandro Adamou, Enrico Daga & Leif Isaksen, toim, *WHiSe 2017. Workshop on Humanities in the Semantic Web*, sivut 45–56. CEUR Workshop Proceedings, 2017. <http://ceur-ws.org/Vol-2014/>.
- Holden Karau, Andy Konwinski, Patrick Wendell & Matei Zaharia. *Learning Spark: Lightning-Fast Big Data Analysis*. O’Reilly, USA, 2015.
- Mikko Koho, Eero Hyvönen, Erkki Heino, Jouni Tuominen, Petri Leskinen & Eetu Mäkelä. Linked Death – Representing, Publishing, and Using Second World War Death Records as Linked Open Data. Teoksessa *The Semantic Web: ESWC 2017 Satellite Events*, sivut 369–383. Springer–Verlag, 2017. URL <http://seco.cs.aalto.fi/publications/2017/koho-et-al-linked-death-extended-2017.pdf>.
- Douglas Laney. 3D data management: Controlling data volume, velocity and variety. MetaGroup Inc., USA, 2001.

- Petri Leskinen. Sotilashenkilöiden ja joukko-osastojen mallintaminen ja käyttö toimijaontologiana. Master's thesis, Aalto University, Dept. of Computer Science, Finland, 2016. URL <http://seco.cs.aalto.fi/publications/2016/leskinen-msc-thesis-2016.pdf>.
- Petri Leskinen, Mikko Koho, Erkki Heino, Minna Tamper, Esko Ikkala, Jouni Tuominen, Eetu Mäkelä & Eero Hyvönen. Modeling and using an actor ontology of Second World War military units and personnel. Teoksessa *Proceedings of the 16th International Semantic Web Conference (ISWC 2017)*, sivut 280–296. Springer-Verlag, 2017. URL <http://seco.cs.aalto.fi/publications/2017/leskinen-et-al-warsa-actor-ontology-2017.pdf>.
- John Lloyd. *Foundations of Logic Programming (2nd edition)*. Springer-Verlag, 1987.
- Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.
- Peter Mika & Mike Potter. Metadata statistics for a large web corpus. Teoksessa *Proceedings of Linked Data on the Web Workshop (LDOW 2012)*, at the *21th International World Wide Web Conference (WWW 2012)*, 2012.
- Eetu Mäkelä. Combining a REST lexical analysis web service with SPARQL for mashup semantic annotation from text. Teoksessa *Proceedings of the ESWC 2014 demonstration track*. Springer-Verlag, 2014. URL <http://seco.cs.aalto.fi/publications/2014/makela-sarpa.pdf>.
- Eetu Mäkelä, Kaisa Hypén & Eero Hyvönen. BookSampo—lessons learned in creating a semantic portal for fiction literature. Teoksessa *Proceedings of ISWC-2011, Bonn, Germany*. Springer-Verlag, 2011a. URL <http://seco.cs.aalto.fi/publications/2011/makela-hypen-hyvonen-booksampo.pdf>.
- Eetu Mäkelä, Aleksi Lindblad, Jari Väättäin, Rami Alatalo, Osmo Suominen & Eero Hyvönen. Discovering places of interest through direct and indirect associations in heterogeneous sources—the TravelSampo system. Teoksessa *Terra Cognita 2011: Foundations, Technologies and Applications of the Geospatial Web*, 2011b. CEUR Workshop Procee-

dings, Vol 798, <http://ceur-ws.org/Vol-798/>.

Eetu Mäkelä, Tuukka Ruotsalo & Hyvönen. How to deal with massively heterogeneous cultural heritage data—lessons learned in CultureSampo. *Semantic Web – Interoperability, Usability, Applicability*, 3(1):85–109, 2012. URL <http://www.semantic-web-journal.net/content/how-deal-massively-heterogeneous-cultural-heritage-data-%E2%80%93-lessons-learned-culturesampo>.

Eetu Mäkelä, Kaisa Hypén & Eero Hyvönen. Fiction literature as linked open data—the BookSampo dataset. *Semantic Web – Interoperability, Usability, Applicability*, sivut 173–188, 2013. URL <http://seco.cs.aalto.fi/publications/2013/makela-et-al-booksampo-swj.pdf>.

Eetu Mäkelä, Juha Törnroos, Thea Lindquist & Eero Hyvönen. WW1LOD: An application of CIDOC CRM to World War I linked data. *International Journal on Digital Libraries*, 18(4):333–343, Nov 2017. doi: 10.1007/s00799-016-0186-2. URL <https://doi.org/10.1007/s00799-016-0186-2>.

Natasha F. Noy & Mathieu d’Aquin. Where to publish and find ontologies? A survey of ontology libraries. *Web Semantics: Science, Services and Agents on the World Wide Web*, 11(0), 2012.

Michael Piotrowski. *Natural Language Processing for Historical Texts*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool, Palo Alto, USA, 2012.

Stuart Russell & Peter Norvig. *Artificial Intelligence. A Modern Approach*. Prentice-Hall, Upper Saddle River, New Jersey, 2010.

Katri Seppälä & Eero Hyvönen. Asiasanaston muuttaminen ontologiaksi. Yleinen suomalainen ontologia esimerkkinä FinnONTO-hankkeen mallista. Technical report, National Library of Finland, Plans, Reports, Guides, 2014. URL <http://www.doria.fi/handle/10024/67050>. <https://www.doria.fi/handle/10024/96825>.

John Sowa. *Knowledge Representation. Logical, Philosophical, and Computational Foundations*. Brooks/Cole, 2000.

Steffen Staab & Rudi Studer, toim. *Handbook on Ontologies (2nd Edition)*. Springer-Verlag, 2009.

Brian Steele, John Chandler & Swarna Reddy. *Algorithms for Data*

Science. Springer-Verlag, 2016.

Leon Sterling & Ehud Y. Shapiro. *The Art of Prolog – Advanced Programming Techniques*, 2nd Ed. MIT Press, 1994.

Osma Suominen, Eero Hyvönen, Kim Viljanen & Eija Hukka. HealthFinland – a national semantic publishing network and portal for health information. *Journal of Web Semantics*, 7(4):287–297, Dec 2009a. URL <http://seco.cs.aalto.fi/publications/2009/suominen-et-al-healthfinland-2009.pdf>.

Osma Suominen, Kim Viljanen & Eero Hyvönen. TerveSuomi-portaalin metatietomäärittely. Technical report, Semantic Computing Research Group, SW 2.0 project, 2009b. <http://seco.cs.aalto.fi/publications/2009/tervesuomi-metatieto-ver2.0.pdf>.

Osma Suominen, Sini Pessala, Jouni Tuominen, Mikko Lappalainen, Susanna Nykyri, Henri Ylikotila, Matias Frosterus & Eero Hyvönen. Deploying national ontology services: From ONKI to Finto. Teoksessa Alessandro Adamou, Enrico Daga & Leif Isaksen, toim, *Proceedings of the Industry Track at the International Semantic Web Conference 2014*. CEUR Workshop Proceedings, 2017. <http://ceur-ws.org/Vol-1383/>.

Minna Tamper. Extraction of entities and concepts from Finnish texts. Master's thesis, Aalto University, Dept. of Computer Science, Finland, 2016. URL <http://www.seco.hut.fi/publications/2005/tamper-msc-thesis-2016.pdf>.

Minna Tamper, Petri Leskinen, Esko Ikkala, Arttu Oksanen, Eetu Mäkelä, Erkki Heino, Jouni Tuominen, Mikko Koho & Eero Hyvönen. AA-TOS: a configurable tool for automatic annotation. Teoksessa *Language, Technology and Knowledge*, sivut 276–289. Springer-Verlag, 2017. URL <http://seco.cs.aalto.fi/publications/2017/tamper-et-al-aatos-2017.pdf>.

Jouni Tuominen, Nina Laureenne, Mikko Koho & Eero Hyvönen. The birds of the world ontology AVIO. Teoksessa *The Semantic Web: ESWC 2013 Satellite Events*, sivut 300–301. Springer-Verlag, 2013. URL <http://seco.cs.aalto.fi/publications/2013/tuominen-et-al-avio.pdf>.

Hakemisto

A

ABox159, 177
adaptivity (sopeutuvuus) ...20
aggregaattifunktio (aggregate
function)103
aggregate function
(aggregaattifunktio)103
aiheiden tunnistus (topic
modeling)210
aiheontologia (domain
ontology)123, 213
AJAX26
AJAX-tekniikat194
aksiooma (axiom)158
alialue (subdomain)57
alikysely (subquery)106
aliominaisuus (subproperty)
137
alternative (vaihtoehto)82
alternative label
(vaihtoehtoinen nimike) .141
aluenimi (domain name) ...56,
111
aluerajoite (domain constraint)
137
anchor (ankkuri)59
ankkuri (anchor)59

annotation property
(annotointiominaisuus) .167,
171
annotointiominaisuus
(annotation property) ...167,
171
appellaatio (appellation)133
appellation (appellaatio)133
arc (kaari)52
arkipäättely (common sense)
185
Artificial Intelligence (tekoäly)
152
arvorajoite (range constraint)
137
arvosijoitus (value assignment)
101
asiakasjärjestelmä (client) ..26
asiantuntijajärjestelmä (expert
system)222
ASK92
atomic formula (atomikaava)
153, 178
atomikaava (atomic formula)
153, 178
automaattinen annotointi
(automatic annotation) ..208

automatic annotation
 (automaattinen annotointi)
 208

avainmekanismi (key) 170

avoimen maailman oletus (open
 world assumption) 183

avoin tieto 21

axiom (aksiooma) 158

B

bag (monijoukko) 82

base namespace
 (kantanimiavaruus) 71

Berners-Lee, Tim 15, 27

big data (suurdata) 24

BITCOIN-tunnisteet 57

blank node (nimetön solmu) 73

broader term (laajempi käsite)
 144

browser (selain) 56

browsing (selailu) 32

C

cardinality constraint
 (kardinaliteettirajoite) ... 126

cataloging rules
 (luettelointiohje) 126

CERN-tutkimuskeskus 15

CIDOC CRM 130, 202

class (luokka) 42, 135

class hierarchy
 (luokkahierarkia) 136

CLEAR 107

client (asiakasjärjestelmä) .. 26

Closed World Assumption
 (suljetun maailman oletus)
 183

collection (kokoelma) 82

Comma Separated Values
 (CSV-muoto) 54

comment (kommentti) 68

common sense (arkipäätely)
 185

concept (käsite) 140

concept scheme (käsitelmä)
 140

conclusion (päätelmä) 152

condition (ehto) 178

constant (vakio) 154

constraint (rajoite) 137

CONSTRUCT 92

container (kokoelmaluokka) .82

container (säiliö) 81

content negotiation
 (sisältöneuvottelu) 113

context (käyttökonteksti) ... 36

contingent (kontingentti) .. 154

contradiction (ristiriita) ... 154

controller (käsittelijä) 194

COPY 107

CREATE 107

CSV-muoto (Comma Separated
 Values) 54

CWA 183

D

data analysis (data-analyysi) 25

data science (datatiede) 25

data silo (datasiilo) 189

data type (tietotyyppi) 74

data validation (datan
 validointi) 210

data-analyysi (data analysis) 25

datajoukko (dataset) 28

datan validointi (data
 validation) 210
 dataset (datajoukko) 28
 datasiilo (data silo) 189
 datatiede (data science) 25
 datatype property
 (tietotyyppi ominaisuus) . 161,
 167
 DC-sovellus 127
 decode (koodin avaus) 60
 deep web (syvä web) 16
 default graph (oletusgraafi) . 92
 DELETE 108
 DELETE DATA 108
 delimiter (erotin) 54
 denotation (tarkoite) 39
 dereference (ratkominen) .. 111
 DESCRIBE 92
 description logic
 (kuvailulogiikka) 177
 Description Logic Program 182
 disambiguation
 (disambiguointi) 41
 disambiguointi
 (disambiguation) 41
 disjoint property (poissulkeva
 ominaisuus) 168
 DL-safe (DL-turvallisuus) . 182
 DL-turvallisuus (DL-safe) . 182
 DOM 26
 domain (sovellusala) 134
 domain constraint (aluerajoite)
 137
 domain name (aluenimi) ... 56,
 111
 domain ontology

 (aiheontologia) 123, 213
 Dougherty, Dale 23
 DROP 107
 Dublin Core 125, 126, 201
 dumb-down principle
 (dumb-down-periaate) .. 127
 dumb-down-periaate
 (dumb-down principle) .. 127

E

edustapalvelin (reverse proxy)
 57
 eheys (soundness) 186
 ehto (condition) 178
 ehto (premise) 152
 eksistentiaalikkvanttori
 (existential quantifier) ... 155
 element (elementti) 126
 elementti (element) 126
 elävä laboratorio (living
 laboratory) 224
 encoding scheme
 (enkoodausmalli) 127
 enkoodausmalli (encoding
 scheme) 127
 entailment rule 175
 entiteetti (entity) 161
 entiteettihaku (entity search)
 45, 119, 190
 entity (entiteetti) 161
 entity search (entiteettihaku)
 45, 119, 190
 epävarmuus (uncertainty) . 222
 erotin (delimiter) 54
 esittää (render) 57
 etuliite (prefix) 69
 Europeana 204

existential quantifier
 (eksistentiaalikvanttori) . 155
 expert system
 (asiantuntijajärjestelmä) . 222
 explanation (selitys) 186

F

Facebook 15, 22, 24, 119
 faceted search (fasettihaku) . 37
 fact (tosiasia) 151, 178
 fasettihaku (faceted search) . 37
 federated query (federoitu
 kysely) 107
 federated search (hajautettu
 haku) 198
 federoitu kysely (federated
 query) 107
 filter (suodatin) 95
 Finna 204
 FinnONTO 30
 Finto 30
 fragment (fragmentti) 59
 fragmentti (fragment) 59
 function (funktio) 154
 functional property
 (funktionaalinen
 ominaisuus) 169
 functional syntax
 (funktionaalinen syntaksi)
 159
 funktio (function) 154
 funktionaalinen syntaksi
 (functional syntax) 159
 funktionaalinen
 ominaisuus (functional
 property) 169
 fuzzyness (sumeus) 222

G

GAV 199
 GGG 28
 GIT-tunnisteet 57
 Global as View 199
 goal (tavoite) 178
 Google 119
 Google Maps 22, 25
 graafien hallinta (graph
 management) 107
 graafien päivitys (graph
 update) 107
 graafihahmo (graph pattern) 93
 graph management (graafien
 hallinta) 107
 graph pattern (graafihahmo) 93
 graph update (graafien
 päivitys) 107
 ground formula (peruskaava)
 155
 ground term (perustermi) . 155
 group concept (ryhmittelevä
 käsite) 146
 GRRDL-mekanismi 119

H

hajautettu haku (federated
 search) 198
 haku (search) 32
 hash URI 114
 head (pää) 82, 178
 header (HTTP) (otsikko
 (HTTP)) 112
 hidden label (piilonimike) . 141
 hierarchical relation
 (hierarkkinen relaatio) ... 144

hierarkkinen relaatio
 (hierarchical relation) ... 144
 Hipla232
 homonymia (homonymy) ... 35
 homonymy (homonymia) ... 35
 Horn clause (Hornin klausuuli)
 177
 Horn logic (Hornin logiikka)
 177
 Hornin klausuuli (Horn clause)
 177
 Hornin logiikka (Horn logic)
 177
 HTML 56
 HTML5 56
 häntä (rest) 82

I

ICOM126, 213
 identifier (tunniste)55
 IFLA125, 213
 incompleteness (puutteellisuus)
 222
 individual (yksilö) 43, 135
 information retrieval
 (tiedonhaku) 32
 INSERT 108
 INSERT DATA 108
 Instagram119
 instance (yksilö)135
 interactivity
 (vuorovaikutteisuus)20
 internet 15
 interoperability
 (yhteentoimivuus) 23
 interpretation (tulkinta) ...157

inverse functional property
 (käänteisesti
 funktionaalinen
 ominaisuus) 169
 inverse property
 (käänteisominaisuus) 168
 inverted index (käänteinen
 hakemisto) 33
 IRI-identifier (IRI-tunniste) .61
 IRI-tunniste (IRI-identifier) .61
 irrefleksiivinen
 ominaisuus (irrefleksive
 property)169
 irrefleksive property
 (irrefleksiivinen
 ominaisuus) 169
 itsenäinen sovellus (stand-alone
 application) 26

J

JavaScript Object System .. 85
 JSON26, 85
 JSON-LD 85, 119
 järjestetty kokoelma (ordered
 collection)147

K

kaari (arc)52
 kantanimiavaruus (base
 namespace)71
 kardinaliteettirajoite
 (cardinality constraint) .. 126
 key (avainmekanismi) 170
 key-value pair
 (ominaisuus-arvo-pari) ... 85
 kielitunnus (language tag) .. 68
 Kirjasampo30, 202

knowledge base (tietämyskanta)	175
knowledge extraction (tietämyksen irrotus)	208
Knowledge Graph	28
knowledge representation (tietämyksen esittäminen) 52, 124	
knowledge-based validation (tietämysperustainen validointi)	211
kokoelma (collection)	82
kokoelmaluokka (container) .	82
kolmikko (triple)	52
kommentti (comment)	68
kontingentti (contingent) ..	154
koodin avaus (decode)	60
KOS	134
Kulttuurisampo	30, 201
kuvailulogiikka (description logic)	177
kvalifointi (qualification) ..	143
kvanttori (quantifier)	155
kysely (query)	32, 179
kyselyn laajentaminen (query expansion)	40, 43
kysymys-vastaus- järjestelmä (question answering system)	46
käsite (concept)	140
käsitelmä (concept scheme) 140	
käsittelijä (controller)	194
käyttökonteksti (context) ...	36
käänteinen todistusmenetelmä (proof by contradiction)	179
käänteisesti	
funktionaalinen	
ominaisuus (inverse functional property)	169
käänteisominaisuus (inverse property)	168
kääre (wrapper)	199
käänteinen hakemisto (inverted index)	33
L	
laajempi käsite (broader term) 144	
label (nimike)	65
Laney, Dough	24
language tag (kielitunnus) ..	68
lause (statement)	53, 155
lauselogiikka (propositional logic)	152
LAV	199
lemmatization (lemmaus) ...	35
lemmaus (lemmatization) ...	35
Linked Data	21
linked data (linkitetty data) 52	
linked data principles (linkitetyn datan periaatteet)	110
linkitetty data (linked data) 52	
linkitetyn datan periaatteet (linked data principles)	110
list (lista)	82
lista (list)	82
literaali (literal)	63
literaali (logiikassa) (literal (in logic))	177

- literal (in logic) (literaali
(logiikassa)) 177
- literal (literaali) 63
- living laboratory (elävä
laboratorio) 224
- LOAD 107
- Local as View 199
- local name (paikallisenimi) .. 69
- LOD Cloud (LOD-pilvi) 28
- LOD-pilvi (LOD Cloud) 28
- logic programming
(logiikkaohjelmointi) 179
- logical completeness (looginen
täydellisyys) 186
- logiikkaohjelmointi (logic
programming) 179
- looginen täydellisyys (logical
completeness) 186
- luettelointiohje (cataloging
rules) 126
- luokka (class) 42, 135
- luokkahierarkia (class
hierarchy) 136
- lähdekritiikki 235
- M**
- malli (model) 194
- Manchester syntax
(Manchesterin syntaksi) . 160
- Manchesterin syntaksi
(Manchester syntax) 160
- mapping (siltaus) 28, 147
- MapWarper 232
- mark-up language
(merkkauskieli) 56
- Matkailusampo 202
- merkitysoppi (semantics) ... 39
- merkkaus (tag) 56
- merkkauskieli (mark-up
language) 56
- meta-search (metahaku) ... 198
- metadata (metadata) 39
- metadata model
(metadatatamalli) 123
- metadata schema
(metadataskeema) .. 123, 125
- metadatatamalli (metadata
model) 123
- metadataskeema (metadata
schema) 123, 125
- metahaku (meta-search) ... 198
- micro format (mikroformaatti)
116
- microdata (mikrodata) 117
- Microsoft 119
- mikrodata (microdata) 117
- mikroformaatti (micro format)
116
- minting (tunnisteiden
luominen) 58
- model (malli) 194
- monihaku (multi-search) .. 198
- monijoukko (bag) 82
- monotonic logic (monotoninen
logiikka) 183
- monotoninen logiikka
(monotonic logic) 183
- multi-search (monihaku) .. 198
- MuseoSuomi 30
- muuttuja (variable) 93, 154
- N**
- N-Triples notation
(N-Triples-notaatio) 67

N-Triples-notaatio (N-Triples notation) 67
 named graph (nimetty graafi) 92
 namespace (nimiavaruus) ... 69
 narrower term (suppeampi käsite) 144
 NER 208
 nil 82
 nimetty graafi (named graph) 92
 nimetön solmu (blank node) 73
 nimiavaruus (namespace) ... 69
 nimike (label) 65
 nimipalvelu 112
 node (solmu) 52
 notaatio (notation) 67, 143
 notation (notaatio) 67, 143
 näkymä (view) 194

O

O'Reilly, Tim 23
 object (objekti) 53
 object property
 (objektiominaisuus) 161, 167
 objekti (object) 53
 objektiominaisuus (object property) 161, 167
 OCLC 126
 OCR recognition
 (OCR-tunnistus) 206
 OCR-tunnistus (OCR recognition) 206
 OID-tunnisteet 57
 olennaisuus (relevance) 33
 oletusgraafi (default graph) . 92
 ominaisuus (property) 63

ominaisuus–arvo-pari
 (key–value pair) 85
 ominaisuushierarkia (property hierarchy) 137
 ominaisuusketju (property chain) 169
 ominaisuuspolku (property path) 100
 ominaisuusrajoite (property restriction) 165
 ongelmien ratkenta (problem solving) 151, 174
 ontologia (filosofiassa) 39
 ontologia (ontology) ... 39, 133
 ontology (ontologia) ... 39, 133
 open world assumption
 (avoimen maailman oletus) 183
 ordered collection (järjestetty kokoelma) 147
 otsikko (HTTP) (header (HTTP)) 112
 OWL 27, 150
 OWL 1 DL 173
 OWL 2 150
 OWL DL 172
 OWL Full 171
 OWL Lite 173
 OWL-XML 159

P

paikallisinimi (local name) .. 69
 palvelin (server) 26
 path expression (polkulauseke) 100
 persistent identifier (pysyvä tunniste) 62

- personalization (personointi) 36
 personointi (personalization) 36
 peruskaava (ground formula)
 155
 perustermi (ground term) . 155
 PID 62
 piilonimike (hidden label) . 141
 poissulkeva
 ominaisuus (disjoint property)
 168
 polkulauseke (path expression)
 100
 polysemia (polysemy) 35
 polysemy (polysemia) 35
 precision (tarkkuus) 34
 predicate (predikaatti) . 53, 154
 predicate logic
 (predikaattilogiikka) 152, 174
 predikaatti (predicate) . 53, 154
 predikaattilogiikka (predicate
 logic) 152, 174
 preferred label (suositeltu
 nimike) 141
 prefix (etuliite) 69
 premise (ehto) 152
 problem solving (ongelmien
 ratkonta) 151, 174
 procedural interpretation
 (proseduraalinen tulkinta)
 179
 profiili (profile) 172
 profile (profiili) 172
 Prolog 179
 proof by contradiction
 (käänteinen
 todistusmenetelmä) 179
 property (ominaisuus) 63
 property chain
 (ominaisuusketju) 169
 property hierarchy
 (ominaisuushierarkia) ... 137
 property path
 (ominaisuuspolku) 100
 property restriction
 (ominaisuusrajoite) 165
 propositio (proposition) ... 151
 proposition (propositio) ... 151
 propositional logic
 (lauselogiikka) 152
 proseduraalinen tulkinta
 (procedural interpretation)
 179
 provenance (provenienssi) . 215
 provenience (provenienssi) .. 83
 proveniensi (provenance) . 215
 proveniensi (provenience) .. 83
 proxy server (välipalvelin) .. 57
 puutteellisuus (incompleteness)
 222
 pysyvä tunnistus (persistent
 identifier) 62
 pää (head) 82, 178
 pääkäsite (top concept) 140
 päätelmä (conclusion) 152

Q
 qualification (kvalifointi) .. 143
 qualified element 127
 quantifier (kvanttori) 155
 query (kysely) 32, 179
 query expansion (kyselyn
 laajentaminen) 40, 43

question answering system
(kysymys-vastaus-
järjestelmä) 46

R

rajoite (constraint) 137
range constraint (arvorajoite)
137

ratkominen (dereference) .. 111

RBox 159, 177

RDF 27, 55

rdf:Property 137

RDFa 117

rdfs:Class 135

rdfs:Container 82

recall (saanti) 34

redirect (uudelleenohjaus) .. 57

refleksiivinen

ominaisuus (refleksive
property) 168

refleksive property

(refleksiivinen
ominaisuus) 168

reification (reifikaatio) 83

reifikaatio (reification) 83

relational search (yhteyshaku)

36, 45

relevance (olennaisuus) 33

render (esittää) 57

resource (resurssi) 55

REST 26

rest (häntä) 82

resurssi (resource) 55

reverse proxy (edustapalvelin)

57

RIA 26, 193

Rich Internet Application . 193

RIF 27, 185

ristiriita (contradiction) ... 154

Rule Interchange Format

(RIF). 185

rule-based system

(sääntöjärjestelmä) 179

ryhmittelevä käsite (group

concept) 146

S

saanti (recall) 34

Sampo model (Sampo-malli)

200

Sampo-malli (Sampo model)

200

sarjallistaminen (serialization)

54, 67

satisfiable (toteutuva) 154

scalability (skaalautuvuus) .. 24

Schema.org 119

search (haku) 32

security server (turvapalvelin)

57

sekvenssi (sequence) 82

selailu (browsing) 32

selain (browser) 56

SELECT 92

selitys (explanation) 186

semantic disambiguation

(semanttinen

disambiguointi) 209

semantic net (semanttinen

verkko) 27, 52

semantic relation (semanttinen

relaatio) 144

semantic web (semanttinen

web) 27

- semantics (merkitysoppi) ... 39
- semantics (semantiikka) 39
- semantiikka (semantics) 39
- semanttinen relaatio (semantic relation) 144
- semanttinen verkko (semantic net) 27, 52
- semanttinen web (semantic web) 27
- semanttinen
 - disambiguointi (semantic disambiguation) 209
- sequence (sekvenssi) 82
- serialization (sarjallistaminen)
 - 54, 67
- server (palvelin) 26
- SGML 56
- siltaus (mapping) 28, 147
- single page application (yhden sivun sovellus) 26
- sisältöneuvottelu (content negotiation) 113
- skaalautuvuus (scalability) .. 24
- SKOS 27, 139
- SKOS-XL 143
- skos:broader 144
- skos:member 146
- skos:memberList 147
- skos:narrower 144
- skos:narrowerTransitive 145
- skos:OrderedCollection 146, 147
- skos:related 144
- SOAP 25
- solmu (node) 52
- sopeutuvuus (adaptivity) ... 20
- Sotasampo 202
- soundness (eheys) 186
- sovellusala (domain) 134
- SPA 26
- SPARQL 27
- SPARQL Inferencing Notation
 - 180
- SPECTRUM 125
- SPIN 180
- stand-alone application
 - (itsenäinen sovellus) 26
- statement (lause) 53, 155
- statement (väittäjä) 53
- stemmaus (stemming) 36
- stemming (stemmaus) 36
- subdomain (alialue) 57
- subject (subjekti) 53
- subjekti (subject) 53
- subproperty (aliominaisuus)
 - 137
- subquery (alikysele) 106
- suljetun maailman oletus
 - (Closed World Assumption)
 - 183
- sumeus (fuzzyness) 222
- suodatin (filter) 95
- suositeltu nimike (preferred label) 141
- suppeampi käsite (narrower term) 144
- suurdata (big data) 24
- SWRL 182
- sylllogism (sylllogismi) 152
- sylllogismi (sylllogism) 152
- symmetric property
 - (symmetrinen ominaisuus) 168

symmetrinen
 ominaisuus (symmetric
 property) 168
 syntaksi (syntax) 67
 syntax (syntaksi) 67
 syvä web (deep web) 16
 säiliö (container) 81
 sääntöjärjestelmä (rule-based
 system) 179

T

tag (merkkaus) 56
 tarkkuus (precision) 34
 tarkoite (denotation) 39
 tautologia (tautology) 152
 tautology (tautologia) 152
 tavoite (goal) 178
 TBox 159, 177
 tekoäly (Artificial Intelligence)
 152
 tekstihaku (text search) 33
 teoreema (theorem) 152
 term (termi) 155
 termi (term) 155
 TerveSuomi 30
 text search (tekstihaku) 33
 theorem (teoreema) 152
 tiedonhaku (information
 retrieval) 32
 tietämyksen esittäminen
 (knowledge representation)
 52, 124
 tietämyksen irrotus (knowledge
 extraction) 208
 tietämyskanta (knowledge
 base) 175
 tietämysperustainen

validointi (knowledge-based
 validation) 211
 tietotyyppi (data type) 74
 tietotyyppiominaisuus
 (datatype property) 161, 167
 top concept (pääkäsite) 140
 TopBraid Composer 180
 topic modeling (aiheiden
 tunnistus) 210
 tosiasia (fact) 151, 178
 toteutuva (satisfiable) 154
 totuustaulu (truth table) .. 153
 transitiivinen
 ominaisuus (transitive
 property) 169
 transitive property
 (transitiivinen
 ominaisuus) 169
 TriG 84
 triple (kolmikko) 52
 truth table (totuustaulu) .. 153
 tulkinta (interpretation) ... 157
 tunniste (identifier) 55
 tunnisteiden luominen
 (minting) 58
 turvapalvelin (security server)
 57
 Twitter 15, 119

U

UDDI 25
 UNA 163, 184
 uncertainty (epävarmuus) . 222
 unicode-merkistö 61
 Unique Name Assumption
 (UNA) (yksikäsitteisten
 nimien

oletus) 163, 184
 universaalikvanttori (universal
 quantifier)155
 universal quantifier
 (universaalikvanttori)155
 upper ontology (yläontologia)
 218
 URI identifier (URI-tunniste)
 57
 URI schema (URI-skeema) . 57
 URI-skeema (URI schema) . 57
 URI-tunniste (URI identifier)
 57
 URL encode (URL-koodaus) 60
 URL-koodaus (URL encode) 60
 URL-osoite 16, 57
 URN schema (URN-skeema) 60
 URN-skeema (URN schema) 60
 uudelleenohjaus (redirect) .. 57

V

vaihtoehto (alternative)82
 vaihtoehtoinen nimike
 (alternative label) 141
 vakio (constant)154
 validate (validointi)71
 validointi (validate)71
 value assignment (arvosijoitus)
 101
 variable (muuttuja)93, 154
 verkkopalvelu (Web Service) 25
 verkkotunnus56
 view (näkyvä) 194
 visualisaattori (visualizer) .. 70
 visualizer (visualisaattori) .. 70
 vuorovaikutteisuus

(interactivity) 20
 väittäjä (statement) 53
 välipalvelin (proxy server) .. 57

W

Watson-järjestelmä 47
 Web 2.0 20
 Web 3.0 27
 Web of Data 21
 Web Service (verkkopalvelu) 25
 WGS 84232
 Wikipedia 22
 wrapper (kääre) 199
 WSDL 25
 WWW:n idea 15

X

XML56

Y

Yahoo119
 Yandex119
 yhden sivun sovellus (single
 page application) 26
 yhteentoimivuus
 (interoperability) 23
 yhteyshaku (relational search)
 36, 45
 yksikäsitteisten nimien
 oletus (Unique Name
 Assumption (UNA)) 163, 184
 yksilö (individual) 43, 135
 yksilö (instance)135
 yläontologia (upper ontology)
 218
 YouTube 22, 24
 YSA 139

Kirjoittaja

Eero Hyvönen¹ on mediatekniikan professori Aalto-yliopiston tietotekniikan laitoksella ja Helsingin yliopiston digitaalisten ihmistieteiden keskuksen HELDIGin johtaja. *Semanttinen web* -teos perustuu kirjoittajan omiin kokemuksiin suomalaisen semanttisen webin tietoinfrastruktuurin ja sovellusten kehittämisessä 2000-luvun alusta lähtien. Hänen uraauurtava tutkimus- ja kehitystyönsä on saanut lukuisia kansainvälisiä tunnustuksia ja kotimaassa muun muassa Tiedonjulkistamisen valtionpalkinnon.



¹ <http://seco.cs.aalto.fi/u/eahyvone/>