SPARQL

Timo Pitkänen

Seminaari
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 1. marraskuuta 2015

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author

Timo Pitkänen

Työn nimi — Arbetets titel — Title

SPARQL

Oppiaine — Läroämne — Subject

Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Seminaari	1. marraskuuta 2015	12

Tiivistelmä — Referat — Abstract

Avainsanat — Nyckelord — Keywords

SPARQL

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Johdanto

Vuonna 1999 World Wide Web Consortium (W3C) standardoima RDF -malli (Resource Description Framework) on yksi semanttisen webin peruspilareista. RDF on malli tiedon vaihtoon sovellusten välillä, erityisesti web-ympäristössä. On olemassa useita RDF -tietomallin kyselykieliä, joilla tietomalliin voidaan tehdä hakuja, muutoksia ja lisäyksiä. Suosituin näistä kyselykielistä on SPARQL (SPARQL Protocol and RDF Query Language).

SPARQL-kielestä tuli W3C:n virallinen standardi RDF-kyselykieleksi vuonna 2008. Vuonna 2009 W3C perusti oman työryhmänsä kehittämään SPARQL-kieltä ja ryhmä julkaisi ensimmäisen version SPARQL v1.1 -kielestä vuonna 2009 ja nykymuotoonsa kieli valmistui vuonna 2012. W3C asetti version 1.1 viralliseksi standardiksi vuonna 2013.

Useat palvelut tarjoavat SPARQL-endpoint web-rajapintoja, joihin voidaan lähettää SPARQL-kyselyitä. Tunnetuimpia SPARQL-endpoint-rajapintoja tarjoavia sovelluksia on DBpedia, joka on RDF-muotoinen vastine wikipedialle.

Tämä työ esittelee SPARQL-kielen kyselyiden rakenteen ja sen kuinka kyselyitä käytetään. Tämän työn esitykset ja esimerkit perustuvat kahteen lähteeseen: W3C:n viralliseen SPARQL v1.1 määrittelydokumenttiin [HSP13] sekä Bob DuCharmen ”Learning SPARQL-kirjaan” [DuC13].

2 RDF

RDF on tietomalli, jossa tallennettu tieto esitetään kolmiosaisina lausumina – kolmikoina (triples). Nämä kolmikot koostuvat subjektista, predikaatista ja objektista, joilla kuvataan jonkin resurssin ominaisuutta.

Resurssilla tarkoitetaan kohdetta, joka identifioidaan URI-tunnisteella. Subjekti esittää kuvailtavan resurssin tunnusta. Predikaatti esittää subjektin ominaisuutta ja objekti esittää resurssin ominaisuuden saamaa arvoa. Seuraavassa Turtle-notaatiolla talletetuista RDF-kolmikoista, jotka toimivat yksinkertaisena osoitekirjana.

```
#filename: ex002.ttl
@prefix ab: <http://learningsparql.com/ns/addressbook#> .

ab:richard ab:homeTel "(229) 276-5135" .
ab:richard ab:email "richard49@hotmail.com" .

ab:cindy ab:homeTel "(245) 646-5488" .
ab:cindy ab:email "cindym@gmail.com" .

ab:craig ab:homeTel "(194) 966-1505" .
ab:craig ab:email "craigellis@yahoo.com" .
ab:craig ab:email "c.ellis@usairwaysgroup.com" .
```

Tässä RDF esimerkissä prefix-termillä määritellään resurssien nimiavaruuden URI-tunniste, jolloin koko osoitetta ei tarvitse toistaa, vaan voidaan käyttää lyhennelmää, tässä "ab:". Subjekti ja predikaatti määritellään merkkijonojen sijaan resursseiksi, jotta vältyttäisiin väärintulkinnalta ja samannimisten termien mahdollisilta konflikteilta.

Tässä RDF-tietomallissa on seitsemän kolmikkoa, jotka kuvaavat kolmea resurssia. Resurssin ab:richard kolmikon *ab:richard ab:homeTel "(229) 276-5135"* vastine luonnollisella kielellä olisi: Richardin puhelinnumero on (229) 276-5135.

3 SPARQL-kysely

SPARQL on kyselykieli, jolla voidaan tehdä hakuja, muutoksia ja lisäyksiä RDF-tietomalliin. SPARQL-kyselyissä määritellään RDF-tietomallista alijoukko, josta haetaan halutut tiedot. Havainnollistetaan tätä seuraavalla SPARQL-kyselyllä, joka hakee kaikki Craigin sähköpostiosoitteet edellisen luvun RDF-tietomallin mukaisesta osoitekirjasta.

```
#filename: ex003.rq

PREFIX ab: <http://learningsparql.com/ns/addressbook#>

SELECT ?craigEmail
WHERE
{ ab:craig ab:email ?craigEmail . }
```

Edellä olevassa kyselyssä määritellään RDF-tietomallista alijoukko WHERE lausekkeessa olevalla kolmikolla. Alijoukosta, joka tässä tapauksessa käsittää kaikki kolmikot, joiden subjekti on "ab:craig"ja "ab:email", haetaan SELECT lausekkeessa määritellyt tiedot. Tässä tapauksessa SELECT-lausekkeessa määriteltyyn *?craigEmail*-muuttujaan haetaan kaikki WHERE-lausekkeen määrittämän alijoukon kolmikoiden objektien arvot. Näin ollen edellä olevasta kyselystä saadaan vastaukseksi kaksi sähköpostiosoitetta:

craigEmail
c.ellis@usairwaysgroup.com
craigellis@yahoo.com

SPARQL-haut perustuvat siis kyselyn WHERE-lausekkeessa annettuihin graafimalleihin(*Graph Pattern*). Graafimalleilla määritellään tietyt ehdot, jotka täyttävistä RDF-tietomallin kolmikoista muodostetaan alijoukko. SPARQL-kyselyn graafimallit ovat joukko kolmikoita, joille voidaan antaa tiettyjä ehtoja, ominaisuuksia tai rajoituksia. Edellä olevassa esimerkissä WHERE-lausekkeen sisällä oleva graafimalli sisältää yhden kolmikon, jonka

objektin paikalla on muuttuja. Muuttuja graafimallin kolmikossa tarkoittaa eräänlaista jokerikorttia, jolloin haettavista kolmikoista hyväksytään muuttujan paikalle mikä tahansa arvo tai resurssi.

Muuttujan ei tarvitse graafimallissa olla vain objektin paikalla, vaan sen voi sijoittaa myös niin subjektin kuin predikaatinkin paikalle. Usein graafimallin kolmikoissa onkin useampi muuttuja. Seuraavalla kyselyllä haetaan graafimallin määrittämästä alijoukosta (kaikki kolmikot, joiden subjekti on ab:cindy) kaikki predikaatit ja objektit, eli subjektin ominaisuudet ja niiden arvot.

```
 #filename: ex010.rq
 PREFIX ab: <http://learningsparql.com/ns/addressbook#>
SELECT ?propertyName ?propertyValue
WHERE
{ ab:cindy ?propertyName ?propertyValue . }
```

Löydettyjen kolmikoiden subjektit ja objektit sidotaan SELECT-lausekkeessa määriteltyihin samannimisiin muuttujiin, jolloin kyselyn vastaukseksi saadaan:

propertyName	propertyValue
ab:homeTel	(245) 646-5488
ab:email	cindym@gmail.com

Graafimallit voivat toki sisältää useampiakin kolmikoita. Graafimallin kolmikot evaluoidaan järjestyksessä vasemmalta oikealle, jolloin edellisen kolmikon muuttujien saamia arvoja käytetään seuraavan kolmikon vertailussa.

```
# filename: ex019.rq
PREFIX a: <http://learningsparql.com/ns/addressbook#>
SELECT ?propertyName ?propertyValue
WHERE
{
?person a:firstName "Cindy" .
?person a:lastName "Marshall" .
?person ?propertyName ?propertyValue .
}
```

Edellä olevassa kyselyssä alijoukon määrittävä graafimalli käsittää kolme kolmikko. Tässä tapauksessa graafimallin ehdot täyttävät kaikki resurssit, joilla on ominaisuudet a:firstName arvolla "Cindy"ja a:lastName arvolla "Marshall". Graafimallin viimeinen kolmikko sisältää pelkästään muuttujia, joten resurssin, jotka täyttävät kahden ensimmäisen kolmikon ehdot, kaikki loputkin ominaisuudet täyttävät ehdot. Tällöin graafimallin määrittelemä alijoukko sisältää kahden ensimmäisen kolmikon ehdot täyttävien resurssien kaikki kolmikot (= kaikki ominaisuudet).

SELECT-lausekkeessa on siis määritelty mitkä tiedot haluamme graafimallin määrittelemästä joukosta. Tässä tapauksessa *?propertyName*- ja *?propertyValue* -muuttujat tarkoittavat alijoukon resurssien kaikkia ominaisuuksia ja niiden arvoja. Täten edellä olevan kyselyn vastausjoukoksi saadaan:

propertyName	propertyValue
ab:homeTel	(245) 646-5488
ab:email	cindym@gmail.com
ab:lastName	Marshall
ab:firstName	Cindy

3.1 OPTIONAL

SPARQL-kyselyiden graafimallien kolmikot toimivat ryhmänä, jolloin kaikkia graafimallin kolmikoita vastaavat kolmikot on löydyttävä haettavasta tietomallista. Tämä ei aina ole toivottavaa, esimerkiksi tilanteessa, jossa haluaisimme listata Luvun 2 osoitekirjaesimerkistä kaikkien henkilöiden nimet ja heidän puhelinnumeronsa. Osoitekirjaesimerkissä kuitenkin joiltakin resursseilta puuttuu puhelinnumero-ominaisuus, jolloin seuraavan kyselyn graafimalli ei täsmää niiden resurssien kanssa, joilla ei ole puhelinnumeroa määriteltynä, ja kyselyn tuloksiin valitaan vain sellaiset resurssit, joilla on puhelinnumero määriteltynä:

```
# filename: ex057.rq
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
SELECT ?first ?last ?workTel
WHERE
{
  ?s ab:firstName ?first .
  ?s ab:lastName ?last .
  ?s ab:workTel ?workTel .
}
```

first	last	workTel
Craig	Ellis	(245) 315-5486

SPARQL-kieli tarjoaa kuitenkin tavan, jolla graafimallien kolmikoita voi asettaa valinnaisiksi. Asettamalla graafimallissa kolmikko OPTIONAL-lausekkeen sisään, kyseinen kolmikko jätetään noteeraamatta, mikäli sen mukainen kolmikko puuttuu resurssilta. Lisäämällä edelliseen kyselyn graafimallin viimeinen kolmikko OPTIONAL-lausekkeen sisään:

```
# filename: ex057.rq
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
SELECT ?first ?last ?workTel
WHERE
{
  ?s ab:firstName ?first ;
  ab:lastName ?last .
  OPTIONAL
  { ?s ab:workTel ?workTel . }
}
```

Kyselyn tulokseksi saadaan:

first	last	workTel
Craig	Ellis	(245) 315-5486
Cindy	Marshall	
Richard	Mutt	

3.2 UNION

SPARQL tarjoaa myös tavan määritellä useampia graafimalleja, joiden määrittelemät resurssit yhdistetään yhdeksi joukoksi. Tällä tavoin voidaan resursseista hakea vaihtoehtoisia ominaisuuksia. Graafimallit yhdistetään SPARQL-kielessä UNION-lausekkeilla. Seuraavassa kyselyssä haetaan annetusta datasta henkilöiden nimet ja heidän soittimensa sellaisilta henkilöiltä, jotka soittavat saksofonia ja/tai trumpettia.

RDF-tietomalli:

```
# filename: ex100.ttl
@prefix ab: <http://learningsparql.com/ns/addressbook#> .
@prefix d:<http://learningsparql.com/ns/data#> .
d:i0432 ab:firstName "Richard" ;
   ab:lastName "Mutt" ;
   ab:instrument "sax" ;
   ab:instrument "clarinet" .

d:i9771 ab:firstName "Cindy" ;
   ab:lastName "Marshall" ;
   ab:instrument "drums" .

d:i8301 ab:firstName "Craig" ;
   ab:lastName "
   Ellis" ;
   ab:instrument "trumpet" .
```

Kysely:

```
# filename: ex101.rq
PREFIX ab: <http://learningsparql.com/ns/addressbook#>

SELECT ?first ?last ?instrument
WHERE
{
  { ?person ab:firstName ?first ;
  ab:lastName ?last ;
  ab:instrument "trumpet" ;
  ab:instrument ?instrument .
  }
UNION
  { ?person ab:firstName ?first ;
  ab:lastName ?last ;
  ab:instrument "sax" ;
  ab:instrument ?instrument .
  }
}
```

first	last	instrument
Craig	Ellis	trumpet
Richard	Mutt	clarinet
Richard	Mutt	sax

4 Kyselyn rajaaminen

SPARQL-kieli tarjoaa useita työkaluja, joilla voidaan rajata kyselyiden tuloksia. Tässä luvussa käydään läpi yleisimpiä tapoja, joilla voidaan asettaa kyselyn tuloksiin ehtoja tai poistaa tiettyjä resursseja tuloksista.

4.1 FILTER

FILTER termillä voidaan suodattaa kyselyn tulosjoukosta tuloksia, jotka eivät täytä annettuja ehtoja. FILTER-lausekkeessa annetaan jokin totuusarvon palauttava lauseke, joka määrittää suodatetaanko graafimallin mukainen kolmikko tulosjoukosta. SPARQL-kielessä on monia erilaisia funktioita, joilla voidaan arvioida muuttujia, joiden perusteella kolmikoiden suodattaminen tuloksista suoritetaan. Tällaisia funktioita ovat muun muassa: regex(), jolla voi tarkistaa löytyykö jokin merkkijono funktiolle parametrina annetusta kolmikon muuttujasta ja isUri(), jolla voi tarkistaa, onko funktiolle parametrina annettu muuttuja URI-osoite. Seuraavassa kyselyssä FILTER lausekkeessa tarkistetaan, onko resurssin ominaisuuden dm:cost arvo alle 10.

```
# filename: ex105.rq
PREFIX dm: <http://learningsparql.com/ns/demo#>
SELECT ?s ?cost
WHERE
{
  ?s dm:cost ?cost .
  FILTER (?cost < 10)
}
```

4.2 FILTER (NOT) EXISTS

FILTER EXISTS ja FILTER NOT EXISTS-suodattimilla voi tarkistaa löytyykö kyseisille suodattimille annettu kolmikko kohdedatasta. Seuraavassa esimerkissä tarkastetaan FILTER EXISTS- ja FILTER NOT EXISTS -suodattimien toimintaa samalla kohdedatalla ja samalla kyselyllä.

Data:

```
@prefix : <http://example/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

:alice rdf:type foaf:Person .
:alice foaf:name "Alice" .
:bob rdf:type foaf:Person .
```

Seuraavassa FILTER NOT EXISTS-esimerkissä kyselyn graafimallin ensimmäiseen kolmikkoon löytyy kohdedatasta kaksi vastaavaa kolmikkoa, mutta FILTER NOT EXISTS-lausekkeen sisällä olevalle kolmikolle löytyy vastaava kolmikko resurssille :alice. Tällöin FILTER NOT EXISTS evaluoidaan todeksi, ja resurssi :alice suodatetaan tuloksista.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?person
WHERE
{
   ?person rdf:type foaf:Person .
   FILTER NOT EXISTS { ?person foaf:name ?name }
}
```

person
<http://example/bob>

Vastaavasti FILTER EXISTS evaluoidaan todeksi resurssille :alice ja epätodeksi resurssille :bob. Joten vain resurssi :alice vastaa kyselyn graafimallia.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?person
WHERE
{
   ?person rdf:type foaf:Person .
   FILTER EXISTS { ?person foaf:name ?name }
}
```

person
<http://example/alice>

4.3 MINUS

SPARQL-kielen MINUS-lausekkeella voidaan etsiä sellaisia resursseja, joilta puuttuu jokin ominaisuus. Seuraavassa kyselyssä haetaan kaikki osoitekirjasta kaikki henkilöt, joilla ei ole puhelinnumero-ominaisuutta ts. resurssin kolmikoilta puuttuu subjekti ab:workTel.

```
# filename: ex068.rq
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
SELECT ?first ?last
WHERE
{
  ?s ab:firstName ?first ;
  ab:lastName ?last .
  MINUS { ?s ab:workTel ?workNum }
}
```

4.4 LIMIT

Vastausten määrää voidaan rajoittaa LIMIT-termillä. LIMIT-termillä voidaan kyselyn vastausjoukosta määrittää kuinka monta vastausta halutaan. Jos vastausjoukkoa ei ole järjestetty, voi LIMIT-termillä rajoitettu vastausjoukko näyttää sattumanvaraisesti valitulta, sillä RDF-malli ei ota kantaa kolmikoiden järjestykseen. Tulosjoukon voi järjestää SPARQL-kyselyssä ORDER BY termillä. Seuraava kysely palauttaa siis vain tulosjoukon 20 ensimmäistä vastausta, vaikka niitä olisikin enemmän.

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

SELECT ?name
WHERE { ?x foaf:name ?name }
LIMIT 20
```

4.5 OFFSET

OFFSET-termillä määritetään kuinka monen vastausrivin yli hypätään, eli kuinka monta riviä jätetään vastauksesta pois tulosjoukon alusta laskien. Myös OFFSET-termiä käytettäessä poistettavat rivit voivat näyttää satunnaiselta, mikäli kyselyn tulosjoukkoa ei ole järjestetty. OFFSET-termiä voidaan käyttää LIMIT-termin kanssa, jolloin voidaan valita vastausjoukoksi pienempi osajoukko keskeltä kyselyn vastausjoukkoa. Seuraavan kyselyn tulosjoukko käsittää vain viisi tulosta alkaen kymmenennestä tuloksesta.

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

SELECT ?name
WHERE { ?x foaf:name ?name }
ORDER BY ?name
LIMIT 5
OFFSET 10
```

4.6 DISTINCT

DISTINCT termillä voidaan tulosjoukosta poistaa samanlaiset vastaukset, eli vastaukset jossa tulosrivin kaikki haetut muuttujat ovat samoja. DISTINCT termillä poistetut termit poistetaan vastausjoukosta ennen LIMIT - ja OFFSET -termien evaluointia. Seuraavassa kyselyssä DISTINCT-termi poistaa tulosjoukosta kaikki samannimiset henkilöt

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?name
WHERE { ?x foaf:name ?name }
```

4.7 ORDER BY

ORDER BY lausekkeella voidaan määrätä vastausjoukon järjestys. Seuraava ORDED BY-lauseke muodostuu muuttujasta, jonka mukaan vastaus järjestetään. Vastauksen järjestys voidaan määrittää nousevaksi järjestykseksi ASC-komennolla tai laskevaksi DESC-komennolla.

```
PREFIX : <http://example.org/ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?name
WHERE { ?x foaf:name ?name ; :empId ?emp }
ORDER BY DESC(?emp)
```

5 Koostefunktiot ja Ryhmittely

SPARQL tarjoaa erilaisen kokoelman koostefunktioita, joilla voi tuottaa laskennallista dataa vastausjoukosta. Koostefunktiot on lisätty SPARQL-kieleen vasta versioon 1.1, josta löytyy seuraavat koostefunktiot:

COUNT: laskee arvojen lukumäärän.

SUM: laskee arvojen summa.

MIN: laskee pienimmän arvon.

MAX: laskee suurimman arvon.

AVG: laskee arvojen keskiarvon.

Seuraava esimerkki havainnollistaa koostefunktioiden käyttöä. Esimerkissä halutaan laskea kirjojen hintojen summa. Kyselyssä etsitään graafimallin mukainen tulosjoukko, jonka kaikkien löydettyjen ?lprice-muuttujien arvot lasketaan summafunktion avulla yhteen. Koostefunktion tulos talletetaan nimettyyn muuttujaan.

```
PREFIX : <http://books.example/>
SELECT (SUM(?lprice) AS ?totalPrice)
WHERE {
  ?org :affiliates ?auth .
  ?auth :writesBook ?book .
  ?book :price ?lprice .
}
```

totalPrice
21

5.1 GROUP BY

Havainnollistetaan GROUP BY-lausekkeen käyttöä koostefunktio-osion esimerkillä, jossa haetaan kaikkien löydettyjen kirjojen yhteishinta. Lisäämällä kyselyyn GROUP BY-lauseke, voidaan laskea kirjojen yhteissumma jokaiselle subjektille erikseen, tässä tapauksessa kirjat on ryhmitelty ?org-muuttujan mukaan. Nyt saamme koostefunktiolle yhden vastauksen sijaan, jokaisen subjektin mukaan ryhmitellyille tulosjoukoille omat summafunktion tulokset.

```
PREFIX : <http://books.example/>
SELECT (SUM(?lprice) AS ?totalPrice)
WHERE {
  ?org :affiliates ?auth .
  ?auth :writesBook ?book .
  ?book :price ?lprice .
}
GROUP BY ?org
```

5.2 HAVING

HAVING toimii kuten FILTER-operaatio, mutta GROUP BY-operaation määrittelemille ryhmille. Havainnollistetaan HAVING-operaatiota GROUP BY-osion kyselyllä, mutta tuloksista poistetaan HAVING-operaatiolla ne ryhmät, joiden summafunktion tulos on alle 100.

```
PREFIX : <http://books.example/>
SELECT (SUM(?lprice) AS ?totalPrice)
WHERE {
  ?org :affiliates ?auth .
  ?auth :writesBook ?book .
  ?book :price ?lprice .
}
GROUP BY ?org
HAVING(SUM(?lprice) < 100)
```

6 Alikyselyt

Alikyselyt ovat SPARQL-kyselyitä toisten kyselyiden sisällä. Alikyselyitä käytetään hajottamaan monimutkaiset kyselyt pienempiin ja hallittavampiin osiin, ja sen avulla voi myös yhdistää tietoja useasta kyselystä yhdeksi vastausjoukoksi. SPARQL-kielen evaluointi järjestyksessä alikyselyt evaluoidaan ensiksi, jolloin niiden tuloksia voidaan käyttää ulommissa kyselyissä. Näin ollen alikyselyn käyttämät muuttujat ovat käytössä uloimmille kyselyille, mutta ei toisin päin.

```
# filename: ex137.rq
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
SELECT ?lastName ?courseName
WHERE
{
  {
    SELECT ?lastName
    WHERE { ?student ab:lastName ?lastName . }
  }
{
  }
    SELECT ?courseName
    WHERE { ?course ab:courseTitle ?courseName . }
  }
```

7 Yhteenveto

SPARQL on RDF-tietomallin kyselykieli, joka perustuu graafimallien määrittelemien resurssien ja niiden ominaisuuksien joukkoon, josta valitaan halutut tiedot. SPARQL-kyselyn graafimalleihin voidaan tuoda joustavuutta OPTIONAL ja UNION-lausekkeilla.

SPARQL-kielessä kyselyiden tulosten rajaamiseen on useita keinoja, kuten suodattaminen ja tulosten poistaminen tietyin ehdoin.

SPARQL-tarjoaa myös useita kooste- ja järjestysfunktioita, joilla RDF-datasta voidaan laskea erilaisia arvoja kuten eri resurssien arvojen summia tai koostefunktioilla voidaan etsiä eri resurssien ominaisuuksien pienimmän arvon.

Lähteet

[DuC13] DuCharme, Bob: *Learning Sparql*. O'Reilly Media, Inc., 2013.

[HSP13] Harris, Steve, Seaborne, Andy ja Prud'hommeaux, Eric: *SPARQL 1.1 query language*. W3C Recommendation, 21, 2013.