LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

PRAKTISKI VIZUĀLI DATU VAICĀJUMI

MAĢISTRA DARBS

Autors: Toms Kirtovskis
Stud. apl. Nr. tk17025
Darba vadītājs: profesors Dr. dat. Kārlis Čerāns

RĪGA 2019

Anotācija

Maģistra darbā ir aprakstīts par saistītajiem datiem, RDF, SPARQL vaicājumiem, ViziQuer rīku, kas paradzēts praktiski vizuāli datu vaicājumu veidošanai, tā darbības būtību, pamatidejām, kā arī piekļuves punktiem, kuros iespējams iegūt datus izmantojot SPARQL vaicājumus. Rīku, ar kura palīdzību iespējams veikt vizuālus datu vaicājumus, šobrīd īsti nav, tāpēc, ņemot vērā to, ka vizuāli sastādīt vaicājumus ir ērtāk kā pārzināt dažādas sintakses, ir svarīgi veikt izpēti un paātrināt attīstību šajā virzienā. Izpētot pieejamos resursus par augstāk minētajām tēmām, veicot praktiskus izmēģinājumus un aplūkojot iegūtos rezultātus, tika izdarīti vairāki secinājumi par ViziQuer rīka darbību un iespējamajiem uzlabojumiem.

Atslēgvārdi: ViziQuer, RDF, Linked Data, SPARQL

Abstract

This paper contains information about linked data, RDF, SPARQL, ViziQuer tool, which is designated to visual data query creation, basic principles of tool itself and endpoints which can be used to explore containing data by using SPARQL queries. There isn't any complete tool which can provide functionality for visual data queries and using of such queries could be much easier than becoming an expert of various querying languages. Because of that it is important to research and to speed up the development in this area. In result of researching resources about those topics and running practical tests and compiling endresults, a list of several conclusions about ViziQuer tool and possible upgrades to it was made.

Title of paper: Practial visual data query

Keywords: ViziQuer, RDF, Linked Data, SPARQL

AUTORREFERĀTS

Veidojot šo darbu, autors piedalījās ViziQuer tiešsasites rīka attīstīšanā un dažādu SPARQL piekļuves punktu izpētē, lai apkopotu informāciju par to saderību ar ViziQuer tiešsasites rīku, kā arī ar SPARQL piekļuves punktu datu shēmu iegūšanas rīku, kas ir svarīga sastāvdaļa vizuālu SPARQL vaicājumu izveidei. Tā rezultātā ir iespējams sekmīgāk veikt tālākus pētījumus un uzlabojumus saistībā ar ViziQuer rīka pielietošanu SPARQL piekļuves punktu pārlūkošanai un datu iegūšanai no šiem piekļuves punktiem.

Autors, apkopojot darba teorētiskās nodaļas, ir izmantojis World Wide Web Consortium (W3C) publikācijas, HTTP saziņas protokolu un URI aprakstošo dokumentāciju, kā arī vikipēdijā atrodamos ierakstus par SPARQL un ar to saistītajiem terminiem.

Gan aplūkotās problēmas, gan risinājumi darbā ir detalizēti aprakstīti. Darba gaitā autora izstrādātās koda rekomendācijas ViziQuer rīka darbības uzlabošanai ir pietiekami testētas un pieejamas kopā ar ViziQuer kodu publiskajā Git koda glabātuvē [16].

Izstrādājot šo darbu, autors turpināja kursa darba ietvaros iesāktos pētījumus par SPARQL un piekļuves punktiem, kuros tiek izmantota šī vaicājumu valoda, tos papildinot ar detalizētāku analīzi, kā arī izstrādājot koda rekomendācijas ViziQuer tiešsasites rīka darbības uzlabošanai.

Iegūtie rezultāti jau tiek pielietoti praksē. Darba teksts ir pārlasīts un atrastās ievades, komatu un vārdu locījumu kļūdas ir izlabotas. Autors ir pārliecinājies, ka darbā nav pareizrakstības kļūdu un ka darbā ir izmantota latviešu valodā oficiāli pieņemtā nozares terminoloģija. Visas idejas, formulējumi, attēli utt., kas aizgūti no citiem autoriem, ir atzīmēti ar attiecīgām literatūras atsaucēm un tie darba teksta gabali, kas ir burtisks tulkojums vai tuvs pārstāsts no kāda viena literatūras avota, ir atzīmēti kā teksta aizguvumi.

SATURA RĀDĪTĀJS

IEVADS

SPARQL piekļuves punkti ieņem nozīmīgu lomu Semantiskajā Tīmeklī, jo tie lietotājiem un programmatūrai nodrošina piekļuvi reāliem datiem. Tas nozīmē to, ka izstrādātājiem un lietotājiem ir svarīgi saprast SPARQL piekļuves punktā esošo datu struktūru, lai varētu tur pieejamos datus efektīvi izmantot. Šobrīd tas rada pamanāmu problēmu, jo šajā virzienā vēl nav paveikts pietiekoši daudz fundamentāla darba ar SPARQL piekļuves punktu shēmu definīcijām un tajos pieejamo datu dokumentēšanu. Šobrīd SPARQL piekļuves punkti ir izstrādāti tikai kā pieejas punkti, kuros, izmantojot SPARQL vaicājumus, ir iespējams iegūt semantiski strukturētus datus. Būtu nepieciešams paveikt daudz darba saistībā ar SPARQL piekļuves punktu pārvaldes procesiem, to dokumentāciju un tajos esošo datu uzbūves principiem. Piemēram, datubāzē, kas veidota pēc SQL principiem programmētājs var samērā vienkārši iegūt un apskatīt datu bāzē esošo datu shēmu, tādējādi atvieglojot izstrādi sistēmai, kas darbosies ar šo datu bāzi. Ir samērā pašaprotami, ka SPARQL piekļuves punkta lietotājam būtu daudz ērtāk un vienkāršāk darboties ar nezināmu piekļuves punktu datiem, ja lietotājam būtu pieejams vairāk informācijas par reālo datu shēmu (ontoloģiju), pēc kuras konkrētā piekļuves punkta instances dati ir organizēti.

ViziQuer ir izstrādāts, balstoties uz "Model Driven Architecture" tehnoloģijām, kas sniedz iespēju tam darboties samērā elastīgi un pieslēgties jebkuram SPARQL piekļuves punktam. ViziQuer rīks ir izstrādāts "ajoo" platformā. Tā nodrošina iespēju viegli veikt darbības ar grafiskām diagrammām, kas ir ļoti svarīgs elements, lai varētu ērti veidot vizuālus datu vaicājumus un grafiski attēlot SPARQL piekļuves punktos esošās datu shemas.

1. RDF

RDF Resursu aprakstīšanas ietvars ir viena no galvenajām sastāvdaļām Saistīto datu izveidē un tā nodrošina vispārīgu uz grafiem bāzētu modeli dažādu vienumu aprakstīšanai, tajā skaitā to relācijām ar citiem vienumiem. RDF dati var tikt aprakstīti vairākos dažādos veidos, kas tiek saukti par serializācijām. RDF serializāciju piemēri ir RDF/XML, Notācija 3 (Notation-3, N3), Turtle, N-triples, RDFa un RDF/JSON [2]

Resursu aprakstīšanas ietvars (angļu Resource Description Framework) jeb RDF ir daļa no World Wide Web Konsorcija (W3C) izveidotajām specifikācijām, ko sākotnēji tika plānots izmantot kā datu modeli metadatu apraksstīsanai. Tā pielietošana ir tikusi paplašināta līdz vispārīgai metodei, ar ko veidot konceptuālu aprakstu vai informācijas modelēšanu, kas tiek izmantota tīmekļa resursos, izmantojot dažādus datu serializācijas formātus un to pieraksta sintakses. RDF tiek lietots arī dažādu organizāciju apkopotās informācijas pārvaldības lietotnēs. RDF tika pieņemts kā W3C rekomendācija 1999. gadā. RDF 1.0 specifikācija tika publicēta 2004. gadā, un RDF 1.1 specifikācija tika publicēta 2014. gadā. [4]

1.1. RDF modelis

Tas ir balstīts uz pamatideju, kas nosaka izteikumu veidošanu par konkrētiem tīmekļa resursiem, tos veidojot formā subjekts - predikāts - objekts, kas tiek saukti par trijniekiem. Subjekts apzīmē konkrēto resursu, un predikāts apzīmē konkrētā resursa aspektus vai īpašības, kā arī apraksta attiecības starp konkrēto subjektu un objektu. Piemēram, lai RDF kā trijnieku aprakstītu izteikumu “Debesis ir zilas”, to varētu sadalīt sekojoši: “debesis” tiek pieņemts kā subjekts, “ir” šajā piemērā ir predikāts un “zilas” ir objekts. Tātad RDF atšķirībā no tipiskā entītija - atribūts - vērtība modeļa objektorientētajā pieejā (entītija - debesis, atribūts - krāsa, vērtība - zila), tiek lietots subjekts, nevis objekts (entītija). RDF ir abstrakts modelis ar vairākiem serializācijas formātiem (piemēram, failu formātiem), kā rezultātā konkrētu resursu vai trijnieku kodējums dažādos formātos var būt atšķirīgs. Resursu aprakstīšanas mehānisms ir nozīmīga W3C Semantiskā Tīmekļa aktivitāšu daļa, tā ir evolucionāra “World Wide Web” fāze, kurā automatizēta programmatūra var uzglabāt, lietot un apmainīties ar “mašīnām izlasāmu” informāciju, kas atrodama tīmeklī, tādējādi nodrošinot to, ka lietotāji var apstrādāt informāciju ar daudz augstāku efektivitāti un noteiktību. RDF vienkāršais datu modelis un tas,

ka to izmantojot var aprakstīt dažādus abstraktus konceptus, ir novedis pie tā pastiprinātas izmantošanas dažādu organizāciju apkopotās informācijas pārvaldības lietotnēs, kas nav saistītas ar Semantiskā Tīmekļa aktivitātēm. REF izteikumu apkopojums būtībā var tikt reprezentēts ar orientētu grafu struktūru. Tas nodrošina to, ka RDF datu modelis ir labāk piemērots noteiktu veidu informācijas reprezentācijai, nekā citi relāciju vai ontoloģiskie modeļi. Neskatoties uz to, praksē RDF dati bieži tiek uzglabāti relāciju datubāzēs vai citos pielāgotos veidos, kas tiek saukti arī par Trijnieku-noliktavām (Tripletstores) vai Četrinieku-noliktavām, ja katram RDF trijniekam tiek uzglabāta arī grafu aprakstoša informācija (named graph). [4]

1.2. RDF attīstība

Sākotnējais RDF projekts tika plānots kā "vispārīgi neitrālu un no operētājsistēmas neatkarīgu metadatus aprakstošas sistēmas izveide", balstoties uz W3C Platformas Interneta Satura Izvēlei (angļu Platform for Internet Content Selection jeb PICS), kā agrīna tīmekļa satura apzīmējumu sistēma, bet izmaiņas projektā radīja idejas no Dublin Core un Meta Content Framework (MCF), ko 1995. - 1997.gadā izstrādāja Ramanathan V. Guha no Apple un Tim Bray no Netscape. Pirmā publiskā RDF versija parādījās 1997.gada oktobrī un to izdeva W3C darba grupa, kurā darbojās pārstāvji no IBM, Microsoft, Netscape, Nokia, Reuters, SoftQuad, un Mičiganas Universitātes. 1999. gadā W3C publicēja pirmo ieteicamo RDF specifikāciju kopā ar modeļu un sintakses specifikāciju ("RDF M&S"). Tas aprakstīja RDF datu modeļus un XML serializāciju. Neilgi pēc tā izveidojās divi bieži sastopami pārpratumi saistībā RDF. Pirmkārt, MCF ietekmes rezultātā un RDF "Resursu Aprakstīšanas" idejas rezultātā, tika uzskatīts, ka RDF ir tieši specializēts metadatu reprezentācijai; otrkārt, tas, ka RDF ir XML formāts, nevis datu modelis un tikai RDF/XML serializācija ir XML bāzēta. 2014. gadā tika izveidoti šādi seši "RDF 1.1" dokumenti: "RDF 1.1 Primer", "RDF 1.1 Concepts and Abstract Syntax", "RDF 1.1 XML Syntax", "RDF 1.1 Semantics", "RDF Schema 1.1" and "RDF 1.1 Test Cases". [4]

2. SAISTĪTIE DATI

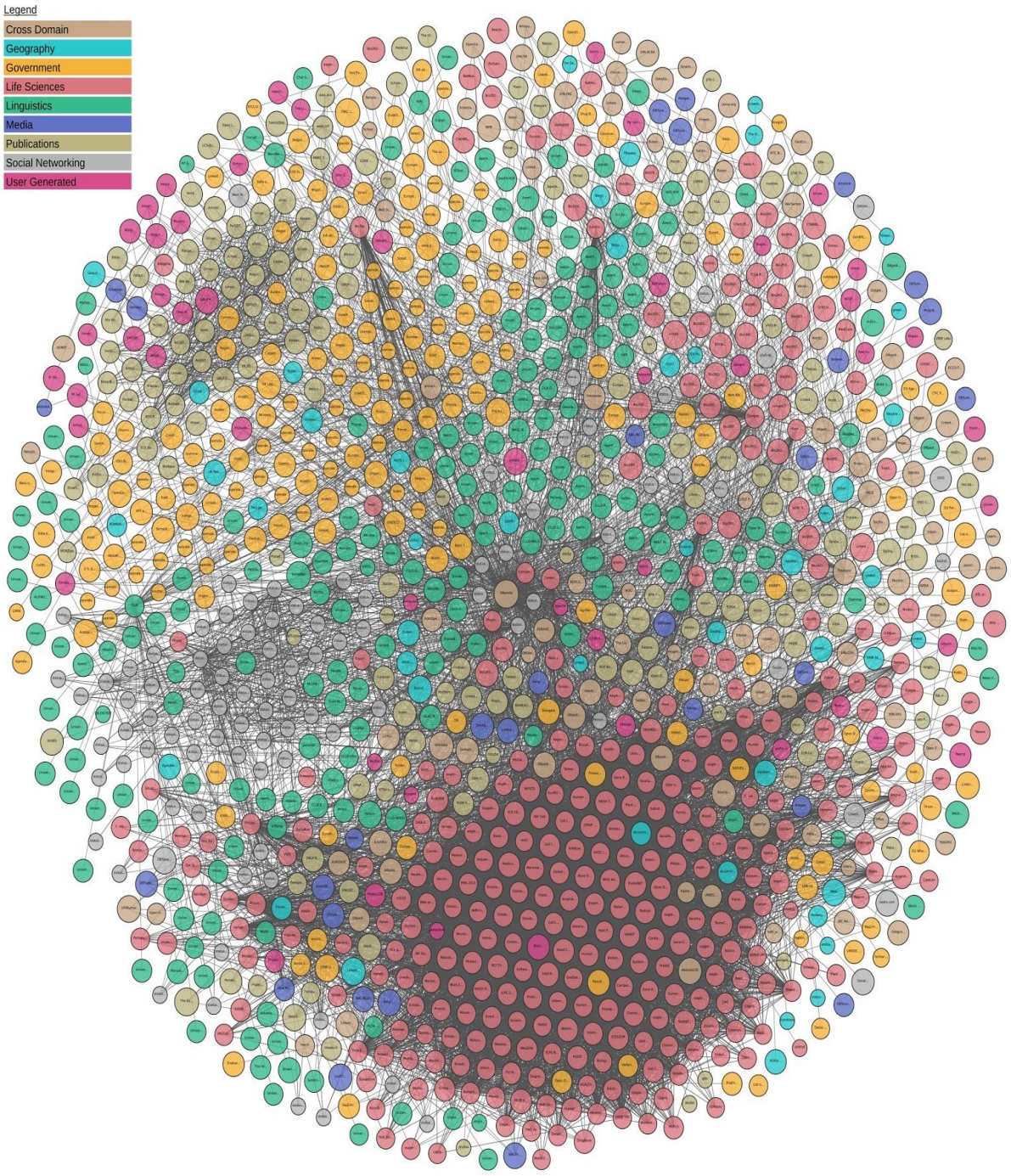
Jēdziens atvērtie "linked data" tiek lietots kopš 2007.gada februāra, kad tika izveidots "Linking Open Data" adresātu saraksts. Sākotnēji šo adresātu sarakstu uzturēja SMILE projekta ietvaros Masačūsetas Tehnoloģiju institūtā.

Četri pamatprincipi no Tima Bernesa Lī "Linked Data" 2006.gadā izdotās publikācijas:

1. Lai nosauktu (identificētu) konkrētus vienumus tiek lietoti URI.
2. Lai šos vienumus atrastu (interpretētu, "dereferencētu"), tiek izmantoti HTTP URI.
3. Jānodrošina noderīga informācija par to, ka konkrētais vienums tiek identificēts, kad tas tiek uzmeklēts, izmantojot tādus atvērtos standartus kā RDF, SPARQL utt.
4. Jāatsaucas uz citiem vienumiem, izmantojot to HTTP URI bāzētos nosaukumus, kad dati tiek publicēti tīmeklī. [1]

2.1. Saistīto datu būtība

Tīmeklis sniedz iespēju saistīt dažādus dokumentus. Līdzīgi tas ļauj sasaistīt radniecīgus datus. Jēdziens "Saistītie Dati" apzīmē vairāku "labās prakses" metožu kopumu strukturētu datu savstarpējai sasaistīšanai un publicēšanai tīmeklī. Nozīmīgākās tehnoloģijas, kas atbalsta Saistīto Datu izmantošanu ir URI (vispārīga metode, lai identificētu konkrētus vienumus un dažādus konceptus), HTTP (vienkāršs, bet universāls mehānisms resursu vai to aprakstu iegūšanai) un RDF (vispārīgs grafus izmantojošs datu modelis, ar kuru strukturēti sasaistīt datus, kas apraksta dažādus vienumus). [2] Zemāk esošajā 10 lappusē 2.1. attēlā ir redzams saistīto datu mākoņa grafisks attēlojums.



2.1. att. Saistīto datu mākoņa attēlojums [17]

2.2. Saistītie dati un to izmantošana

Semantiskais tīkls ir datu tīkls ar datumiem un virsrakstiem, daļskaitļiem, ķīmiskajām īpašībām, vai jebkuriem citiem iedomājamiem datiem. Pieejamās semantisko tīklu tehnoloģijas (RDF, OWL, SKOS, SPARQL, utt.) rada vidi, kurā lietotne var pārbaudīt šos datus, izdarīt secinājumus, izmantojot vārdnīcas, utt.

Lai padarītu datu tīklu par realitāti, nepieciešams formatēt milzīgo internetā pieejamo datu apjomu pēc vienota standarta, lai semantiskie tīkli spētu tam piekļūt un to apstrādāt. Bez tam, semantiskajam tīklam nepieciešama ne tikai piekļuve datiem, bet arī attiecībām starp šiem datiem, lai izveidotu datu tīklu (pretstatā vienkāršam datu apkopojumam). Šī savstarpēji saistīto datu kolekcija arī ir minētie sasaistītie dati.

Lai radītu sasaistītos datus, jābūt pieejamām tehnoloģijām, kas izmanto vienotu formātu (RDF), lai varētu konvertēt vai procesa gaitā piekļūt esošām datu bāzēm (XML, HTML, utt.) Ir arī svarīgi, lai būtu iespējams uzstādīt vaicājuma piekļuves punktus, lai ērtāk piekļūtu šiem datiem. W3C nodrošina tehnoloģiju paleti (RDF, GRDDL, POWDER, RDFa, the upcoming R2RML, RIF, SPARQL), kas ļauj piekļūt datiem.

Sasaistītie dati ir semantiskā tīkla pamatā - liela mēroga datu integrācija un spriestspēja internetā. Praktiski visas aplikācijas, kas uzskaitītas collection of Semantic Web Case Studies and Use Cases, būtībā ir balstītas uz sasaistīto datu pieejamību un integrāciju dažādās komplicētības pakāpēs. [3]

2.3. Dažas populārākās saistīto datu kopas

1. DBpedia - datukopa, kas satur datus, kas ir iegūti no Vikipēdijas; tā satur aptuveni 3,4 miljonus konceptu, ko apraksta miljards trijnieku, tajā skaitā ierakstus 11 dažādās valodās. DBpedia ir nozīmīga, jo iekļauj ne tikai Vikipēdijas datus, bet arī saites uz citām datu bāzēm internetā, piem., Geonames. Pateicoties šīm papildus saitēm, aplikācijas var izmantot papildus pieejamo (un, iespējams, precīzāko) informāciju no citām datu bāzēm, izstrādājot jaunas aplikācijas. Integrējot faktus no dažādām datu bāzēm, aplikācijas var nodrošināt daudz labāku lietošanas kvalitāti.
   saite: https://wiki.dbpedia.org/
2. GeoNames – satur RDF aprakstus par vairāk kā 25,000,000 ģeogrāfiskiem nosaukumiem, kas saistīti ar vairāk kā 11,800,000 unikāliem objektiem.
   saite: https://www.geonames.org/
3. Wikidata – kopdarbīgi radīta saistīta datukopa, kas tiek izmantota kā Wikimedia Fundation paralēlo projektu strukturēto datu centrālā noliktava
   saite: https://www.wikidata.org/wiki/Wikidata:Main_Page
4. Global Research Identifier Database (GRID) –internacionāla 89 506 akadēmiskos pētījumos iesaistītu institūciju datubāze ar 14 401 savstarpējām relācijām, satur divu

veidu relācijas: vecāka - bērna relācijas, kas definē subordinātu asociāciju un saistīta relācija, kas apraksta citas asociācijas

saite: https://www.grid.ac/

3. SPARQL

SPARQL (Izrunā "spārkl", tas ir rekursīvs akronīms SPARQL protokola un RDF vaicājumu valodas apzīmēšanai, angļu SPARQL Protocol and RDF Query Language) ir RDF vaicājumu valoda - tas ir, semantisko vaicājumu valoda datu bāzēm, kas ļauj piekļūt un apstrādāt datus RDF formātā. To standartizēja RDF Data Access Working Group (DAWG) no World Wide Web Consortium, un to uzskata par vienu no galvenajām semantiskā tīkla tehnoloģijām. 2008.gada 15. janvārī, SPARQL 1.0 tika pasludināta par oficiālo W3C ieteikumu, un SPARQL 1.1 - 2013. gada martā. SPARQL ļauj vaicājumam sastāvēt no trīskāršiem paterniem, konjunkcijām, disjunkcijām un opcionāliem paterniem. [7], [8]

3.1. Priekšrocības

SPARQL ļauj lietotājiem rakstīt vaicājumus pēc t.s. "atslēgvērtības" datiem, vai precīzāk, datiem, kas atbilst RDF specifikācijām W3C. Tādā veidā vesela datu bāze kļūs par "subjekts - predikāts - objekts" trijnieku. Tas ir līdzīgi dažās NoSQL datu bāzēs lietotajam apzīmējumam "dokuments - atslēga - vērtība", piem., MongoDB.

SQL relāciju datu bāzu terminos RDF datus var uzskatīt par tabulu ar trim kolonnām - subjektu, predikātu un objektu kolonnu. Subjekts RDF ir analogs vienumam SQL datu bāzē, kurā datu elementi (vai lauki) katram darba objektam ir izvietoti vairākās kolonnās, nereti vairākās dažādās tabulās, un atpazīti pēc unikālas atslēgas. RDF formātā, šos lauciņus aizstāj atsevišķas predikātu/objektu rindas ar kopēju subjektu un nereti arī kopēju unikālo atslēgu, kur predikāts ir analogs kolonnas nosaukumam un objekts pašiem datiem. Atšķirībā no relacionālajām datu bāzēm, objektu kolonna ir heterogēna - šūnās esošo datu tips parasti tiek norādīts ar predikāta vērtību. Atšķirībā no SQL, RDF var būt arī vairāki ieraksti katram predikātam - piemēram, vairāki ieraksti "Bērns" katram predikātam "Persona" var tikt uzrādīti kā objektu kopums "Bērni".

Šādi SPARQL nodrošina pilnu analītisko vaicājumu operāciju komplektu, piem., JOIN, SORT, AGGREGATE datiem, kuru shēma ir neatdalāma daļa no datiem bez nepieciešamības pēc atsevišķas shēmas definīcijas. Tomēr shēmas informācija (ontoloģija) nereti tiek nodrošināta ārēji, ļaujot nepārprotami apvienot dažādas datu bāzes. Piedevām SPARQL nodrošina specifisku grafisko traversālo sintaksi datiem, kurus var attēlot kā grafikus.

Zemāk esošajā piemērā redzams vienkāršs vaicājums, izmantojot foaf (angļu friend of a friend) ontoloģijas definīciju. Izmantojot šo konkrēto vaicājumu, var iegūt vārdus un epastus visām datu kopā esošajām personām:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
        ?email
WHERE
{
        ?person  a        foaf:Person .
       ?person  foaf:name  ?name .
       ?person  foaf:mbox  ?email .
}
```

3.1. att. SPARQL vaicājuma piemērs

Šis vaicājums apvieno visus trijniekus ar kopīgu subjektu, kuriem tipa predikāts, "a", ir persona (foaf:Person) un katrai personai ir viens vai vairāki vārdi (foaf:name) un epasta adreses (foaf:mbox). Subjekts šajā vaicājumā tiek apzīmēts ar mainīgo "?person", lai nodrošinātu vaicājuma lasāmību. Tā kā jebkura trijnieka pirmais elements ir subjekts, tad tā apzīmēšanai būtu iespējams izmantot jebkuru citu mainīgā vārdu, piemēram, "?subj" vai "?x". Neatkarīgi no tā kāds mainīgā vārds tiek izmantots, šajā piemērā tam katrā rindiņā ir jāizmanto viens un tas pats mainīgā vārds, lai izmantojot vaicājumu mehānismu, varētu iegūt apvienojumu no trijniekiem, kuriem ir kopīgs subjekts.

Apvienojuma rezultāts ir kopa ar rindiņām - ?person, ?name, ?email. Šis vaicājums atgriež ?name un ?email tāpēc, ka ?person bieži ir sarežģīts URI, nevis cilvēkiem draudzīgi izveidots string tipa mainīgais. Jāņem vērā, ka jebkuram mainīgajam ?person, var atbilst vairākas epasta adreses, tātad atgrieztajā kopā ?name rindiņa var parādīties vairākas reizes, lai atbilstu katrai epasta adresei. Šis vaicājums var tikt vienlaicīgi izpildīts vairākos SPARQL piekļuves punktos (servisi izmantojot SPARQL vaicājumus atgriež rezultātus), apstrādāts un, apkopojot starprezultātus, iegūts gala rezultāts. Šo procedūru sauc par apvienoto vaicājumu (angļu federated query). Neatkarīgi no tā vai tiek izmantots apvienotais vaicājums, vai lokālais, vaicājumam pievienojot papildus trijnieku definīcijas, ir iespējams apvienot dažādus

subjektu tipus, kā, piemēram, izveidot vienkāršu vaicājumu, kas atgrieztu sarakstu ar cilvēku vārdiem un epastiem, kuri izmanto automašīnas, kurām ir zems degvielas patēriņš. [7], [8]

3.2. Vaicājumu formāti un sintakse

Gadījumos, kad nepieciešams iegūt datus no datu bāzēm, SPARQL valodā dažādu mērķu sasniegšanai tiek izmantoti četri dažādi vaicājumu varianti.

- "SELECT" vaicājumi - tiek izmantoti, lai no piekļuves punktiem iegūtu neapstrādātas vērtības, kas tiek atgrieztas tabulas veidā
- "CONSTRUCT" vaicājumi - tiek izmantoti, lai iegūtu informāciju no SPARQL piekļuves punktiem un pārveidotu iegūtos rezultātus tālāk apstrādājamā RDF
- "ASK" vaicājumi - tiek izmantoti, lai no SPARQL piekļuves punkta iegūtu vienkāršu true/false tipa rezultātu
- "DESCRIBE" vaicājumi - tiek izmantoti, lai iegūtu RDF grafu no SPARQL piekļuves punkta, tā saturs ir atkarīgs no katra komkrētā piekļuves punkta

Katrs no šiem vaicājumiem izmanto "WHERE" bloku, lai ierobežotu vaicājuma atgrieztos rezultātus, lai gan "DESCRIBE" vaicājuma gadījumā "WHERE" bloks nav obligāts.
SPARQL 1.1 specifikācija apraksta valodu, kas saturs vairākas jaunas vācājumu formas, ar kuru palīdzību ir iespējams ne tika iegūt, bet arī saglabāt informāciju datu bāzēs

```
PREFIX ex: <http://example.com/exampleOntology#>
SELECT ?capital
          ?country
WHERE
{
        ?x  ex:cityname      ?capital   ;
             ex:isCapitalOf    ?y        .
         ?y  ex:countryname    ?country   ;
             ex:isInContinent  ex:Africa  .
}
```

3.2 att. SPARQL vaicājuma piemērs, kas atlasa visas Āfrikas valstu galvaspilsētas.

Mainīgos apzīmēi ar a ? or $ prefiksu. Mainīgie ?capital un ?country apzīmē vērtības, kas tiks atgrieztas. Ja trijnieku noslēdz semikols, tad šī trijnieka subjekts tiek izmantots arī tam sekojošajam trijniekam. Tas nozīmē to, ka šajā piemērā ex:isCapitalOf ?y apzīmē to pašu ko ?x ex:isCapitalOf ?y.

SPARQL vaicājumu procesors meklēs kopas, kas atbilst šiem četriem trijniekiem, apzīmējot atgrieztās vērtības ar katru trijniekos esošo konkrēto atbilstošo mainīgo. Svarīga iezīme šeit ir "orientācija pēc īpašībām" , angļu property orientation, (klašu atbilstības var tikt veidotas izmantojot vienīgi klašu atribūtus vai īpašības, ņemot vērā "Duck typing"). Lai nodrošinātu vaicājumu lakoniskumu SPARQL tiek izmantota prefiksu definēšana un pamata URI, līdzīgi kā tas tiek darīts Turtle sintaksē. Piemēra vaicājumā prefix "ex" aizstāj saiti "http://example.com/exampleOntology#". [7], [8]

4. VIZIQUER PIELIETOJUMS:

ViziQuer ir atvertā pirmkoda rīks, kas izmanto tīmekļa resursus vizuālu vaicājumu izveidei un izpildīšanai diagrammu veidā pār RDF/SPARQL. ViziQuer rīks ļauj veikt datu instances līmeņa un statistiskus vaicājumus, nodrošinot vizuālu attēlojumu lielākajai daļai SPARQL 1.1 SELECT vaicājumiem, tajā skaitā agregācijām un apakšvaicājumiem.

4.1. ViziQuer darbības principi

Viziquer rīks piedāvā jaunu pieeju SPARQL piekļuves punktu izpētei un vaicājumu izpildei tajos. Šo rīku ir samērā vienkārši lietot, jo lietotājam ir jāievada tikai interesējošā SPARQL piekļuves punkta adrese, taču pilnvērtīgai ViziQuer lietošanai ir nepieciešama arī konkrētā piekļuves punkta datu shēma, ko iegūst izmantojot SPARQL piekļuves punktu shēmu iegūšanas rīku. [15] Ielādējot iegūto shēmu ViziQuer tiešsasites rīkā, ir iespējams grafiski vizualizēt konkrētā piekļuves punkta datu shēmu. Lietotājam tiek sniegta iespēja pārlūkot šo datu shēmu un izmantot to, lai veidotu šai shēmai atbilstošus SPARQL vaicājumus. Vaicājumu vide ir organizēta lietotājiem ērtā veidā - sadalot to projektos. Katram projektam ir atseviška datu shēma nosaukumu uzglabāšanai un SPARQL vaicājumu ģenerēšanai, kā arī savs SPARQL piekļuves punkts, lai nodoršinātu savienojumu vaicājumu izpildei. Katrs projekts sastāv no diagrammām. Katrā diagrammā var būt viens vai vairāki vaicājumi.

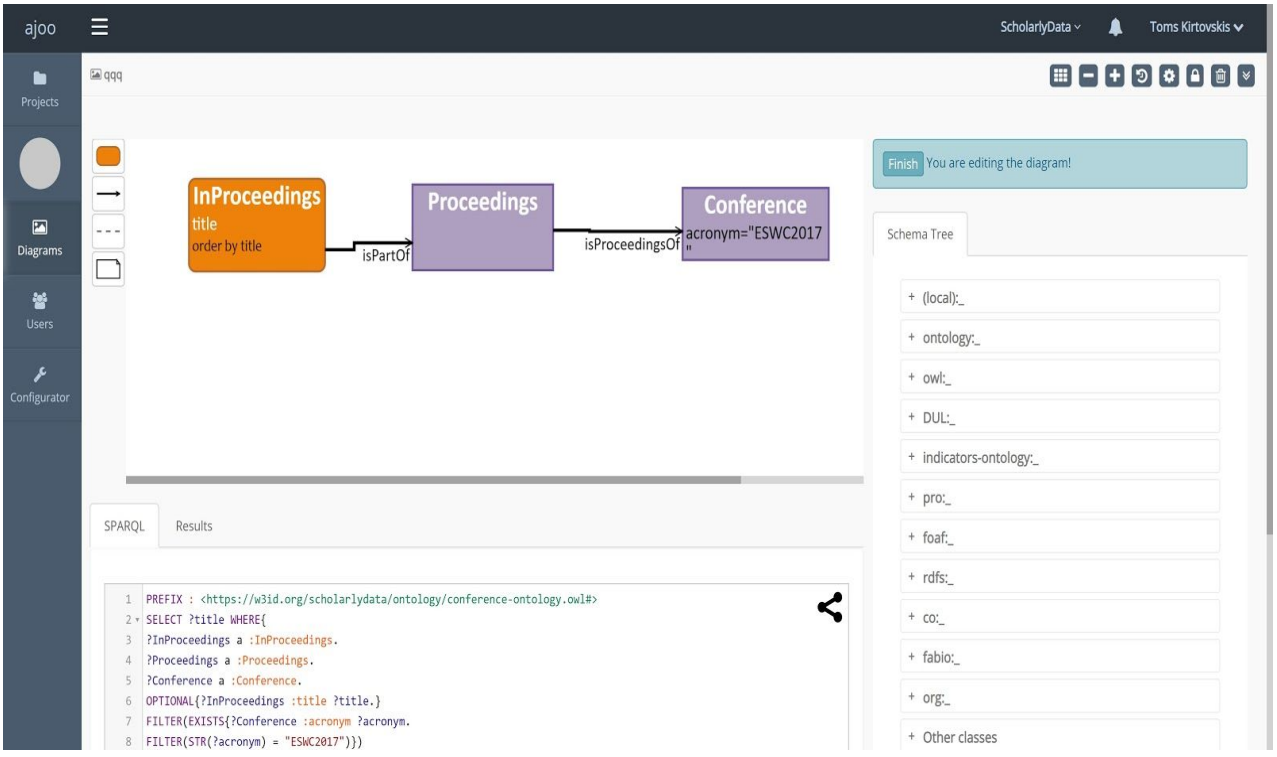
ViziQuer rīks nodrošina iespēju pieslēgties SPARQL piekļuves punktam, kas satur tipveida datus. Tipveida dati ir dati, kuros objekti pieder klasēm, kas ir definētas izmantojot rdf:type relācijas. Šo rīku var izmantot, lai izpētītu un izprastu, kā arī, lai veiktu vaicājumus SPARQL semantisko datu bāžu piekļuves punktos, kas šķiet interesanti un/vai noderīgi. ViziQuer sastāv no divām daļām, un to ir ieteicams lietot kopā ar SPARQL piekļuves punktu shēmu iegūšanas rīku. Pirmkārt, tas sastāv no rīka, kas nodrošina iespēju lietotājam aplūkot iegūtās datu shēmas klašu struktūru, attēlotu klašu koka veidā. Otrkārt, lietotājs var izmantot ViziQuer rīku, lai izveidotu SPARQL vaicājumu atbilstoši kādai konkrētai datu shēmai tā, lai iegūto vaicājumu būtu iespējams izpildīt atbilstošajā SPARQL piekļuves punktā. Izmantojot SPARQL piekļuves punktu shēmu iegūšanas rīku, ir iespējams pieslēgties un iegūt konkrētā SPARQL piekļuves punktā esošo datu shēmu. Pieslēgšanās process norisinās, ievadot konkrētā lietotājam interesējošā SPARQL piekļuves punkta adresi. Izmantojot iepriekš

definētu SPARQL vaicājumu kopumu, SPARQL piekļuves punktu shēmu iegūšanas rīks var iegūt konkrētā SPARQL piekļuves punkta elementāru datu shēmu. Ekstrakcijas process var aizņemt salīdzinoši ilgu laiku, jo datu shēmas iegūšana ir atkarīga no katras konkrētās ontoloģijas izmēra un katra konkrētā SPARQL piekļuves punkta darbības ātruma. ViziQuer ir iespējams lietot tikai ar tipveida datiem. Šāds nosacījums ir svarīgs, lai būtu iespējams iegūt vismaz daļu no nepieciešamās datu shēmas, kā piemēram klases, to atribūtus un relācijas.

Nozīmīga ViziQuer rīka piedāvāta iespēja ir daļēji automātiska konkrētai datu shemai (ontoloģijai) atbilstošu SPARQL vaicājumu izveide. ViziQuer rīkam ir pašam sava vaicājumu valoda, kas nodrošina pamatfunkcijas datu kopu un apakškopu pārlūkošanai. Tā kā vaicājumi ir veidoti izmantojot konkrētu ontoloģiju, tad vaicājumu izpildīšana ir līdzīga apakškopu veidošanai no lietotāju interesējošajiem datiem, ka atbilst konkrētajai ontoloģijai. Vaicājumu izveide ir iespējama, izmantojot divas paradigmas. Pirmā - lietotājs var veidot vaicājumu izvēloties kādu konkrētu klasi un veicot tālāku pārlūkošanu, līdzīgi kā izmantojot vairāku vienlaicīgu filtru pārlūkošanu, kur lietotājs izvēlas ceļu tālāk no klasēm, kas jau ir izvēlētas pārlūkošanai. Otrā - lietotājs var izmantot savienošanu izmantojošu vaicājumu izveidi. Izvēloties šādu pieeju, lietotājs izvēlas divas tam interesējošas klases un pievieno tās diagrammas veida vaicajumam un savieno tās ar līniju, kas apzīmē to, ka šīs abas klases ir jāapvieno vaicājumam. ViziQuer izmanto samērā sarežģītu algoritmu, lai ieteiktu piemērotākos veidus, kā šīs klases varētu tikt saistītas un lietotājam vienkārši ir jāizvēlas veids, kas vislabāk atbilst izvēlētajam mērķim. Tātad, ja lietotājs vēlas savienot klases, kas ir salīdzinoši atstatu viena no otras, pieņemamu savienošanas veidu izvēle ir noderīgāka kā minēšana, veicot pašrocīgu pārlūkošanu, kas, darbojoties ar vienlaicīgu vairāku filtru pārlūkošanas paradigmu, var būt pat ļoti sarežģīta, ja lietotājs nav labi pazīstams ar konkrētās ontoloģijas struktūru. Papildus tam apakškopas ir iespējams ierobežot ar dažiem vienkāršākiem klašu atribūtu filtriem. [6], [10]

4.1.1. ViziQuer darbības piemēri

ViziQuer diagrammu redaktorā izveidoti vizuālu vaicājumi piemēri redzami 4.1., 4.3. un 4.5. attēlos. Šie vaicājumi izveidoti, izmantojot no Scholarlydata piekļuves punkta iegūtu datu shēmu. Datu shēmas fails ir pieejams ViziQuer rīka Git koda glabātuvē ar nosaukumu "scholarlyData.json". Lai nodrošinātu Scholarlydata piekļuves punkta datu shēmas jaunākās versijas izmantošanu, šo shēmu ir iespējams iegūt, izmantojot SPARQL datu shēmu iegūšanas rīku. 4.1. attēlā redzams vizuāla vaicājuma piemērs, kurā izmantots filtrs. 4.2. attēlā redzams šim vizuālajam vaicājumam atbilstošs SPARQL vaicājums, kas ģenerēts izmantojot ViziQuer rīku.

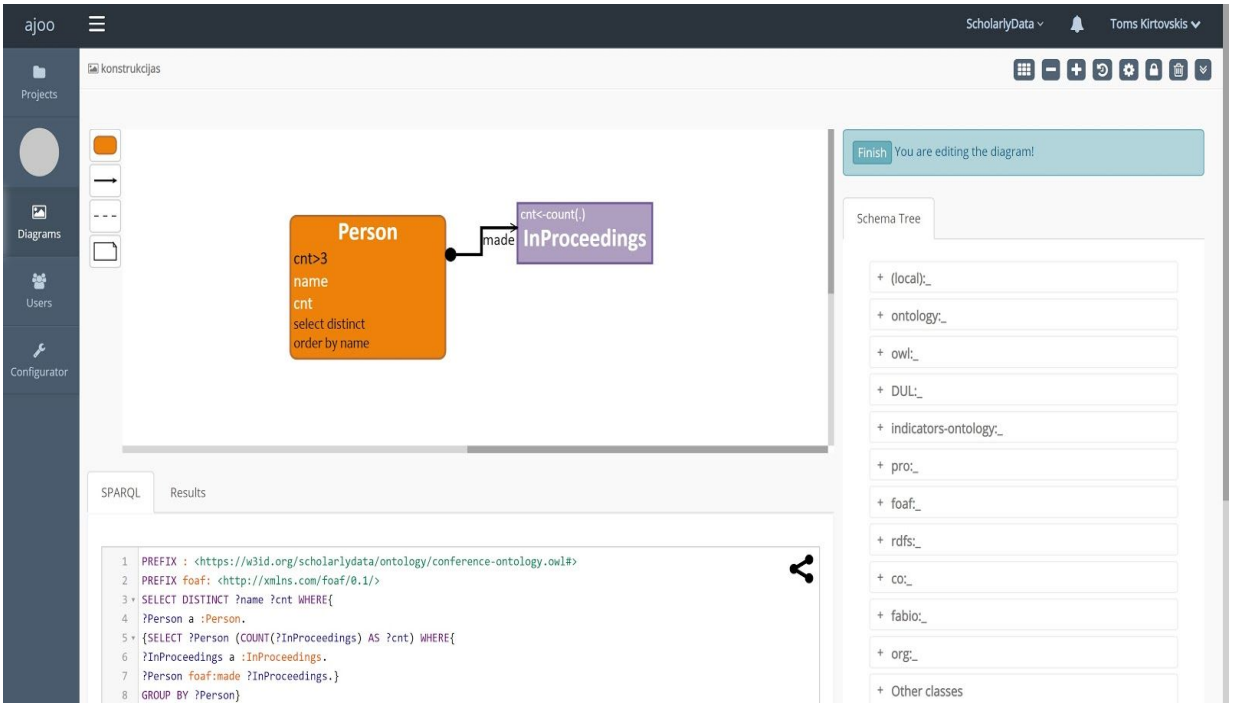


4.1. att. Vizuāls vaicājums, ar kura palīdzību var iegūt visas publikācijas no konferences ar akronīmu "ESWC2017"

```
PREFIX : <https://w3id.org/scholarlydata/ontology/conference-ontology.owl#>
SELECT ?title WHERE{
?InProceedings a :InProceedings.
?Proceedings a :Proceedings.
?Conference a :Conference.
OPTIONAL{?InProceedings :title ?title.}
FILTER(EXISTS{?Conference :acronym ?acronym.
FILTER(STR(?acronym) = "ESWC2017")})
?InProceedings :isPartOf ?Proceedings.
?Proceedings :isProceedingsOf ?Conference.}
ORDER BY ?title
```

4.2. att. SPARQL vaicājums, ar kura palīdzību var iegūt visas publikācijas no konferences ar akronīmu “ESWC2017”.

4.3. attēlā redzams vizuāla vaicājuma piemērs, kurā izmantots apakšvaicājums un filtrs. 4.4. attēlā redzams šim vizuālajam vaicājumam atbilstošs SPARQL vaicājums, kas ģenerēts izmantojot ViziQuer rīku.



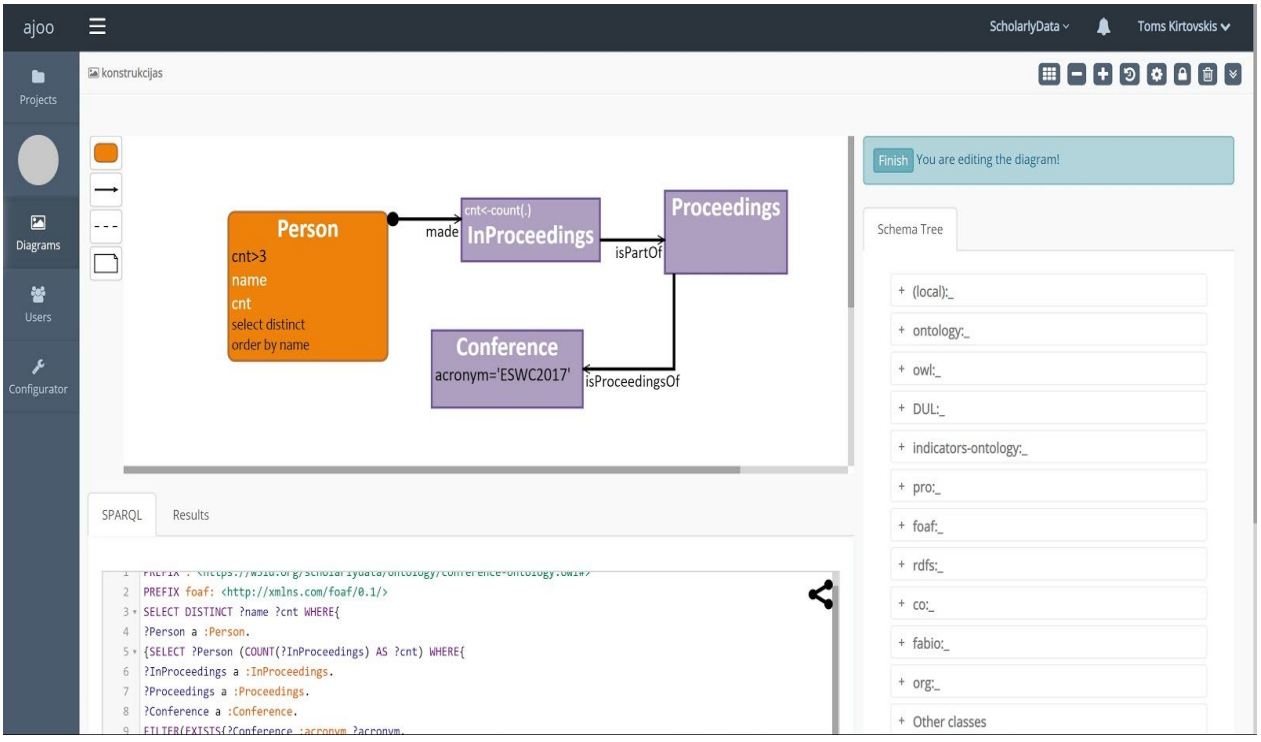
4.3. att. Vizuāls vaicājums, ar kura palīdzību var iegūt visu autoru vārdus, kuriem ir vismaz 3 publikācijas, skārtotus alfabētiskā kārtībā.

```
PREFIX : <https://w3id.org/scholarlydata/ontology/conference-ontology.owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?name ?cnt WHERE{
?Person a :Person.
{SELECT ?Person (COUNT(?InProceedings) AS ?cnt) WHERE{
?InProceedings a :InProceedings.
?Person foaf:made ?InProceedings.}
GROUP BY ?Person}
OPTIONAL{?Person :name ?name.}
FILTER(?cnt > 3)}
ORDER BY ?name
```

4.4. att. SPARQL vaicājums, ar kura palīdzību var iegūt visu autoru vārdus, kuriem ir vismaz 3 publikācijas, sakārtotus alfabētiskā kārtībā.

4.5. attēlā redzams vizuāla vaicājuma piemērs, kurā izmantots apakšvaicājums un divi filtri.

4.6. attēlā redzams šim vizuālajam vaicājumam atbilstošs SPARQL vaicājums, kas ģenerēts izmantojot ViziQuer rīku.



4.5. att. Vizuāls vaicājums, ar kura palīdzību var iegūt visu autoru vārdus, kuriem ir vismaz 3 publikācijas, kas ir publicēti konferencē ar akronīmu “ESWC2017”, skārtotus alfabētiskā kārtībā

```
PREFIX : <https://w3id.org/scholarlydata/ontology/conference-ontology.owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?name ?cnt WHERE{
?Person a :Person.
{SELECT ?Person (COUNT(?InProceedings) AS ?cnt) WHERE{
?InProceedings a :InProceedings.
?Proceedings a :Proceedings.
?Conference a :Conference.
FILTER(EXISTS{?Conference :acronym ?acronym.
FILTER(STR(?acronym) = 'ESWC2017')})
?Person foaf:made ?InProceedings.
?InProceedings :isPartOf ?Proceedings.
?Proceedings :isProceedingsOf ?Conference.}
GROUP BY ?Person}
OPTIONAL{?Person :name ?name.}
FILTER(?cnt > 3)}
ORDER BY ?name
```

4.6. att. SPARQL vaicājums, ar kura palīdzību var iegūt visu autoru vārdus, kuriem ir vismaz 3 publikācijas, kas ir publicēti konferencē ar akronīmu “ESWC2017”, sakārtotus alfabētiskā kārtībā.

4.2. ViziQuer un SPARQL piekļuves punktu savstarpējā saziņa

Lai no kāda konkrēta SPARQL piekļuves punkta iegūtu informāciju, uz to ir jānosūta SPARQL vaicājums. Vaicājuma nosūtīšanai tiek izmantots HTTP pieprasījums, kuru veic izmantojot GET vai POST metodes.

4.2.1. Pieprasījumu tipi

Pieprasījuma operācija tiek lietota, lai nosūtītu SPARQL vaicājumu servisam un saņemtu vaicājuma rezultātus. Pieprasījuma operāciju OBLIGĀTI jāveic ar vai nu HTTP GET, vai HTTP POST metodi. Klienta pieprasījumiem šai operācijai jāiekļauj tieši vienu SPARQL vaicājuma teksta virkni (parametra nosaukums: vaicājums), un tie var iekļaut nulle vai vairāk noklusētos grafus URI (parametra nosaukums: noklusētais - grafs - uri (angļu

default - graph - uri) un nosauktos grafus URI (parametra nosaukums: nosauktais - grafs - uri (angļu named - graph - uri). Atbilde uz vaicājuma pieprasījumu ir vai nu SPARQL XML Results formātā, SPARQL CSV/TSV Results formātā, vai arī RDF serializācija, atkarībā no vaicājuma formas [7] un informācijas par satura formātu [10].

4.1. tabula

HTTP pieprasījumu apraksts

	HTTP metode	Vaicājuma teksta virknes parametri	Pieprasījuma satura tips	Pieprasījuma teksta saturs
Pieprasījums izmantojot GET	GET	vaicājums (tieši 1) noklusētais-grafs-uri (0 vai vairāk) nosauktais-grafs-uri (0 or more)	Nav	Nav
Pieprasījums izmantojot URL-iekodētu POST	POST	Nav	application/x-www-form-urlencoded	URL-iekodēti, ar "&" atdalīts pieprasījuma parametri pieprasījums (tieši 1) noklusētais-grafs-uri (0 vai vairāk) nosauktais-grafs-uri(0 vai vairāk)
Tiešs pieprasījums izmantojot POST	POST	noklusētais-grafs-uri (0 vai vairāk) nosauktais-grafs-uri (0 vai vairāk)	application/sparql-query	Neiekodēta SPARQL vaicājumu teksta virkne

4.2.1. Vaicājums izmantojot GET

Protokolu izmantojošā klienta puse var nosūtīt protokola pieprasījumus ar HTTP GET metodi. Izmantojot GET metodi, klienta pusē tiek izmantota procent-iekodēšana visiem parametriem un jāiekļauj tie kā vaicājumu parametru teksta virknes ar augstāk norādītajiem nosaukumiem.

HTTP vaicājumu teksta virkņu parametriem jābūt atdalītiem ar "&" zīmi. Klienta pusē var iekļaut vaicājumu teksta virkņu parametrus jebkādā secībā. HTTP pieprasījums nedrīkst saturēt ziņojuma tekstu. [13]

4.2.2. Pieprasījums izmantojot POST ar URL-iekodētiem parametriem

Protokola klienta puse var nosūtīt protokola pieprasījumus ar HTTP POST metodi ar URL-iekodētiem parametriem. Izmantojot šo metodi, klientiem procentuāli jāiekodē [11] URL visi parametri, un tie jāiekļauj kā parametri pieprasījuma saturā ar *application/x-www-form-urlencoded* mediju tipu ar augstāk norādīto nosaukumu. Parametriem jābūt atdalītiem ar "&" zīmi. Klienta puse var iekļaut parametrus jebkādā secībā. Satura tipa galvenei HTTP pieprasījumā jābūt iestatītai uz *application/x-www-form-urlencoded*.

Šis ir noklusētais satura tips. Ziņojumiem, kas nosūtīti ar šādu satura tipu, jābūt iekodētiem sekojoši:

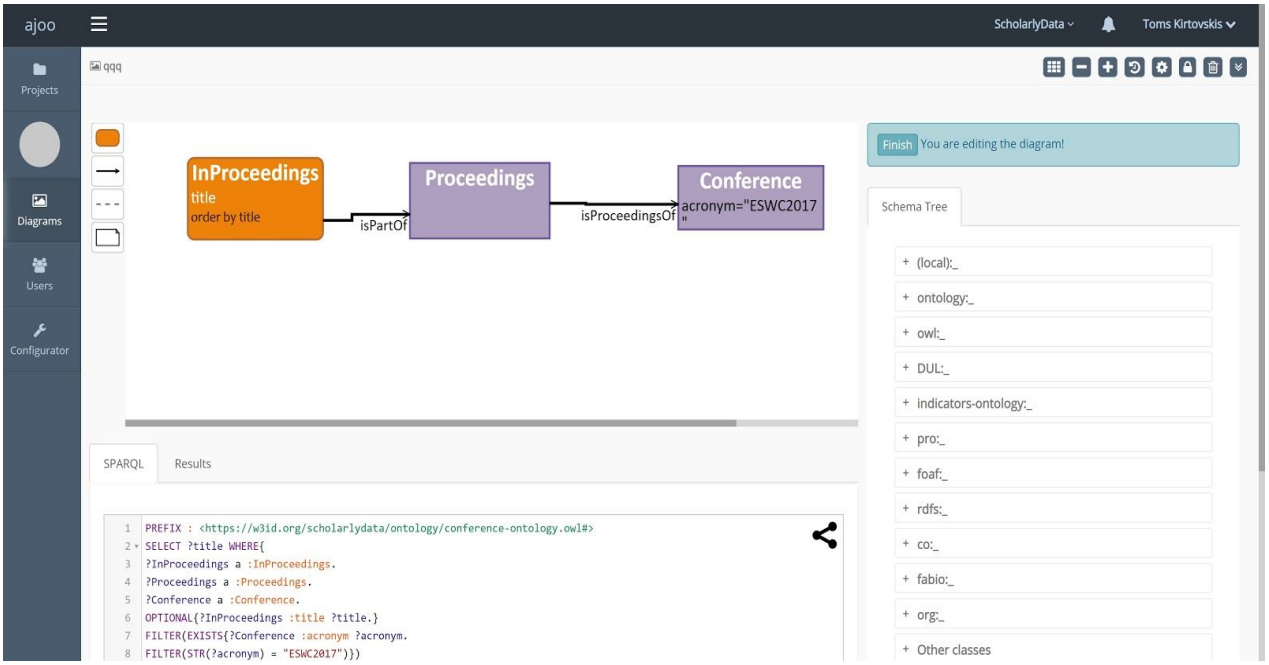
- Kontroles apzīmējumi un vērtības ir izcelti (angļu "escaped"). Atstarpju zīmes aizstāj ar "+", rezervētās zīmes izceļ tā, kā aprakstīts *"Uniform Resource Locators", T. Berners-Lee, L. Masinter, and M. McCahill, December 1994, section 2.2:* Ne-alfanumerālās zīmes tiek aizstātas ar "%HH", procentu zīmi un diviem heksadecimālajiem skaitļiem kas apzīmē attiecīgās zīmes atbilstošo ASCII kodu. Rindu pārtraukumi tiek apzīmēti kā "CR LF" pāri (piem., `%0D%0A').
- Kontroles apzīmējumi/vērtības tiek uzskaitītas tādā secībā, kādās tie parādās dokumentā. Nosaukumu atdala no vērtības ar "=" zīmi, un nosaukumu/vērtību pāri tiek atdalīti viens no otra ar "&". [13]

4.2.3. Tiešs pieprasījums izmantojot POST

Protokola klienta puse var nosūtīt protokola pieprasījumus ar HTTP POST metodi, tiešā veidā iekļaujot vaicājumu neiekodētu kā HTTP pieprasījuma ziņojuma saturu. Izmantojot šādu pieeju, klientiem jāiekļauj neiekodēta SPARQL vaicājuma teksta virkne un nekas cits, kā pieprasījuma ziņojuma saturs. Klientam jāiestatī HTTP pieprasījuma satura tipa galveni kā *application/sparql-query*. Klienti var iekļaut neobligātos noklusētais-grafs-uri un nosauktais-grafs-uri parametrus kā HTTP vaicājuma teksta virknes parametrus pieprasījuma URI. Jāņem vērā, ka šeit derīgs ir tikai UTF-8 zīmju komplekts. [13]

4.2.4. Tiešā POST pieprasījuma un POST pieprasījuma ar URL-iekodētiem parametriem salīdzinājums ViziQuer rīka gadījumā

Kā piemēra vaicājums izmantots vaicājums pār Scholarlydata piekļuves punktu, kas attēlo visas publikācijas no konferences ar akronīmu “ESWC2017”.



4.7. att. Vizuāls vaicājums, ar kura palīdzību var iegūt visas publikācijas no konferences ar akronīmu “ESWC2017”.

```
PREFIX : <https://w3id.org/scholarlydata/ontology/conference-ontology.owl#>
SELECT ?title WHERE{
?InProceedings a :InProceedings.
?Proceedings a :Proceedings.
?Conference a :Conference.
OPTIONAL{?InProceedings :title ?title.}
FILTER(EXISTS{?Conference :acronym ?acronym.
FILTER(STR(?acronym) = "ESWC2017")})
?InProceedings :isPartOf ?Proceedings.
?Proceedings :isProceedingsOf ?Conference.}
ORDER BY ?title
```

4.8. att. SPARQL vaicājums, ar kura palīdzību var iegūt visas publikācijas no konferences ar akronīmu “ESWC2017”.

ViziQuer rīks šo vaicājumu pārveido HTTP POST pieprasījumā. Izmantojot tiešo HTTP POST pieprasījumu tiek iegūts:

```
HTTP.call("POST", options.endPoint, options.params, function(err, resp) {
    //kļūdu un/vai iegūtā rezultāta apstrāde
}
```

4.9.att. ViziQuer HTTP POST tiešais pieprasījums

Kā parametri šai funkcijai tiek padoti gan piekļuves punkts, gan SPARQL vaicājums ar visām nepieciešamajām detaļām. Parametra "options.endPoint" saturs, kas uzģenerēts izmantojot 4.1.att. un 4.2.att redzamo vaicājumu redzams 4.3. att.

```
http://www.scholarlydata.org/sparql/
```

4.10. .att. Parametra "options.endPoint" saturs

Parametra "options.params" saturs, kas uzģenerēts izmantojot 4.1.att. un 4.2.att redzamo vaicājumu redzams 4.4. att.

```
{ params:
    { 'default-graph-uri': '',
        query: 'PREFIX :
        <https://w3id.org/scholarlydata/ontology/conference-ontology.owl#>\nS
        ELECT * WHERE {SELECT ?title WHERE{\n?InProceedings a
        :InProceedings.\n?Proceedings a :Proceedings.\n?Conference a
        :Conference.\nOPTIONAL{?InProceedings :title
        ?title.}\nFILTER(EXISTS{?Conference :acronym
        ?acronym.\nFILTER(STR(?acronym) =
        "ESWC2017")})\n?InProceedings :isPartOf
        ?Proceedings.\n?Proceedings :isProceedingsOf ?Conference.}\nORDER
        BY ?title } LIMIT 50'
    }
}
```

4.11. att. Parametra "options.params" saturs.

Izmantojot HTTP POST pieprasījumu ar URL-iekodētiem parametriem tiek iegūts:

```
HTTP.call("POST", options.endPoint, function(err, resp) {
    //kļūdu un/vai iegūtā rezultāta apstrāde
}
```

4.12. att. ViziQuer HTTP POST pieprasījums ar URL-iekodētiem parametriem

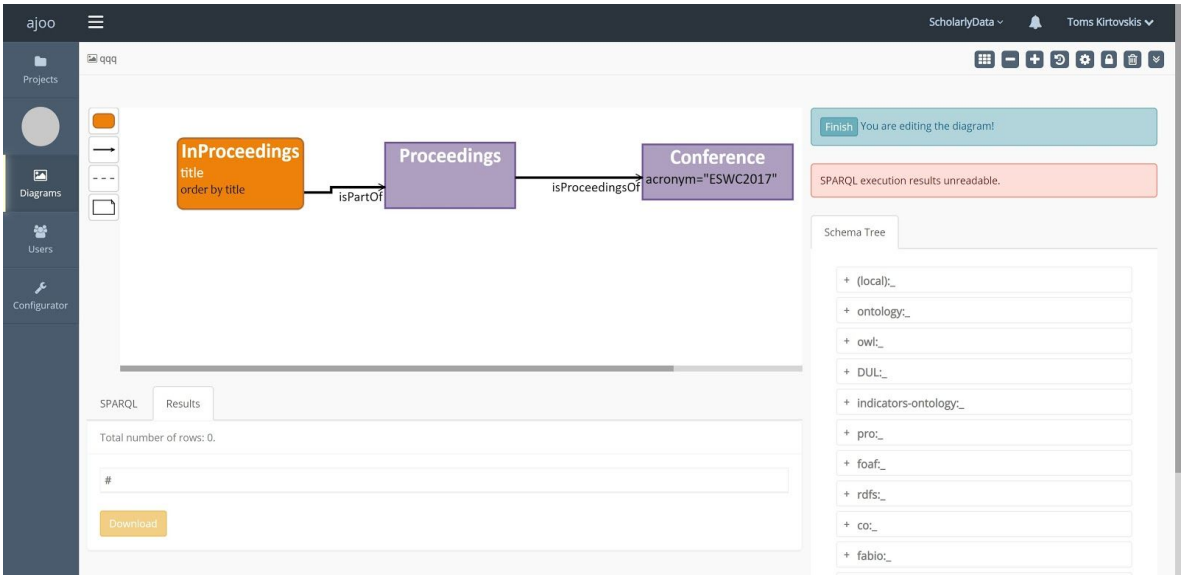
Kā parametrs šai funkcijai tiek padots piekļuves punkts, kam ar URL-iekodēšanas metodi pievienots vaicājums un visi pārējie parametri. Parametra "options.endPoint" saturs, kas uzģenerēts izmantojot 4.1.att. un 4.2.att redzamo vaicājumu redzams 4.6. att.

http://www.scholarlydata.org/sparql/?default-graph-uri=&query=PREFIX%20%3A%20%3Chttps%3A%2F%2Fw3id.org%2Fscholarlydata%2Fontology%2Fconference-ontology.owl%23%3E%20SELECT%20%2A%20WHERE%20%7BSELECT%20%3Ftitle%20WHERE%7B%20%3FInProceedings%20a%20%3AInProceedings.%20%3FProceedings%20a%20%3AProceedings.%20%3FConference%20a%20%3AConference.%20OPTIONAL%7B%3FInProceedings%20%3Atitle%20%3Ftitle.%7D%20FILTER%28EXISTS%7B%3FConference%20%3Aacronym%20%3Facronym.%20FILTER%28STR%28%3Facronym%29%20%3D%20%22ESWC2017%22%29%7D%29%20%3FInProceedings%20%3AisPartOf%20%3FProceedings.%20%3FProceedings%20%3AisProceedingsOf%20%3FConference.%7D%20ORDER%20BY%20%3Ftitle%20%7D%20LIMIT%2050

4.13.att. Parametra "options.endPoint" saturs.

4.2.5. Tiešā HTTP POST pieprasījuma un HTTP POST pieprasījuma ar URL-iekodētiem parametriem rezultātu salīdzināšana ViziQuer rīkā

Veicot tiešo HTTP POST pieprasījumu pār Scholarlydata piekļuves punktu izmantojot 4.2.4. apakšpunktā aprakstīto SPARQL vaicājumu ViziQuer rīkā tiek iegūts kļūdas ziņojums, kas redzams 4.14. attēlā.



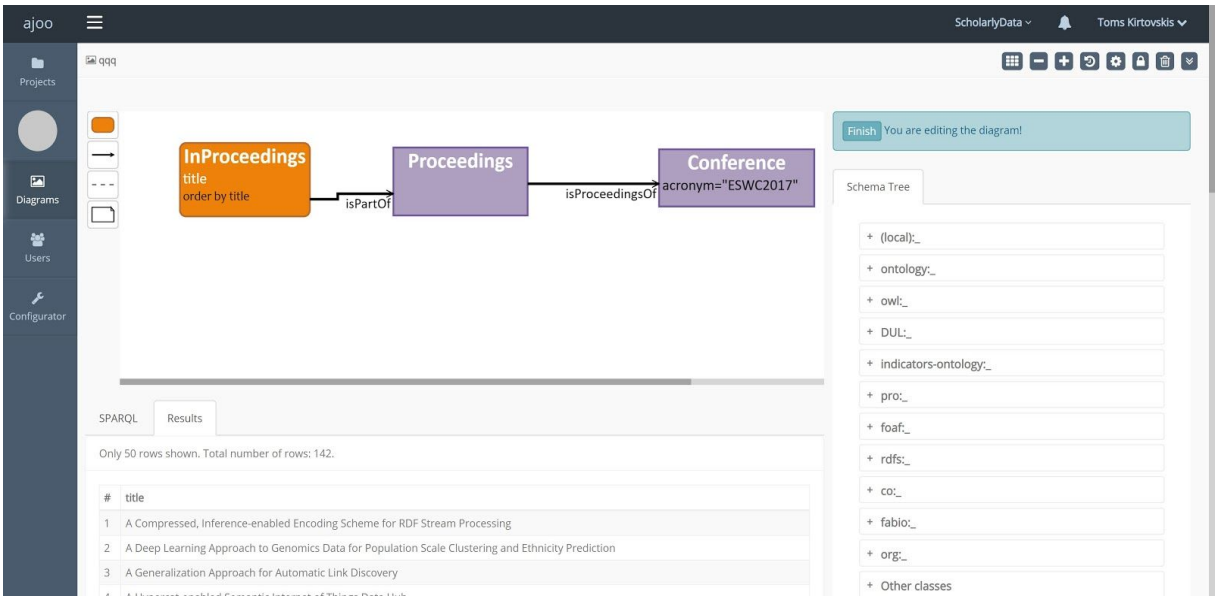
4.14. att. Kļūdas ziņojums, kas iegūts veicot 4.2.4. apakšpunktā aprakstīto SPARQL vaicājumu ViziQuer rīkā

Veicot izpēti serverī, kuru izmantojot tiek darbināts ViziQuer, ir redzams, ka no Scholarlydata piekļuves punkta tiek atgriezts rezultāts ar sekmīgu kodu (statusCode: 200), taču rezultāta saturs norāda uz kļūdaini formulētu vaicājumu.

{ statusCode: 200,
content: 'Virtuoso 37000 Error SP030: SPARQL compiler, line 2: syntax error at \'}\'\n\nSPARQL query:\ndefine sql:big-data-const 0 PREFIX',
headers:
{ server: 'aruba-proxy',
date: 'Thu, 16 May 2019 17:12:31 GMT',
'content-type': 'text/html; charset=UTF-8',
'transfer-encoding': 'chunked',
connection: 'close',
vary: 'Accept-Encoding',
'x-powered-by': 'PHP/7.2.17',
'access-control-allow-origin': '*',
'x-servername': 'ipvsproxy13.ad.aruba.it' },
data: null}

4.15. att. Rezultāts, kas tiek atgriezts no Scholarlydata piekļuves punkta, izmantojot tiešo HTTP POST pieprasījumu.

Veicot HTTP POST pieprasījumu ar URL-iekodētiem parametriem pār Scholarlydata piekļuves punktu un izmantojot 4.2.4. apakšpunktā aprakstīto SPARQL vaicājumu, ViziQuer rīkā tiek iegūts sekmīgs rezultāts, kas redzams 4.9. attēlā.



4.16. att. Sekmīgs rezultāts, kas iegūts veicot 4.2.4. apakšpunktā aprakstīto SPARQL vaicājumu ViziQuer rīkā.

Veicot šādu salīdzinājumu vairākos piekļuves punktos, tiek iegūti līdzīgi rezultāti, tāpēc tiešā HTTP POST pieprasījuma vietā tiek rekomendēts lietot HTTP POST pieprasījumu ar URL-iekodētiem parametriem.

4.2.6. Rekomendācijas ViziQuer darbības uzlabošanai

Testējot ViziQuer darbību sazinoties ar dažādiem SPARQL piekļuves autors novēroja, to, ka ViziQuer rīks nesekmīgi veic saziņu ar vairākiem no tiem (piemēram, nespēj noteikt atgriezto ierakstu skaitu, vai rodas problēmas ar iegūtā rezultāta interpretēšanu).

Veicot ViziQuer rīka koda analīzi autors uzzināja, ka saziņai ar piekļuves punktiem tiek pielietots tiešais HTTP POST pieprasījums. Veicot tālāku izpēti autors noskaidroja to, ka url-iekodētais HTTP POST pieprasījums šādai saziņai ir atbilstošāks un tāpēc rekomendē url-iekodētā HTTP POST pieprasījuma izmantošanu ViziQuer un SPARQL piekļuves punktu savstarpējai saziņai. Autora rekomendācija ViziQur rīka koda uzlabošanai pievienota kā 1. pielikums. Url-iekodētā HTTP POST pieprasījuma detalizētāks apraksts atrodams 4.3. apakšnodaļā.

4.3. URL iekodētais HTTP POST pieprasījums

Lai nodrošinātu savstarpējo savietojamību visas pasaules mērogā, URI tiek šifrēti vienotā veidā. Tā kā URI izveidē tiek lietoti tikai aptuveni 60 simboli, tad, lai nodrošinātu savietojamību ar vismaz lielāko daļo no pasaulē pielietotajiem simboliem, tiek lietots sekojošs no diviem soļiem sastāvošs process:

- Pārveidot simbolu virkni par baitu virkni izmantojot UTF-8 kodējumu
- Pārveidot katru baitu, kas nav ASCII burts vai cipars par %HH, kur HH ir konkrētā baita heksadecimālā vērtība. Piemēram, teksta virkne “François” tiktu pārveidota par “Fran%C3%A7ois” ("ç" tiek pārveidots UTF-8 kodējumā kā divi baiti C3 (hex) unA7 (hex), kas tālāk tiek pārveidoti par trim simboliem "%c3" un "%a7".)

Šādā veidā URI var tikt ievērojami pagarināti (viens Unikoda simbols var tikt aizstāts ar līdz pat 9 ASCII simboliem), bet pamatideja nodrošina to, ka pārlūkprogrammām šīs virknes ir jāattēlo tikai atšifrētā veidā un daudzi protokoli spēj nosūtīt UTF-8 kodējumu arī bez pārveidošanass uz %HH formātu. [14]

4.3.1. URL iekodēšana

Par URL iekodēšanu sauc noteiktu kādā URL esošu simbolu aizvietošanu ar vienu vai vairākiem simbolu trijniekiem, kas sastāv no procentu apzīmējuma “%”, kam seko divi heksadecimālās skaitīšanas sistēmas skaitļi. Šie abi heksadecimālās skaitīšanas sistēmas skaitļi katrā trijniekā apzīmē aizvietoto simbolu skaitliskās vērtības.

Jēdziens URL iekodēšana ir nedaudz neprecīzs, jo šī iekodēšanas procedūra tiek pielietota ne tikai darbojoties ar URL (Uniformi Resursu Lokatori), bet var tikt izmantota arī darbā ar cita veida URI (Uniformi Resursu Identifikatori), kā, piemēram, URN (Uniformi Resursu Nosaukumi). Šī iemesla dēļ precīzāks apzīmējums būtu procent-iekodēšana. [14]

4.3.2. URL izmantojamie simboli

URI iekļaujamie simboli tiek iedalīti rezervētajos un nerezervētajos (kā arī procentu apzīmējums, kas tiek izmantots kā daļa no procent-iekodēšanas). Rezervētie simboli ir tie simboli, kuriem konkrētos gadījumos ir speciāla nozīme, nerezervētajiem simboliem nav speciālas nozīmes. Izmantojot procent-iekodēšanu, simboli, kuri citos gadījumos nebūtu URI iekļaujamo simbolu grupā, tiek aizstāsti ar simboliem, kas ir iekļauti URI iekļaujamo simbolu

grupā. Rezervēto un nerezervēto simbolu grupas un nosacījumi, pie kuriem daži simboli iegūst speciālu nozīmi, ir tikuši nedaudz mainīti ar katru nākošo URI un URI shēmu aprakstošās specifikācijas revīziju.

Saskaņā ar RFC 3986, URL ir jāsatur simboli tikai no strikti definētas grupas, kurā ir iepriekš noteikti rezervētie un nerezervētie ASCII simboli. Citus simbolus URL saturēt nedrīkst.

Nerezervētie simboli var tikt iekodēti, bet tiem nevajadzētu būt iekodētiem. Nerezervētie simboli ir:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9 - _ . ~

Rezervētajiem simboliem iekodēšana var tikt pielietota tikai noteiktos gadījumos. Rezervētie simboli ir:

! * ' ( ) ; : @ & = + $, / ? % # [ ]

[14]

4.3.3. Teksta iekodēšana/atšifrēšana

RFC 3986 [11] nenosaka, saskaņā ar kuru zīmju iekodēšanas tabulu būtu jāiekodē ne-ASCII zīmes (piem., umlauti ä, ö, ü). Tā kā URL iekodēšana izmanto heksadecimālu zīmju pārus, un heksadecimālu zīmju pāris ir līdzvērtīgs 8 bitiem, teorētiski būtu iespējas izmantot vienu no 8 bitu koda lapām, kas satur ASCII neesošas zīmes (piem., ISO-8859-1 umlautiem).

No otras puses, tā kā daudzām valodām ir pašām savas 8 bitu koda lapas, rīkošanās ar visām šīm daudzajām 8 bitu koda lapām būtu diezgan neparocīga. Dažas valodas pat neietilpst 8 bitu koda lapā (piem., ķīniešu). Tādēļ RFC 3629 [12] ierosina izmantot UTF-8 zīmju iekodēšanas tabulu ASCII neesošām zīmēm. Minētais instruments ņem to vērā un piedāvā izvēlēties starp ASCII un UTF-8 zīmju iekodēšanas tabulām. Ja izvēlēsieties ASCII iekodēšanas tabulu, parādīsies brīdinājuma ziņojums, ja URL iekodētais/atšifrētais teksts saturēs ASCII neesošās zīmes. [14]

4.3.4. URL iekodēšanas lietojums

Kad dati, kas ievadīti HTML formātā ir nosūtīti, noformējuma lauka nosaukumi un vērtības tiek iekodētas un nosūtītas uz serveri HTTP pieprasījuma ziņojumā, izmantojot GET vai POST metodi, vai vēsturiski, caur e-pastu. Pēc noklusējuma lietotā iekodēšana balstās uz ļoti agrīnu versiju vispārējiem URI procentuālās iekodēšanas noteikumiem ar dažām

modifikācijām, piemēram, jaunlīniju normalizāciju un atstarpju aizvietošanu ar “+”, nevis “%20”/ Šādā veidā iekodēts MIME datu tips ir application/x-www-form-urlencoded, un šobrīd tiek definēts (joprojām ļoti novecojušā veidā) HTML un XForms specifikācijās. Piedevām, CGI specifikācijas satur noteikumus, kas nosaka, kā tīkla serveri atšifrē šī tipa datus un padara tos pieejamus aplikācijām.

Kad nosūta HTTP GET pieprasījumu, application/x-www-form-urlencoded dati tiek iekļauti vaicājuma komponentē pieprasījuma URI. Nosūtot HTTP POST pieprasījumu caur e-pastu, dati tiek novietoti ziņojuma saturā, un medija tips tiek norādīts ziņojuma satura galvenē. [14]

5. Piekļuves punktu novērtējums

Veicot šo darbu, autors saskārās ar samērā daudz un dažādiem piekļuves punktiem, taču detalizēti apraksti visbiežāk bija pieejami tikai populārākajiem no tiem. Tāpēc autors izvēlējās vairākus no tiem un veica to izpēti gan nedaudz aplūkojot to saturu, gan veicot iepriekš sagatavotus SPARQL vaicājumus, lai apkopojot un analizējot iegūtos rezultātus varētu veikt secinājumus par konkrēto piekļuves punktu statistikas rādītājiem, darbību, kā arī to piemērotību un saderību darbam ar ViziQuer, tajā skaitā arī piekļuves punktu shēmas iegūšanu, izmantojot shēmu iegūšanas rīku[15].

5.1. Populārāko iekļuves punktu apraksts

Scholarlydata: Scholarlydata datu kopa, kas izmanto Semantic Web Dog Food (SWDF), tā, lai saglabātu šo datu kopu labā stāvoklī un nodrošinātu tās izaugsmi. Tas ir unikāls datu modelis konferenču ontoloģiju uzglabāšanai, kas satur "Semantic Web Conference Ontology" uzlabojošus elementus, taču turpina izmantot ontoloģiju izstrādes labās prakses ieteikumus. Visi šobrīd pieejamie dati var tikt iegūti vairākos dažādos formātos (tajā skaitā HTML, RDF/XML, Turtle, N-TRIPLES un JSON-LD) izmantojot URI dereferencēšanu, veicot SPARQL vaicājumus, vai tos lejupielādejot kā atsevišķus RDF kopas katrai konkrētajai konferencei vai semināram. Šim piekļuves punktam ir pieejama gan ontologija, gan datu shēma, tādējādi nodrošinot to, ka šis piekļuves punkts ir saderīgs darbam ar ViziQuer. Saite:http://www.scholarlydata.org/sparql/

Europeana: Europeana SPARQL API sniedz iespēju aplūkot Europeana datu saistību ar tādiem ārējiem datu avotiem kā, piemēram, VIAF, Iconclass, Getty Vocabularies (AAT), Geonames, Wikidata un DBPedia. Šis API nodrošina arī veidu kā piekļūt un apzināt Europeana struktūrētos metadatus (piemēram, noskaidrot kādi ir kāda konkrēta perioda gleznotāji, kam ir zināmi vismaz pieci darbi,kas ir pieejami Europeana datu bāzē). Visi šobrīd pieejamie dati var tikt iegūti vairākos dažādos formātos (tajā skaitā HTML, RDF/XML, Turtle, N-TRIPLES un JSON-LD), izmantojot URI dereferencēšanu, veicot SPARQL vaicājumus vai tos lejupielādejot kā atsevišķus RDF kopas. Saite: http://sparql.europeana.eu/

Europena izstrādātāji un uzturētāji piedāvā arī vairākus SPARQL vaicājumu piemērus:

1)
```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
PREFIX ore: <http://www.openarchives.org/ore/terms/>
SELECT ?title ?creator ?mediaURL ?date
WHERE {
 ?CHO edm:type "SOUND" ;
    ore:proxyIn ?proxy;
    dc:title ?title ;
    dc:creator ?creator ;
    dc:date ?date .
 ?proxy edm:isShownBy ?mediaURL .
} LIMIT 100
```

5.1. att. Pirmie 100 “SOUND” tipa ieraksti, kam ir nosaukums, radītājs, media URL un gads

2)
```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
PREFIX ore: <http://www.openarchives.org/ore/terms/>
SELECT ?title ?creator ?mediaURL ?date
WHERE {
 ?CHO edm:type "SOUND" ;
    ore:proxyIn ?proxy;
    dc:title ?title ;
    dc:creator ?creator;
    dc:date ?date .
 ?proxy edm:isShownBy ?mediaURL .
 FILTER (?date > "1780" && ?date < "1930")
}
ORDER BY asc (?date)
LIMIT 100
```

5.2. att. 5.1. att. esošais piemērs papildināts ar FILTER un ORDER BY

3)
```
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
SELECT ?DataProvider
WHERE { ?Aggregation edm:dataProvider ?DataProvider }`
```

5.3. att. Saraksts ar Europeana saturu sastādošajiem datu provaideriem

4)
```
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
SELECT DISTINCT ?Dataset
WHERE {
 ?Aggregation edm:datasetName ?Dataset ;
    edm:country "Italy"}
```

5.4. att. Saraksts ar visām datu kopām no Itālijas

5)

```
PREFIX ore: <http://www.openarchives.org/ore/terms/>
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
SELECT DISTINCT ?ProvidedCHO ?year
WHERE {
 ?Aggregation edm:aggregatedCHO ?ProvidedCHO ;
    edm:country "France" .
 ?Proxy ore:proxyFor ?ProvidedCHO ;
    edm:year ?year .
 FILTER (?year > "1700" && ?year < "1800")
} ORDER BY asc(?year) LIMIT 100
```

5.5. att. Europena objekti no 18. gs Francijas

6)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
SELECT ?Agent
WHERE { ?Agent rdf:type edm:Agent }
LIMIT 100
```

5.6. att. Saraksts, kas satur elementus ar edm:Agent tipu

7)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ore: <http://www.openarchives.org/ore/terms/>
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
SELECT DISTINCT ?ProvidedCHO
WHERE {
 ?Place rdf:type edm:Place .
 ?Proxy ?property ?Place ;
    ore:proxyIn ?Aggregation .
 ?Aggregation edm:aggregatedCHO ?ProvidedCHO                    }
```

5.7. att. Europena objekti, kas saistīti ar edm:Place tipu

8)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ore: <http://www.openarchives.org/ore/terms/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
SELECT DISTINCT ?ProvidedCHO
WHERE {  ?Concept rdf:type skos:Concept .
 FILTER strstarts(str(?Concept), "http://vocab.getty.edu/aat/") .
 ?Proxy ?property ?Concept ;
    ore:proxyIn ?Aggregation .
 ?Aggregation edm:aggregatedCHO ?ProvidedCHO
} LIMIT 100
```

5.8. att. Europeana objekti, kas piesaistīti, izmantojot skos:concept no Getty vārdnīcas

UNESCO Thesaurus: UNESCO Thesaurus piekļuves punkts ir sakārtots un strukturēts saraksts, kas satur jēdzienus, kas tiek lietoti tēmu analīzē un dažādu dokumentu un publikāciju par tādām tēmām kā izglītība, kultūra, dabas zinātnes, sociālās un humanitārās zinātnes, komunikācija un sakari iegūšanu. Šo piekļuves punktu iedala septiņās lielākās tēmās, kas ir sadalītas apakštēmās (microthesauri), tādējādi nodrošinot ērtāku kādas konkrētas tēmas pārlūkošanu. Visi šobrīd pieejamie dati var tikt iegūti vairākos dažādos formātos (tajā skaitā HTML, RDF/XML, Turtle, N-TRIPLES un JSON-LD), izmantojot URI dereferencēšanu, veicot SPARQL vaicājumus, vai tos lejupielādejot kā atsevišķus RDF kopas. UNESCO Thesaurus izstrādātāji un veidotāji piedāvā tādus vaicājumu piemērus kā:

- Neliels esošā satura paraugs;
- Visu francisko mikrotēzauru saraksts;
- Visu tēmu kādā zināšanu apgabalā saraksts;
- Visu tēmu saraksts;
- Visu mikrotēzauru saraksts;
- Visu angļu-franču tulkojumu saraksts;
- Visu angļu-krievu tulkojumu saraksts;
- Visu, kas jaunākas par kādu datumu saraksts;
- Valstu saraksts;
- Visu, kādai tēmai piešķirto īpašību saraksts;
- Hierarhiska tēmu tabula, kas satur ID;
- u.c.

Neliels ieskats šī piekļuves punkta parametros, kas aplūkojami zemāk esošajā 4.1. tabulā.

Saite: http://vocabularies.unesco.org/sparql-form/

5.1. tabula

UNESCO Thesaurus piekļuves punkta parametri

Triplets	69766
skos:Collections	96
skos:ConceptScheme	1
skos:Concepts	4408
skos:altLabel	13868

skos:narrower / skos:broader	4244 / 4244
skos:prefLabel	17980
skos:related	12196
Atjaunots	11.10.2013, 2:00 AM (UTC+02:00)
Izveidots	20.05.2013, 4:23 PM (UTC+02:00)

Eiropas Bioinformātikas institūts (European Bioinformatics Institute (EMBL-EBI)): EMBL-EBI platforma ir izveidota ar mērķi apvienot vairākus EMBL-EBI resursus, kas nodrošina pieeju institūta datiem, izmantojot Semantiskā Tīmekļa tehnoloģijas. Tas sniedz iespēju visus šos resursus pārlūkot, izmantojot W3C SPARQL vaicājumu valodu. Šis projekts apvieno tādus EMBL-EBI iekšejos resursus kā:

- **Biosamples**. BioSample Datubāze (BioSD) ir datubāze, kas satur informāciju par bioloģiskiem paraugiem un DNS pētījumiem.
- **ChEMBL.** ChEMBL ir datubāze, kas satur informāciju par bioaktīvām molekulām ar medikamantiem pielīdzināmām īpašībām.
- **Ensembl.** Ensembl satur informāciju par mugurkaulnieku genomiem, kas palīdz veikt pētījumus salīdzinošajā ģenētikā, evolūcijas pētījumos utt. Ensembl satur komentārus par gēniem, dažādus aprēkinus, slimību aprakstošus datus utt.
- **Reactome.** Reactome ir atvērtā pirmkoda datubāze, kuras mērķis ir izveidot intuitīvu bioinformātikas rīku, datu vizualizācijai, interpretācijai un analīzei.
- **Expression Atlas.** Šis resurss satur informāciju par gēnu un olbaltumvielu izplatību dažādās sugās un bioloģiskajos apstākļos.
- **Ontology Lookup Service (OLS)**. Resurss, kas veidots, lai nodrošinātu ērtu pieeju jaunākajām biomedicīnisko ontoloģiju versijām.

Visi šobrīd pieejamie dati var tikt iegūti vairākos dažādos formātos (tajā skaitā HTML, RDF/XML, Turtle, N-TRIPLES un JSON-LD). EMBL-EBI platformas izstrādātāji un veidotāji piedāvā aptuveni 33 dažādus vaicājumu piemērus, katram resursam aptuveni 3. Saite: https://www.ebi.ac.uk/rdf/services/sparql

5.2. Piekļuves punktu izpēte

Lai veiktu detalizētāku SPARQL piekļuves punktu izpēti, autors apkopoja vairākus statistikas rādītājus par populāriem un/vai viegli un ērti izmantojamiem SPARQL piekļuves punktiem. Šie statistikas rādītāji tika iegūti, izmantojot vairākus SPARQL vaicājumus un analizējot un apkopojot no katra piekļuves punkta iegūtos rezultātus. Izmantojot zemāk redzamo SPARQL vaicājumu, tika noskaidroti sekojoši raksturlielumi: klašu kopskaits, visu klašu nosaukumi, tajā skaitā pēc instanču skaita lielākās klases nosaukums, instanču skaits katrā klasē, kā arī, veicot statistiskus aprēķinus, klašu skaits ar vismaz 5 un vismaz 10 instancēm, kā arī to procentuālais apjoms. SPARQL vaicājums, kas tika izmantots, lai iegūtu šos piekļuves punktus raksturojošos lielumus tika izmantots:

select ?C (count(?x) as ?cx) where {?x a ?C} group by ?C order by desc(?cx)

5.2. tabula

Populārāko piekļuves punktu pamata raksturlielumi

Apzīmējums	Klašu skaits	Lielākā klase	Instanču skaits	Links
Linked GeoData	191	www.w3.org/2003/01/geo/wgs84_pos#Point	4277929	http://geo.linkeddata.es/sparql
pubmed.bio2rdf.org	661	bio2rdf.org/dbsnp_vocabulary:Resource	24803866	http://pubmed.bio2rdf.org/sparql
Europeana	217	www.openarchives.org/ore/terms/Proxy	72488639	http://sparql.europeana.eu/
ScholarlyData	90	www.w3.org/2000/01/rdf-schema#Resource	97143	http://www.scholarlydata.org/sparql/
Unesco	8	www.w3.org/2004/02/skos/core#Concept	4417	http://vocabularies.unesco.org/sparql-form/
SKOS	4	http://www.w3.org/2004/02/skos/core#Concept	27042	http://skos.um.es/sparql/
Data.Ontotext	69	org:Membership	1316949	http://data.ontotext.com/sparql
statistics.gov	269	purl.org/linked-data/cube#Observation	92600418	https://statistics.gov.scot/sparql
LRI IS	36	schema.org/Article	523	https://sparql.lri.fr/sparql
wikiPathways	50	vocabularies.wikipathways.org/gpml#Point	330090	http://sparql.wikipathways.org/

webThesauru	519	schema.semantic-web.at/ppt/history#WorkflowHistoryEvent	6484	http://vocabulary.semantic-web.at/PoolParty/sparql/semweb
Linked Movie Database	159	data.linkedmdb.org/resource/movie/performance	197271	http://85.254.199.72:8890/sparql
DBtune	9	purl.org/NET/c4dm/timeline.owl#RelativeTimeLine	8484	http://dbtune.org/jamendo/cliopatria/yasgui/index.html
OpenLink Virtuoso	724	demo.openlinksw.com/schemas/PrestoDB/lineitem	1972634	http://demo.openlinksw.com/sparql/

5.3. tabula

Populārāko piekļuves punktu pamata raksturlielumi (tabulas turpinājums)

Apzīmējums	Klašu skaits	Klašu skaits ar 5 un vairāk instancēm	Procentuāli	Klašu skaits ar 10 un vairāk instancēm	Procentuāli
Linked GeoData	191	135	70,68%	120	62,83%
pubmed.bio2rdf.org	661	585	88,50%	564	85,33%
Europeana	217	80	36,87%	66	30,41%
ScholarlyData	90	71	78,89%	64	71,11%
Unesco	8	5	62,50%	4	50,00%
Data.Ontotext	69	56	81,16%	53	76,81%
statistics.gov	269	138	51,30%	73	27,14%
LRI IS	36	20	55,56%	15	41,67%
wikiPathways	50	38	76,00%	35	70,00%
webThesauru	519	48	9,25%	33	6,36%
Linked Movie Database	159	133	83,65%	104	65,41%
DBtune	9	9	100,00%	9	100,00%
OpenLink Virtuoso	724	180	24,86%	128	17,68%

Tā kā, lai sekmīgi varētu izmantot visu ViziQuer piedāvāto funkcionalitāti, ir nepieciešams izmantot arī SPARQL piekļuves punktu shēmas iegūšanas rīku [15], autors veica izpēti par to, kādus rezultātus var iegūt no populārākajiem un/vai ērti piejamajiem SPARQL piekļuves punktiem, izmantojot SPARQL vaicājumus, kas tiek izmantoti shēmas

iegūšanas rīka darbības procesā, vai tiem pielīdzināmus vaicājumus. Autors izmantoja 3 dažādus SPARQL vaicājumus un apkopoja informāciju par iegūtajiem rezultātiem, pievēršot uzmanību atgrieztajam instanču skaitam un ātrdarbības salīzinājumam divu pirmo vaicājumu gadījumā. Tika izmantoti sekojoši vaicājumi -

1) vaicājums 1:
   select (count(?x) as ?cx) where {?x a :ccc } :ccc - lielākā klase
2) vaicājums 2:
   select (count(distinct ?x) as ?cx) where {?x a :ccc } :ccc - lielākā klase
3) vaicājums 3:
   select (count(?x) as ?cx)
   where {?x a :ccc. FILTER NOT EXISTS{?x a :ddd } }
   - saskaitīt tos x, kas ir klasē ccc, un nav klasē ddd - otra lielākā klase

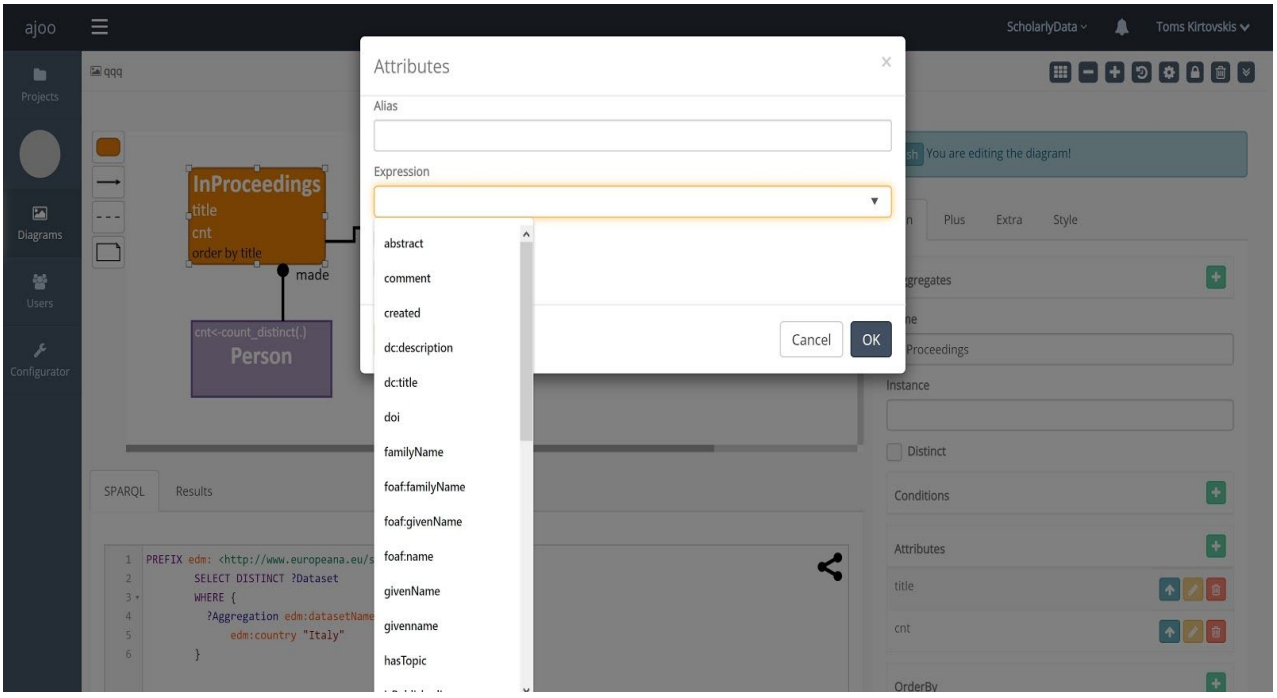
5.4. tabula

Populārāko piekļuves punktu analīze un shēmas iegūšana

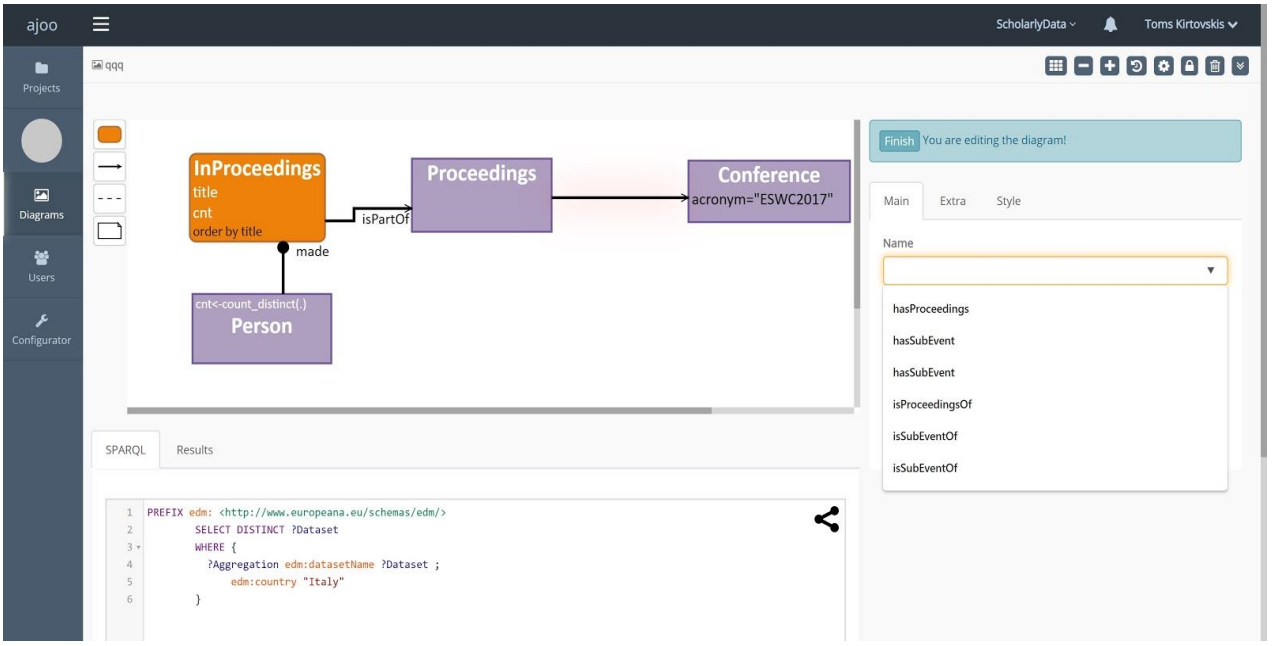
Nosaukums	Klašu skaits	Vaicājums 1	Ilgums	Vaicājums 2	Ilgums	Vaicājums 3	sekmīga shēmas iegūšana	Ilgums
Linked GeoData	191	4277929	1,12 sec	4277929	6,93 sec	-	nē	48 min
pubmed.bio2rdf.org	661	24803866	1,43 sec	24790642	36,86 sec	5801	jā	3,7 min
Europeana	217	72488639	504 ms	72483933	55,84 sec	72488639	jā	~ 6h
ScholarlyData	90	97143	144 ms	71539	211 ms	10354	jā	3,9 min
Unesco	8	4417	71 ms	4417	81 ms	4417	nē	-
SKOS	4	27042	191 ms	-	-	-	nē	-
Data.Ontotext	69	1487211	3,6 sec	1487211	3,6 sec	1487211	nē	-
statistics.gov	277	93309095	399 ms	-	-	-	nē	-
LRI IS	36	523	59 ms	514	158 ms	523	jā	59,85 sec
wikiPathways	50	332479	81 ms	332479	1,27 sec	332479	jā	3,1 min
webThesauru	519	6487	371 ms	6487	369 ms	6487	jā	33,5 min
Linked Movie Database	159	197271	39 ms	197271	227 ms	197271	jā	36,75 sec
DBtune	13	335925	1,86 sec	335925	2,20 sec	233121	nē	-
OpenLink Virtuoso	1243	6001204	282 ms	6001204	13,13 sec	6001204	nē	-

Aplūkojot 5.4. tabulā esošos SPARQL piekļuves punktus aprakstošos lielumus var izdarīt vairākus secinājumus par šo piekļuves punktu darbības īpatnībām. Piemēram, to, ka no visiem minētajiem piekļuves puktiem nav iespējams iegūt shēmu, kā arī to, ka šī shēmas

ieguve var notikt samērā ilgstoši un to, ka kopējais klašu skaits vai instanču skaits klasēs nav galvenais ielādes iespējamību noteicošais parametrs. Interesanti ir arī tas, ka ne visi piekļuves punkti ir saderīgi ar filtru izmantošanu vaicājumos (piemēram SKOS saite: http://skos.um.es/sparql/). Tā kā, lai pilnīgi izmantotu Viziquer rīkā pieejamās iespējas, ir nepieciešams iegūt katra konkrētā SPARQL piekļuves punkta shēmu, tad tālākai izpētei autors izmantoja tikai tos SPARQL piekļuves punktus, kuru shēmas sekmīgi izdevās iegūt izmantojot piekļuves punktu shēmu iegūšanas rīku [15]. Attēlos 5.9. un 5.10. redzami attiecīgi addAttributes un addLink dialogi, kas norāda uz sekmīgu shēmas iegūšanu ar SPARQL piekļuves punktu datu shēmas iegūšanas rīku un iegūtu shēmu ielādi ViziQuer rīkā.



5.9. att. addAttributes dialogs



5.10. att. addLink dialogs

5.5. tabula

Populārāko piekļuves punktu analīze un ielāde ViziQuer

Apzīmējums	Klašu skaits	Sekmīga ielāde ViziQuer rīkā
pubmed.bio2rdf.org	661	jā
Europeana	217	jā
ScholarlyData	90	jā
LRI IS	36	jā
wikiPathways	50	jā
webThesauru	519	jā
Linked Movie Database	159	jā

Augstāk esošajā 5.5. tabulā redzami SPARQL piekļuves punkti, kuru shēmas autoram sekmīgi izdevās ielādēt ViziQuer rīkā. Izmantojot ar SPARQL piekļuves punktu shēmas iegūšanas rīku izveidoto konkrēto piekļuves punktu datu shēmas, ViziQuer tiešsasites rīka lietotāji var veidot vizuālus vaicājumus un, izmantojot ViziQuer rīkā nodrošināto funkcionalitāti, ģenerēt SPARQL vaicājumus, ar kuru palīdzību veikt SPARQL piekļuves punktu pārlūkošanu.

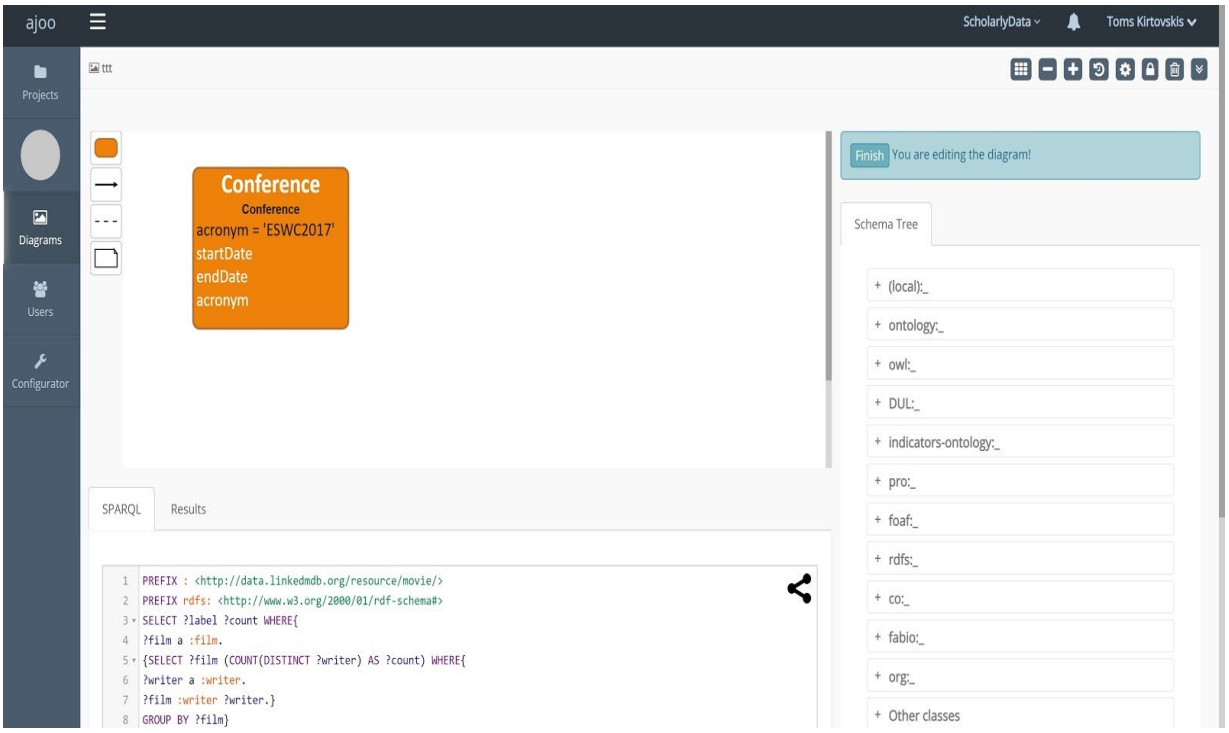
5.6. tabula

Piekļuves punktu, kuriem nav pieejama pilnvērtīga ViziQuer funkcionalitāte, apkopojums

Apzīmējums	Saite	iemesls
Linked GeoData	http://geo.linkeddata.es/sparql	nepilnīgas datu shēmas iegūšana
Unesco	http://vocabularies.unesco.org/sparql-form/	nesekmīga shēmas iegūšana
SKOS	http://skos.um.es/sparql/	nepilnīgas datu shēmas iegūšana
Data.Ontotext	http://data.ontotext.com/sparql	nesekmīga shēmas iegūšana
statistics.gov	https://statistics.gov.scot/sparql	nesekmīga shēmas iegūšana
DBtune	http://dbtune.org/jamendo/cliopatria/yasgui/index.html	nesekmīga shēmas iegūšana
OpenLink Virtuoso	http://demo.openlinksw.com/sparql/	nesekmīga shēmas iegūšana

Augstāk esošajā 5.5. tabulā apkopoti tie autora apskatītie SPARQL piekļuves punkti, kuriem nav pieejama pilvērtīga ViziQuer rīka funkcionalitāte. Kā galvenais iemesls tam ir grūtības ar datu shēmas iegūšanu, kas manāmi apgrūti šo piekļuves punktu apstrādi ViziQuer rīkā. Tabulā minētajiem piekļuves punktiem ir iespējams pieslēgties, izmantojot ViziQuer, kas nozīmē to, ka ir iespējams izpildīt SPARQL vaicājumus, taču ir ļoti apgrūtinoši izveidot konkrētajam piekļuves puktam atbilstošus vizuālus vaicājumus.

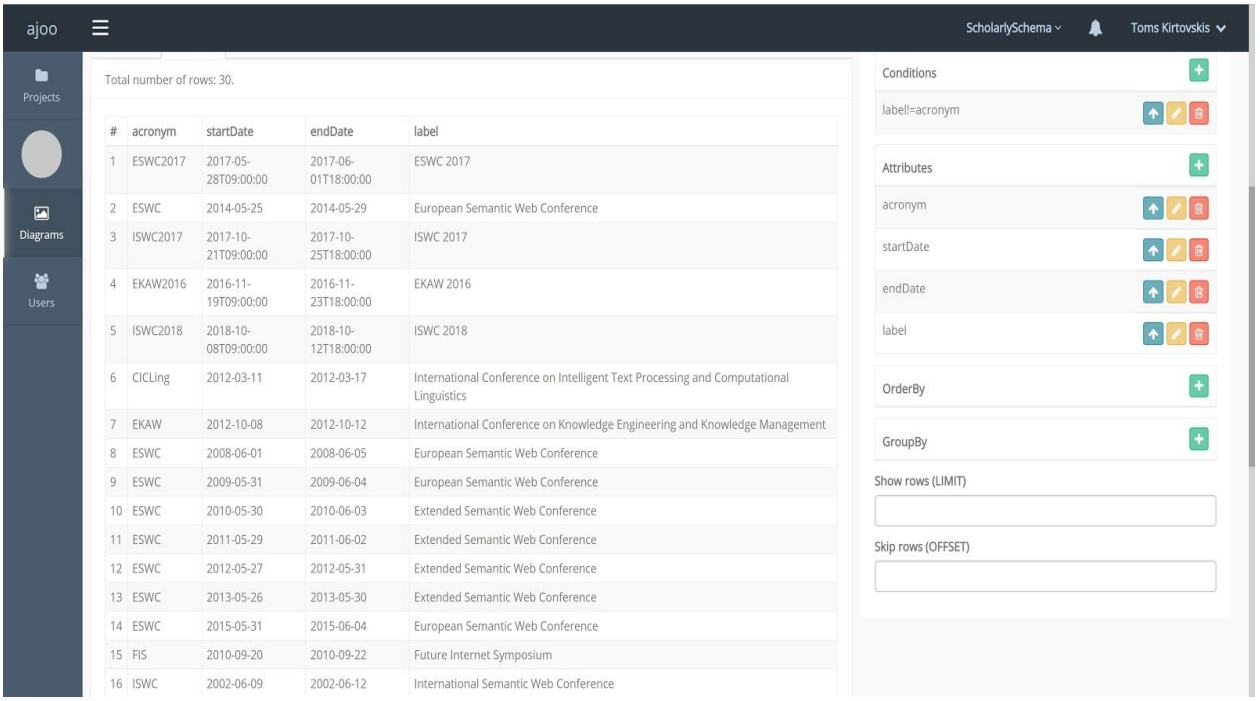
5.2.1. Vaicājumi piemēri, izmantojot ScholarlyData piekļuves punktu.



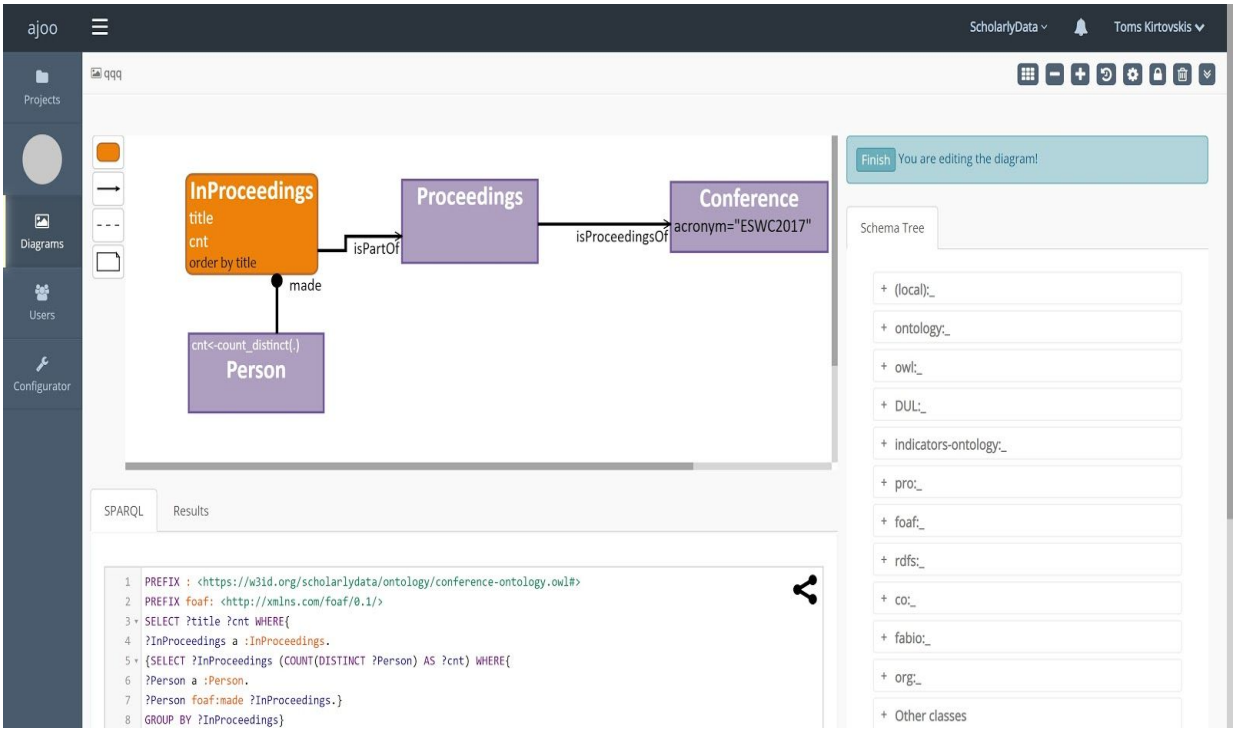
5.11. att. Vizuāls vaicājums, kas attēlo visas konferences, to akronīmus, sākuma un beigu datumus un nosaukumus.

```
PREFIX : <https://w3id.org/scholarlydata/ontology/conference-ontology.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?acronym ?startDate ?endDate ?label WHERE{
?Conference a :Conference.
OPTIONAL{?Conference :acronym ?acronym.}
OPTIONAL{?Conference :startDate ?startDate.}
OPTIONAL{?Conference :endDate ?endDate.}
OPTIONAL{?Conference rdfs:label ?label.}
FILTER(?label != ?acronym)}
```

5.12. att. Sparql vaicājums, kas attēlo visas konferences, to akronīmus, sākuma un beigu datumus un nosaukumus



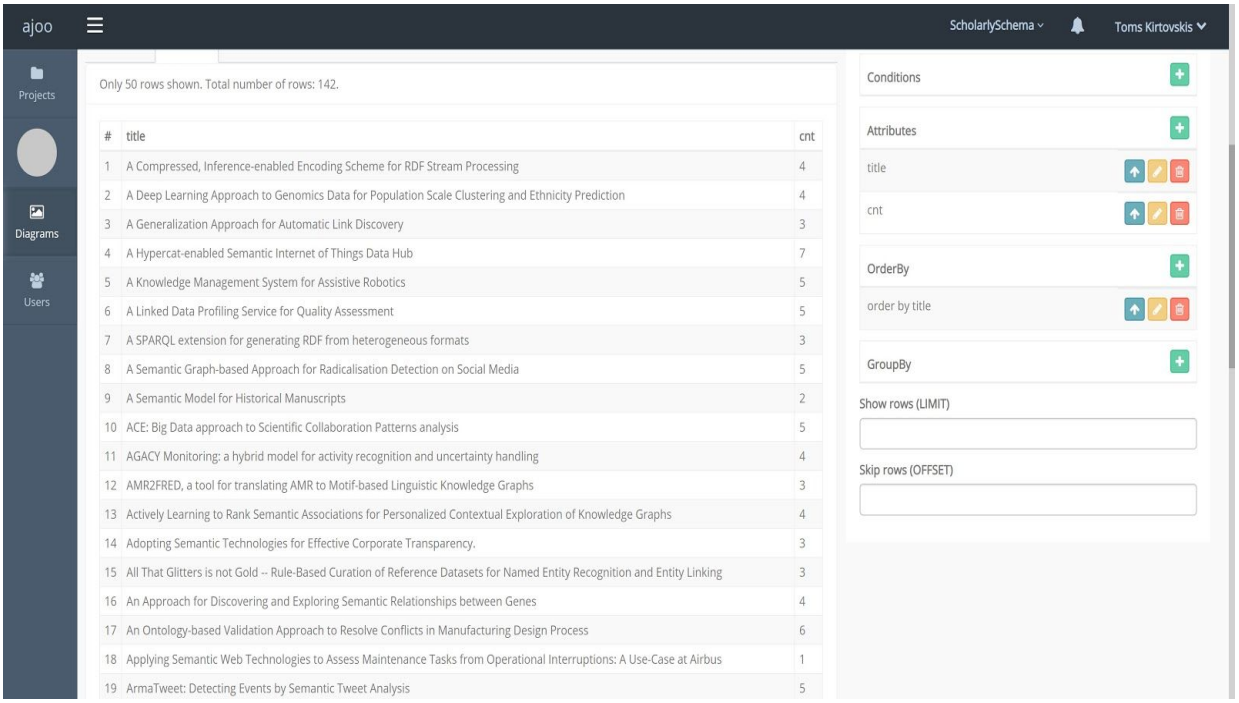
5.13. att. Vaicājuma rezultāts, kas attēlo visas konferences, to akronīmus, sākuma un beigu datumus un nosaukumus.



5.14. att. Vizuāls vaicājums, kas attēlo visas publikācijas, kas tika publicētas ESWC2017 konferencē un šo publikāciju autoru skaitu.

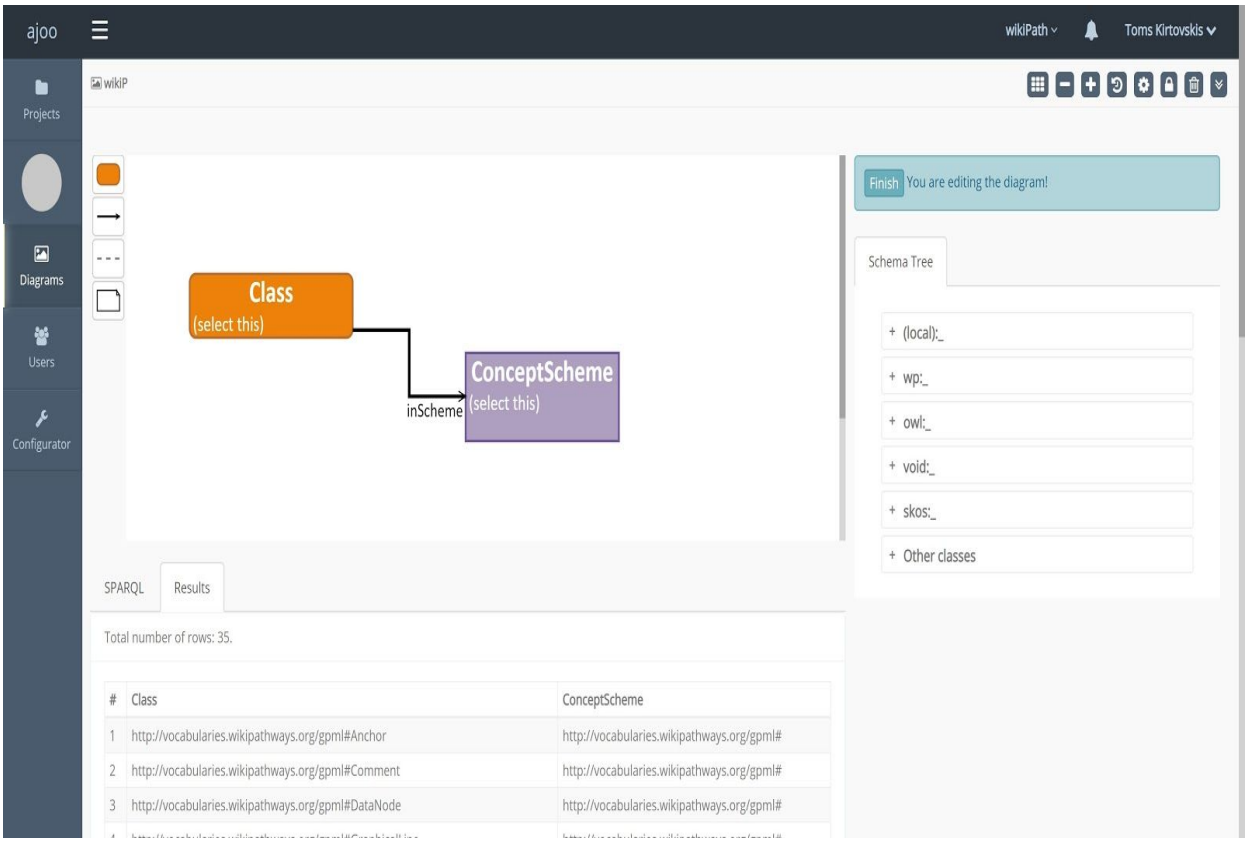
```
PREFIX : <https://w3id.org/scholarlydata/ontology/conference-ontology.owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?title ?cnt WHERE{
?InProceedings a :InProceedings.
{SELECT ?InProceedings (COUNT(DISTINCT ?Person) AS ?cnt) WHERE{
?Person a :Person.
?Person foaf:made ?InProceedings.}
GROUP BY ?InProceedings}
?Proceedings a :Proceedings.
?Conference a :Conference.
OPTIONAL{?InProceedings :title ?title.}
FILTER(EXISTS{?Conference :acronym ?acronym.
FILTER(STR(?acronym) = "ESWC2017")})
?InProceedings :isPartOf ?Proceedings.
?Proceedings :isProceedingsOf ?Conference.}
ORDER BY ?title
```

5.15. att. SPARQL vaicājums, kas attēlo visas publikācijas, kas tika publicētas ESWC2017 konferencē un šo publikāciju autoru skaitu.



5.16. att. Vaicājuma rezultāts, kas attēlo visas publikācijas, kas tika publicētas ESWC2017 konferencē un šo publikāciju autoru skaitu.

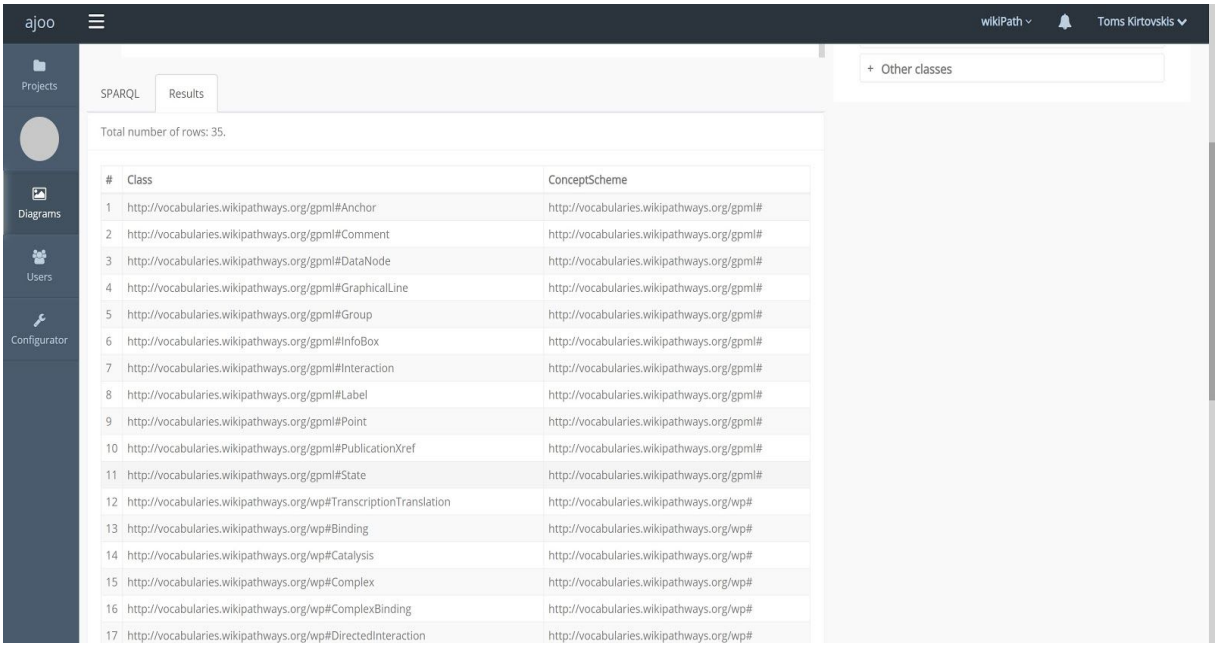
5.2.2. Vaicājumu piemēri, izmantojot wikipathways piekļuves punktu.



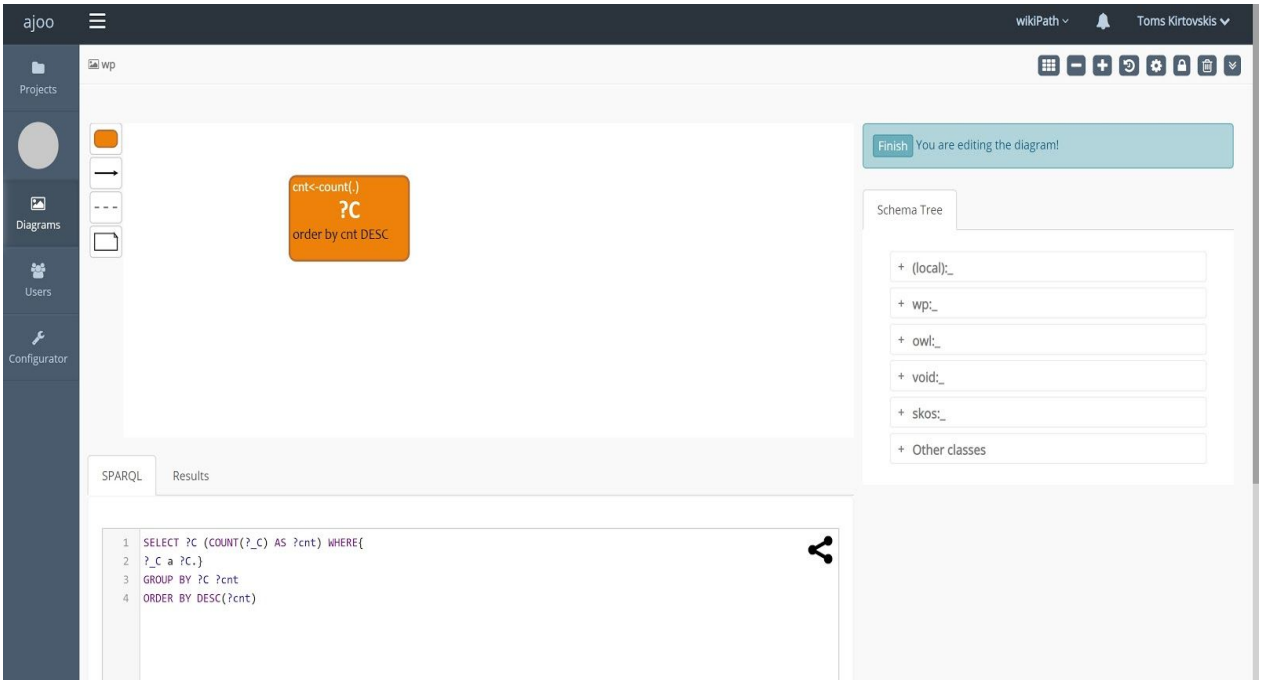
5.17. att. Visu klašu un to konceptu shēmu sarakstu attēlojošs vizuāls vaicājums.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX : <http://www.w3.org/2004/02/skos/core#>
SELECT ?Class ?ConceptScheme WHERE{
?Class a owl:Class.
?ConceptScheme a :ConceptScheme.
?Class :inScheme ?ConceptScheme.}
```

5.18. att. Visu klašu un to konceptu shēmu sarakstu attēlojošs attēlojošs SPARQL vaicājums.



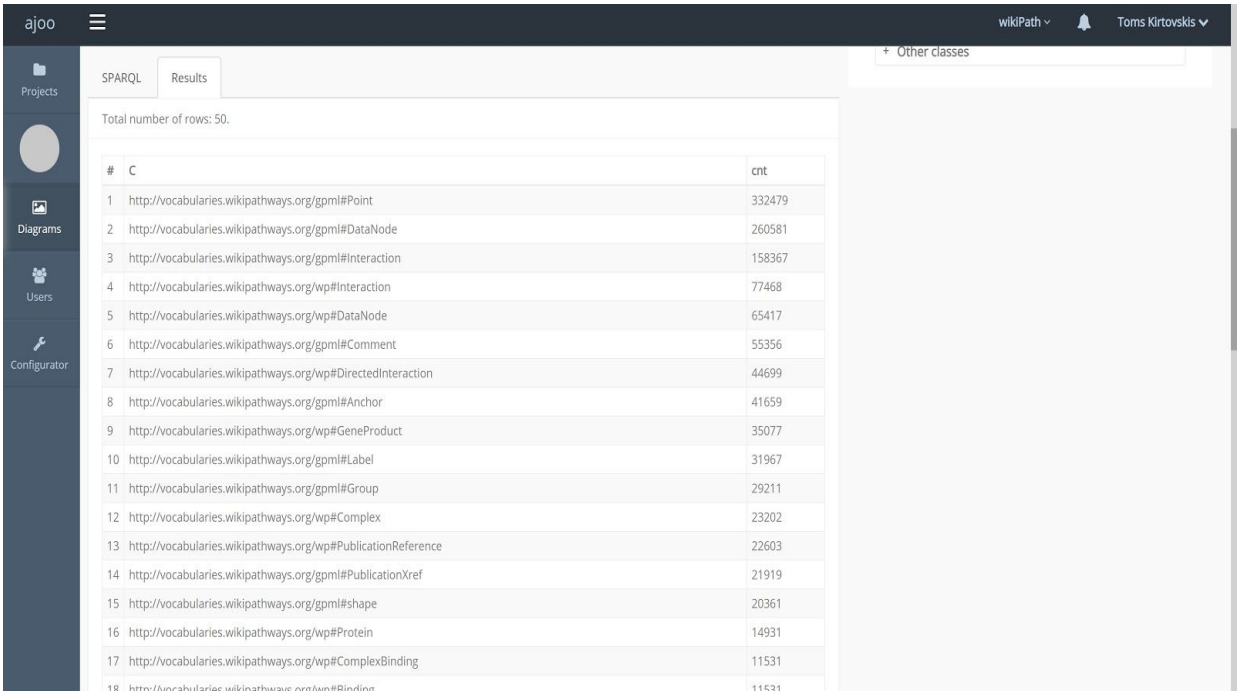
5.19. att. Visu klašu un to konceptu shēmu sarakstu attēlojošs vaicājuma rezultāts.



5.20. att. Vizuāls vaicājums, kas attēlo klašu uzskaitījumu un instanču skaitu tajās.

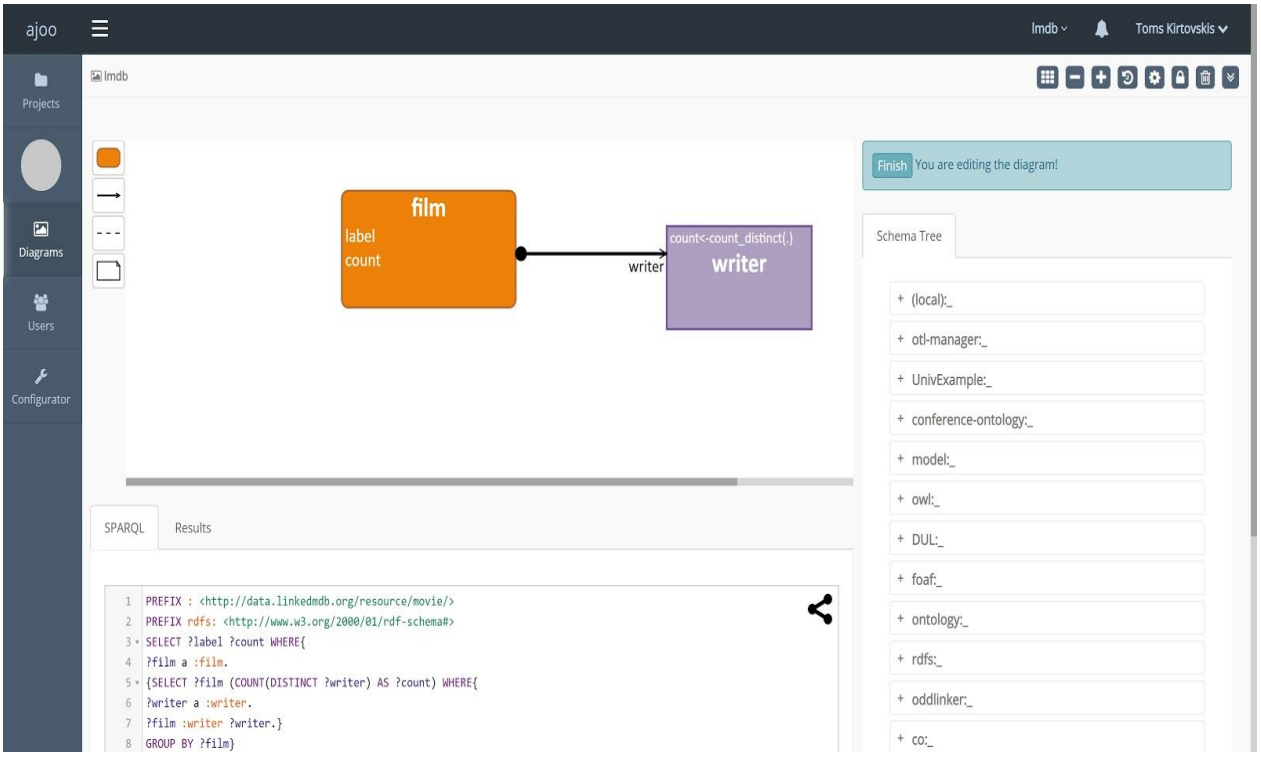
```
SELECT ?C (COUNT(?_C) AS ?cnt) WHERE{
?_C a ?C.}
GROUP BY ?C ?cnt
ORDER BY DESC(?cnt)
```

5.21. att. SPARQL vaicājums, kas attēlo klašu uzskaitījumu un instanču skaitu tajās.



5.22. att. Vaicājuma, kas attēlo klašu uzskaitījumu un instanču skaitu tajās, rezultāts.

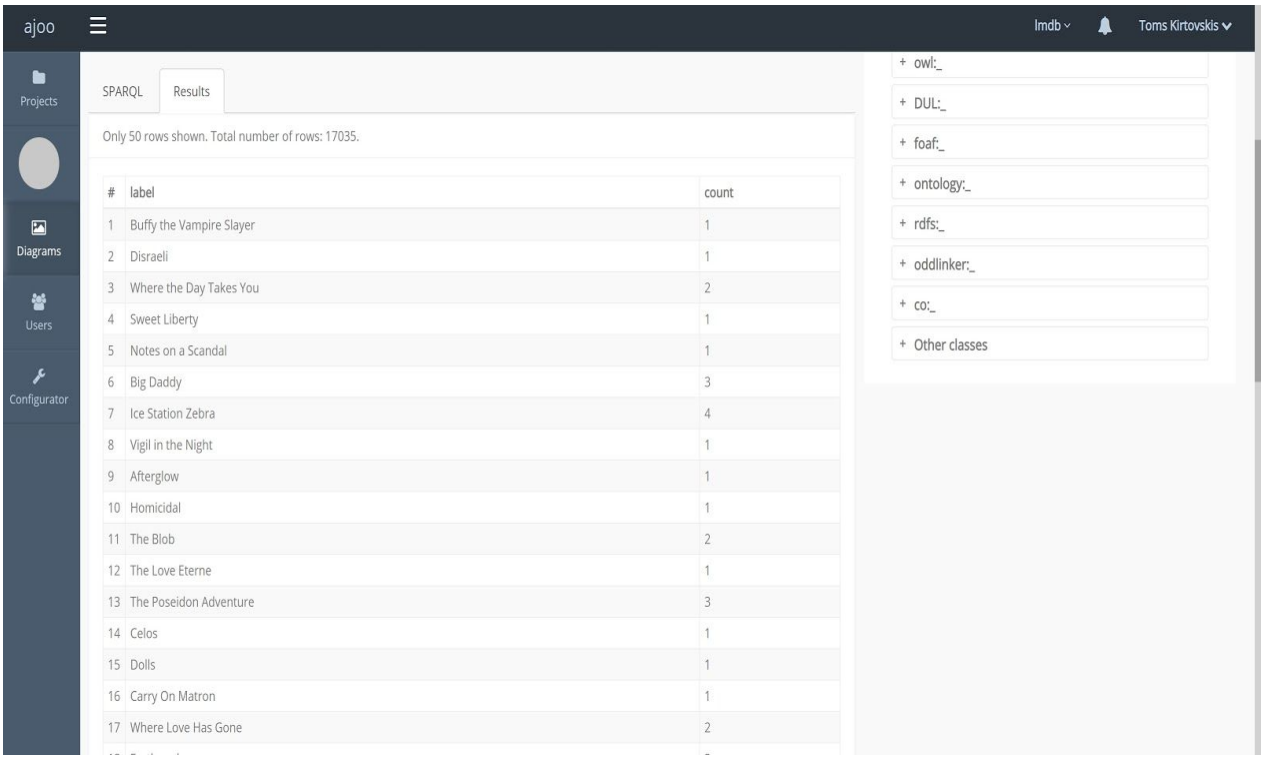
5.2.3. Vaicājumu piemēri izmantojot Linked Movie Database piekļuves punktu



5.23. att. Vizuāls vaicājums, kas attēlo visu piekļuves punktā esošo filmu sarakstu kopā ar katras filmas scenārija autoru skaitu.

```
PREFIX : <http://data.linkedmdb.org/resource/movie/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?label ?count WHERE{
?film a :film.
{SELECT ?film (COUNT(DISTINCT ?writer) AS ?count) WHERE{
?writer a :writer.
?film :writer ?writer.}
GROUP BY ?film}
OPTIONAL{?film rdfs:label ?label.}}
}
```

5.24. att. SPARQL vaicājums, kas attēlo visu piekļuves punktā esošo filmu sarakstu kopā ar katras filmas scenārija autoru skaitu.



5.25. att. Vaicājuma, kas attēlo visu piekļuves punktā esošo filmu sarakstu kopā ar katras filmas scenārija autoru skaitu, rezultāts

SECINĀJUMI

Veicot piekļuves punktu izpēti un aplūkojot to saderību ar ViziQuer rīku autors novēroja, ka piekļuves punkta saderību lielā mērā nosaka katra konkrētā piekļuves punkta klašu daudzums. Tātad jo vairāk klašu, jo grūtāk ir konkrēto piekļuves punktu pielietot kopā ar ViziQuer rīku. Lai šādu problēmu atrisinātu, būtu nepieciešams uzlabot ViziQuer rīka darbību, apstrādājot gadījumus ar lielu (100 un vairāk) klašu skaitu.

Sīkāk aplūkojot Eiropas Bioinformātikas institūta piedāvāto risinājumu, viegli pamanīt, ka tajā ir salīdzinoši liels skaits klašu, kas nozīmē, ka to šobrīd ir samērā sarežģīti lietot kopā ar ViziQuer rīku. Pie tam lielākā daļa Eiropas Bioinformātikas institūta platformas izstrādātāju piedāvāto SPARQL vaicājumu piemēru nav centrēti ap klasēm un tas neatbilst ViziQuer funkcionalitātes pamatidejai par ērtu vaicājumu veidošanu, jo vizuālo vaicājumu tips šādiem vaicājumiem nav piemērots.

Aplūkojot piekļuves punktu piemērus, kuriem nav iespējama sekmīga datu shēmas un/vai ontoloģijas iegūšana, ir sastopama sekojoša problēma - vaicājumus iespējams veikt tikai SPARQL formātā, t.i., ja nav pieejama SPARQL piekļuves punkta datu shēma, tad vizuālo vaicājumu izveide ir ļoti apgrūtināta, vai neiespējama.

Analizējot tos piekļuves punktus, kuru datu shēmas ir iespējams iegūt, izmantojot SPARQL datu shēmu spiegotāju, viegli pamanīt, ka ir piekļuves punkti, kuri neatbalsta visu vaicājumu izpildi, piemēram, SKOS piekļuves punkts (saite: http://skos.um.es/sparql/). Taču ir arī vairāki piekļuves punkti, ar kuriem iespejams pilnvērtīgi darboties izmantojot ViziQuer rīku. Tajā skaitā: pubmed.bio2rdf.org (saite:http://pubmed.bio2rdf.org/sparql), Europeana (saite:http://sparql.europeana.eu/), ScholarlyData (saite:http://www.scholarlydata.org/sparql/), wikiPathways (saite: http://sparql.wikipathways.org/), Social semantic web thesaurus(saite: http://vocabulary.semantic-web.at/PoolParty/sparql/semweb), Linked Movie Database (saite: http://85.254.199.72:8890/sparql)

IZMANTOTĀ LITERATŪRA UN AVOTI

[1] Linked Data, 2006. [tiešsaiste]. - [atsauce 28.12.2018.]. Pieejams: https://www.w3.org/DesignIssues/LinkedData.html

[2] linkeddata.org, [tiešsaiste]. - [atsauce 15.12.2018.]. Pieejams: http://linkeddata.org/faq

[3] www.w3.org. [tiešsaiste]. - [atsauce 05.01.2019.]. Pieejams: https://www.w3.org/standards/semanticweb/data

[4] Resource Description Framework. [tiešsaiste]. - [atsauce 20.12.2018.]. Pieejams: https://en.wikipedia.org/wiki/Resource_Description_Framework

[5] SPARQL. [tiešsaiste]. - [atsauce 15.01.2019.]. Pieejams: https://en.wikipedia.org/wiki/SPARQL

[6] ViziQuer: a Tool to Explore and Query SPARQL Endpoints. (vecāka rīka versija) [tiešsaiste]. - [atsauce 15.01.2019.]. Pieejams: https://pdfs.semanticscholar.org/f987/c85716db71e6fc249480508f08cde57c76d0.pdf

[7] SPARQL 1.1 Overview, 2013 [tiešsaiste]. - [atsauce 22.01.2019.]. Pieejams: https://www.w3.org/TR/sparql11-query/

[8] SPARQL 1.1 Query Language, 2013 [tiešsaiste]. - [atsauce 22.01.2019.]. Pieejams: https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/

[9] ViziQuer/web Tool for RDF Data Analysis Queries (jaunā tīmeklī bāzētā versija) [tiešsaiste]. - [atsauce 22.01.2019.]. Pieejams: http://viziquer.lumii.lv/

[10] Hypertext Transfer Protocol -- HTTP/1.1 [tiešsaiste]. - [atsauce 22.04.2019.]. Pieejams: http://www.ietf.org/rfc/rfc2616.txt

[11] Uniform Resource Identifier (URI): Generic Syntax [tiešsaiste]. - [atsauce 24.04.2019.]. Pieejams: https://www.ietf.org/rfc/rfc3986.txt

[12] UTF-8, a transformation format of ISO 10646 [tiešsaiste]. - [atsauce 25.04.2019.]. Pieejams: https://tools.ietf.org/html/rfc3629

[13] SPARQL 1.1 Protocol [tiešsaiste]. - [atsauce 25.04.2019.]. Pieejams: https://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/

[14] Percent-encoding [tiešsaiste]. - [atsauce 25.04.2019.]. Pieejams: https://en.wikipedia.org/wiki/Percent-encoding

[15] Schema Extractor for Visual Query Tool [tiešsaiste]. - [atsauce 25.04.2019.]. Pieejams: http://viziquer.lumii.lv/schema-extractor/index.html

[16] ViziQuer Git repository [tiešsaiste]. - [atsauce 25.04.2019.]. Pieejams: https://github.com/LUMII-Syslab/viziquer
[17] The Linked Open Data Cloud [tiešsaiste]. - [atsauce 25.04.2019.]. Pieejams: https://lod-cloud.net/

PIELIKUMI

1. pielikums Koda paraugs

```
VQ_sparql_logs = new Mongo.Collection("VQ_Exec_SPARQL_Logs");
Meteor.methods({
executeSparql: function(list) {
var user_id = Meteor.userId();
if (is_project_member(user_id, list)) {
   console.log("in exec sparql");
   var options = list.options;
   var limit_set = false;
   var number_of_rows = 0;
   var sparql_log_entry = {};
   _.extend(sparql_log_entry, list);
   _.extend(sparql_log_entry, {user:user_id});
   var newDate = new Date();
   _.extend(sparql_log_entry, {date:newDate.toLocaleDateString(),
time:newDate.toLocaleTimeString()});
   if (!options.paging_info) {
      // let's try to determine the number of rows in the result
      try {
         // clone object. It is an efficient hack
         var count_options = JSON.parse(JSON.stringify(options));
         // inserting SELECT COUNT before the first occurence of SELECT
         count_options.params.params.query =
buildEnhancedQuery(count_options.params.params.query, "SELECT", " SELECT
(COUNT(*) as ?number_of_rows_in_query_xyz) WHERE { ", "}");
         count_options.params.params.format="application/sparql-results+json";
///////////////Autora rekomendācija//////////////////
         // to modify endpoint by adding URL encoded querry
         let query = count_options.params.params.query;
         let namedGraph = count_options.params.params['default-graph-uri'];
```

```
        query = query.replace(/(\r\n|\n|\r)/gm," ");
        query = encodeURIComponent(query);
        query = query.replace(/[*]/g, '%2A');
        query = query.replace(/[(]/g, '%28');
        query = query.replace(/[)]/g, '%29');
        count_options.endPoint = count_options.endPoint + '?'+ 'default-graph-uri=' +
namedGraph +'&query=' + query + '&format=JSON';
///////////////Autora rekomendācija//////////////////
        var qres = HTTP.post(count_options.endPoint);
        if (qres.statusCode == 200) {
        var content = JSON.parse(qres.content);
        number_of_rows = content.results.bindings[0].number_of_rows_in_query_xyz.value;
        _.extend(sparql_log_entry, {successfull:true});
        if (number_of_rows > 50) {
        options.params.params.query = buildEnhancedQuery(options.params.params.query,
"SELECT", "SELECT * WHERE {", "} LIMIT 50");
        limit_set = true;}}
      }
      catch (ex) {
      // ERROR - pass the original SPARQL to the server
      _.extend(sparql_log_entry, {successfull:false, error_message:ex})
      console.error(ex);
      };
   } else {
      if (!options.paging_info.download) {
      options.params.params.query = buildEnhancedQuery(options.params.params.query,
"SELECT", "SELECT * WHERE {", "} OFFSET "+options.paging_info.offset+" LIMIT
"+options.paging_info.limit);
      limit_set = true;
      number_of_rows = options.paging_info.number_of_rows;
      } else {
      // Do not change query
      // Since no refresh is intended = no additional parameters required  } }
```

```
    _.extend(sparql_log_entry, {number_of_rows:number_of_rows});
    add_sparql_log(sparql_log_entry);
    Future = Npm.require('fibers/future');
    var future = new Future();
    try {
///////////////Autora rekomendācija//////////////////
        // to modify endpoint by adding URL encoded querry
        let query = options.params.params.query;
        let namedGraph = options.params.params['default-graph-uri'];
        query = query.replace(/(\r\n|\n|\r)/gm," ");
        query = encodeURIComponent(query);
        query = query.replace(/(\*)/g, '%2A');
        query = query.replace(/[(]/g, '%28');
        query = query.replace(/[)]/g, '%29');
        options.endPoint = options.endPoint + '?'+ 'default-graph-uri=' + namedGraph
+'&query=' + query;
///////////////Autora rekomendācija//////////////////
        HTTP.call("POST", options.endPoint, function(err, resp) {
        if (err) {
            future.return({status: 505, error:err, limit_set: false, number_of_rows: 0});
        } else {
            xml2js.parseString(resp.content, function(json_err, json_res) {
                if (json_err) {
                    future.return({status: 504, error:json_err, limit_set: false, number_of_rows: 0});
                } else {
                    if (limit_set) {
                        if (options.paging_info) {
                                _.extend(json_res, {limit: 50, offset:options.paging_info.offset + 50});
                        }
                        else { _.extend(json_res, {limit: 50, offset: 50}); }
                    }
                _.extend(json_res, {limit_set: limit_set, number_of_rows: number_of_rows});
                    future.return({status: 200, result: json_res});
```

```
            }});}});
    } catch (ex) {
        future.return({status: 503, ex:ex, limit_set: false, number_of_rows: 0});
    };
    return future.wait();}
},
testProjectEndPoint: function(list) {
    var user_id = Meteor.userId();
    if (is_project_member(user_id, list)) {
        if (!list.endpoint && !list.uri) {
            console.error("No data specified");
            return {status: 500,};        }
        Future = Npm.require('fibers/future');
        var future = new Future();
        var params = {};
////////////////Autora rekomendācija//////////////////
        let query = 'SELECT ?a ?b ?c where{?a ?b ?c} LIMIT 10'
        query = encodeURIComponent(query);
        query = query.replace(/(\*)/g, '%2A');
        query = query.replace(/[(]/g, '%28');
        query = query.replace(/[)]/g, '%29');
        list.endpoint = list.endpoint + '?query=' + query;
////////////////Autora rekomendācija//////////////////
        HTTP.call("POST", list.endpoint, params, function(err, resp) {
        if (err) {future.return({status: 500,});
        } else {
            xml2js.parseString(resp.content, function(json_err, json_res) {
            if (json_err) { future.return({status: 500,});
            }  else {future.return({status: 200,});
            } });}});
        return future.wait();}},
});
```

Dokumentārās lapas forma

Maģistra darbs “ Praktiski vizuāli datu vaicājumi” izstrādāts LU Datorikas fakultātē.

Darba teksta galīgā versija izgatavota 2019. gada 19. maijā.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____

(Autora paraksts un datums)

Ar savu parakstu apliecinu, ka esmu lasījis augstāk minēto maģistra darbu un atzīstu to par p i e m ē r o t u / n e p i e m ē r o t u (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu maģistrantūrā.

Darba vadītājs: _____

(Vadītāja paraksts un datums)

Darbs iesniegts maģistratūras sekretariātā _____.

(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.

Studiju metodiķe: _____.

(Metodiķes paraksts)

Recenzents: _____

(Akad.amats, zin.grāds, vārds, uzvārds)

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____

(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____

(Sekretāra paraksts)