LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

LIETOTĀJA SASKARNES RISINĀJUMI VIZUĀLĀ VAICĀJUMU RĪKĀ

MAĢISTRA DARBS

Autors: **Jūlija Hodakovska**

Studenta apliecības Nr.: jh18008

Darba vadītājs: profesors Dr. dat. Kārlis Čerāns

RĪGA 2020

Anotācija

Mūsdienās arvien biežāk ir nepieciešams atlasīt informāciju, tajā skaitā arī tādu, kas glabājas strukturētā veidā – datu bāzēs. Tāpēc ir nepieciešams izmantot vaicājumvalodas, kas ļauj definēt atlasīšanas kritērijus, bet pārsvarā tādas valodas nav domātas iesācējiem. Ar šo problēmu cīnās tajā skaitā arī vizuālie vaicājumu rīki, kas nodrošina iespēju grafiski attēlot savas prasības vienkāršotā veidā, un pēc tam bez lietotāja palīdzības "pārtulko" attēlu par vaicājumu.

Darbā ir veikta esošo vizuālo vaicājumu rīku analīze, pievēršot uzmanību lietotāja saskarnes risinājumiem, un piedāvāti risinājumi vienam no tiem – LU MII izstrādātajam ViziQuer rīkam. Izvēlētie risinājumi ir izstrādāti, implementēti un izmantoti ViziQuer rīka publiski izplatāmā versijā.

Atslēgvārdi: SPARQL, vaicājumu rīki, vizuālie vaicājumi

Abstract

Title: **USER INTERFACE SOLUTIONS FOR VISUAL QUERY TOOL**

Nowadays it is important to select some part of the information, including structured information from data bases. This makes it a necessity to use query language to define the selection criteria, but mostly these languages are not friendly for casual user. One of the methods to solve the situation is visual query tools, that allow graphically represent requirements in a simplified way, and then automatically "translate" drawn symbols to textual query.

In this work analysis of existing visual query tools was made, paying more attention to user interface solutions, and some solutions were proposed for one of them – ViziQuer tool that has been developed in the Institute of Mathematics and Computer Science, University of Latvia. Proposed solutions were developed, implemented and now are part of the publicly available ViziQuer tool version.

Keywords: SPARQL, query tool, visual query

Autoreferāts

Mūsdienu pasaulē palielinās informācijas skaits, tajā skaitā arī tādas, kurai ir pievienoti metadati, kas atbilst RDF prasībām, un tas ļauj izmantot SPARQL vaicājumvalodu lai atlasīt lietotājam nepieciešamo, definējot nosacījumus un informācijas daudzumu. Tas, savukārt, palielina pieprasījumu pēc lietotājam draudzīgiem rīkiem, kas spēj "pārtulkot" prasības uz SPARQL vaicājumvalodas vaicājumiem.

Darba autore izstrādājot šo darbu ir veikusi 18 vizuālo vaicājumu rīku analīzi, balstoties uz pieejamas informācijas, galvenokārt, zinātniskajiem rakstiem un rīkam veltītajiem tiešsaistes resursiem, no kuriem 5 rīki tika izpētīti arī no praktiskā viedokļa, apskatot tiešsaistē pieejamo versiju un izveidojot vaicājumus ar šo rīku palīdzību. Skaitļi ir doti neieskaitot LU MII izstrādāto ViziQuer rīku, tātad 18 apskatītie rīki nav saistīti ar darba autores iepriekšējiem vai esošiem izstrādes darbiem.

Darba autore ir sākusi savu darbu pie ViziQuer vizuālā vaicājumu rīka 2016. gadā, šajā laikā izstrādājot divus kursa darbus, kvalifikācijas darbu, bakalaura darbu un maģistra darbu par vizuālajiem vaicājumu rīkiem. Darba ietvaros ir apskatīti trīs pamata un divi papildus lietotāja saskarnes funkcionalitātes bloki (Add Link, Aggregate Wizard un Connect Classes, kā arī papildus Add Outer Query un Add Union), kas ir izstrādāti un implementēti ViziQuer rīka publiski izplatāmā versijā. Risinājumu izstrāde ir veikta patstāvīgi, konsultējoties ar ViziQuer rīka izstrādē iesaistītiem cilvēkiem par rīka esošajām vai topošajām funkcionalitātēm, kuru izstrādē darba autore nebija iesaistīta, vai izvēloties kādu no iespējamajiem risinājumiem un to atbilstību rīka kopējai koncepcijai. Risinājumu apjoms no koda lieluma skatu punkta ir aptuveni 2 500 rindiņas.

Gatavojot izstrādāto funkcionalitāšu novērtēšanas testēšanu, kas iekļauj sevī vaicājumu un lietotāju aptaujas jautājumu definēšanu, darba autore izpētīja jau esošus līdzīgus novērtēšanas testus un veica savu vaicājumu uzdevumu izstrādi, pamatojot to izmantošanu ar atbilstošu testēšanas mērķi. Visi vaicājumi ir novērtēti veicot vaicājumu definēšanai nepieciešamo darbību analīzi, pierādot funkcionalitāšu efektivitāti.

Darbā ir atsauces uz dažādiem avotiem kopskaitā 81, no kuriem 36 ir zinātniskie raksti, ieskaitot divus rakstus, kuru tapšanā piedalījās arī darba autore. Darba noformējums atbilst maģistra darba izstrādes un aizstāvēšanas metodiskajiem norādījumiem. Darbā ir izmantota oficiāli pieņemtā nozares terminoloģija.

SATURS

APZĪMĒJUMU SARAKSTS

ASCII (no angļ. American Standard Code for Information Interchange) - Septiņu bitu kods rakstzīmju kopas elementu identificēšanai. Kods sākotnēji izstrādāts kā ASV nacionālais standarts, bet vēlāk tas kļuvis arī par starptautisku standartu. [1]

IBM – ASV firma, kas ražo datorus un programmatūru [1]

IoT (no angļu *Internet of Things*) – lietu internets [1]

IRI (no angl. Internationalized Resource Identifier) – paplašināts resursu identifikators – tīmekļa protokola standarts, kas balstās uz URI, bet atbalsta plašāku simbolu skaitu (arī ne-ASCII simbolus) [2]

IT – informācijas tehnoloģijas

JSON (no angļu *JavaScript Object Notation*) – datu apmaiņas formāts, kas strukturētus datus apraksta tekstuālā formā [3]

LU MII – Latvijas Universitātes Matemātikas un Informātikas Institūts

OWL (no angļ. Web Ontology Language) – semantiska iezīmēšanas valoda, kas paredzēta ontoloģiju publicēšanai un koplietošanai tīmeklī [4]

QBE (no angļu *Query by Example*) – vizuāla vaicājumvaloda relācijas datu bāzēm

RDF (no angļu *Resource Description Framework*) – resursu aprakstīšanas ietvars, W3C uzturētas specifikācijas informācijas konceptuālajiem aprakstiem vai modelēšanai, kuru izmanto tīmekļa resursos; ir daļa no asociatīvās sēmas koncepcijas

SPARQL (rekursīvs saīsinājums no angļu *SPARQL Protocol and RDF Query Language*) – no 2008. gada ir W3C rekomendēta RDF vaicājumvaloda

SQL (no angļ. Structured Query Language) - Firmas IBM izstrādāta valoda, ko lieto datu bāzes pārvaldības sistēmās dažāda tipa datoros [1]

URI (no angļu *Uniform Resource Identifier*) – vienotais resursu identifikators vai universālais resursu identifikators – kods dokumenta atsevišķas kopijas adreses vai cita resursa identificēšanai internetā [5]

Vaicājums (angļu *query*) – lietotāja vēršanās pie datu bāzes, lai iegūtu nepieciešamo informāciju [6]

W3C (no angļu *World Wide Web Consortium*) – galvenā starptautiska organizācija, kas nodarbojas ar standartiem priekš globālā tīmekļa

IEVADS

Attīstoties tehnoloģijām un palielinoties pieejamo datu apjomam ir parādījusies nepieciešamība arvien biežāk atlasīt noteiktu informāciju. Ja informācija glabājās datu bāzēs, tad par rīku kļūst vaicājumvalodas, kas ļauj definēt atlasīšanas kritērijus izmantojot stingri definētas konstrukcijas, kuras ir vieglāk pārveidot par izpildāmo programmu. Neskatoties uz attīstītām tādu valodu iespējām, tas pārsvarā nav domātas iesācējiem, bet cilvēkiem ar izglītību IT jomā un specializāciju atbilstošajā nozarē (piemēram, SQL vai SPARQL vaicājumvalodas pielietojumos), ierobežojot cilvēku-lietotāju loku. Ar šo problēmu cīnās tajā skaitā arī vizuālie vaicājumu rīki, kas nodrošina iespēju grafiski attēlot savas prasības vienkāršotā veidā, salīdzinot ar tekstuāliem vaicājumiem, un pēc tam "pārtulko" attēlu par vaicājumu. Tas uzlabo informācijas pieejamību arī cilvēkiem, kuriem nav atbilstošas izglītības, ļaujot iegūt datus no tādiem avotiem, kuriem ir pieejami meklēšanas rīki un atbilstoša ontoloģija, piemēram, meklēšana pār Wikidata datu bāzes ierakstiem, izmantojot SPARQL vaicājumvalodu [7].

Katra rīka mērķis ir nodrošināt lietotājam maksimāli pozitīvo pieredzi tā izmantošanas laikā, kam kalpo arī saskarnes dizains un pieejama funkcionalitāte, tajā skaitā arī biežāk izmantoto iespēju ātrāka izsaukšana. Rūpīgi izstrādāta un labi pārdomāta saskarne piešķir rīkam papildus vērtību un sekmē to pielietošanu, nodrošinot saprotamu darbību secību, kas noved pie pareiza uzdevuma risinājuma. Slikti pārdomāta vai no daudziem līmeņiem sastāvoša saskarne izraisa nepieciešamību meklēt papildus informāciju par rīka iespējām, tērējot tam laiku, vai apgrūtina pareizo darbību secības iegaumēšanu, un tādejādi izraisa negatīvas emocijas un samazina rīka subjektīvo novērtējumu.

Darbā ir veikta esošās situācijas vizuālo vaicājumu jomā analīze: kādi rīki tika izstrādāti, veiksmes stāsti, tajā skaitā atsevišķu problēmu sekmīgi risinājumi, un pētījumos atrastie ierobežojumi un nepilnības. Identificētas ViziQuer rīka, kas ir izstrādāts un attīstīts Latvijas Universitātes Matemātikas un Informātikas institūtā (LU MII), esošās lietojamības problēmas un piedāvāti potenciāli risinājumi. Izvēlētie risinājumi ir implementēti, un ir novērtēta lietotāju pieredze ar mērķi noteikt lietojamības uzlabošanas pakāpi, kā arī novērtēt ar tiem saistītus rīka tālākas attīstības virzienus lai nodrošināt lietotājiem saprotamu un ērtu rīka izmantošanu. Darba rezultāti tika publicēti starptautisko konferenču rakstu krājumos [8, 9].

Darba mērķi un uzdevumi

Šī darba mērķis ir izpētīt saskarnes risinājumus vizuālo vaicājumu jomā un piedāvāt dažus risinājumus ViziQuer rīkam. Atbilstoši mērķiem tika uzstādīti sekojoši uzdevumi:

- Izpētīt esošu situāciju vizuālo vaicājumu rīku jomā, nosakot pēdējo gadu tendences un aktuālos risinājumus;
- Noteikt ViziQuer rīka saskarnes risinājumu esošo stāvokli, apskatīt attīstības dinamiku, ka arī nepieciešamas izmaiņas, lai nodrošināt lietotāju ar saprotamām un ērtām vaicājuma definēšanas metodēm;
- Implementēt izvēlētos saskarnes risinājumus tādā līmeni, kas ļauj pievienot tos ViziQuer rīka publiski izplatāmai versijai.

Darba struktūra

Darbs sastāv no 5 nodaļām un 8 apakšnodaļām. Nodaļā "Ievads" ir īsi raksturota apskatāma problēma un definēti darba mērķis un uzdevumi. Nodaļā "Vizuāli vaicājumu rīki" ir veikts rīku vēsturiskais pārskats, raksturoti uz diagrammām balstītie rīki no lietotāja saskarnes risinājumu skatu punkta un apskatīti tipiskie piemēri rīku novērtēšanas testi. Nodaļā "ViziQuer vizuālais vaicājumu rīks" ir veikta atbilstošā rīka analīze un izstrādāto risinājumu pārskats un novērtējums, ka arī minēti rīka tālākas attīstības virzieni. Nodaļā "Rezultāti" ir pārskaitīti sasniegumi, kas atspoguļo darba paveikto. Nodaļā "Secinājumi" ir nosaukti secinājumi, kurus darba autore izdarīja izstrādājot šo darbu.

Darbs ir uzrakstīts uz 74 lapām, tas satur 29 attēlus un 6 tabulas.

1. VIZUĀLI VAICĀJUMU RĪKI

Mūsdienu pasaulē arvien lielāku lomu spēlē informācija: jau no senajiem laikiem ir zināma tās vērtība, ko labi raksturo Rotšildu dinastijas pamatlicēja slavenā frāze "Kam pieder informācija, tam pieder pasaule" (*Who owns the information, he owns the world*) [10]. Tomēr arī citu pazīstamo cilvēku novērojumi ir aktuāli. Einšteins ir atzinis, ka informācija vēl nav zināšanas (*Information is not knowledge*) [11], bet *Firefox* pārlūka autors M.Kapors (*Mitchell Kapor*) tēlaini salīdzināja informācijas ieguvi no tīmekļa ar mēģinājumu padzert no ugunsdzēsības hidranta (*Getting information off the Internet is like taking a drink from a fire hydrant*) [12]. Apvienojot šo divu autoru teikto var apgalvot, ka mūsdienas ir pieejams milzīgs informācijas apjoms, lietderīgi izmantot kuru nav triviāls uzdevums. Ja parasto meklētāju programmētājs spēj izveidot par 100 US$ (aptuvena atbilstoša mācību kursa cena), tad tādu milžu, kā Google kompānija, produkta analoga izstrāde maksātu jau 100 miljonus US$ [13].

Daļa no tīmeklī pieejamajiem datiem glabājas strukturētā veidā: daļēji kā datu bāžu sastāvdaļa, bet daļēji kā lietu interneta (*Internet of Things*, IoT) koncepcijas realizācija. Starp šiem jēdzieniem ir aptuveni 50 gadu starpība: pirmās datu bāzes paradījās 1960.-os gados, bet IoT koncepcija tika izveidota 2000.-os gados, kad T.Bērnerss-Lī (*Tim Berners-Lee*) izklāstīja savu redzējumu par saistīto informāciju (*Linked Data*) [14]. Abus variantus apvieno iespēja izmantot atbilstošu vaicājumvalodu un ar to saistītās problēmas, kuras izraisa lietotāju sagatavotības nepieciešamība, lai tie spētu iegūt sev nepieciešamo informāciju.

Uz saistītās informācijas koncepta izveidotais semantiskais tīmeklis (*Semantic Web*) kļūst aizvien populārāks, jo ļauj atrast datus izmantojot metadatus, t.i. piešķirot datiem papildus vērtību, kategorizējot tos un veidojot iespēju noteikt attiecības starp objektiem. Jo tas ir izdarīts datoriem "saprotama" veidā, tad pastāv rīki, kas spēj atlasīt nepieciešamus objektus, kuri atbilst izvirzītajiem kritērijiem. Par meklēšanas rīku kļuva SPARQL vaicājumvaloda, kas ir W3C Resursu aprakstīšanas ietvaram (*Resource Description Framework*, RDF) rekomendēta vaicājumvaloda, bet tā nav domāta lietotājam, kas nav IT specialisti, un tiek uzskatīta par lietotājam nedraudzīgo [15,16]. Lai atrisināt šo situāciju tika piedāvāti vairāki rīki, kas vienā vai otrajā veidā piedāvā izveidot vaicājumu, neizmantojot vaicājumvalodas konstrukcijas.

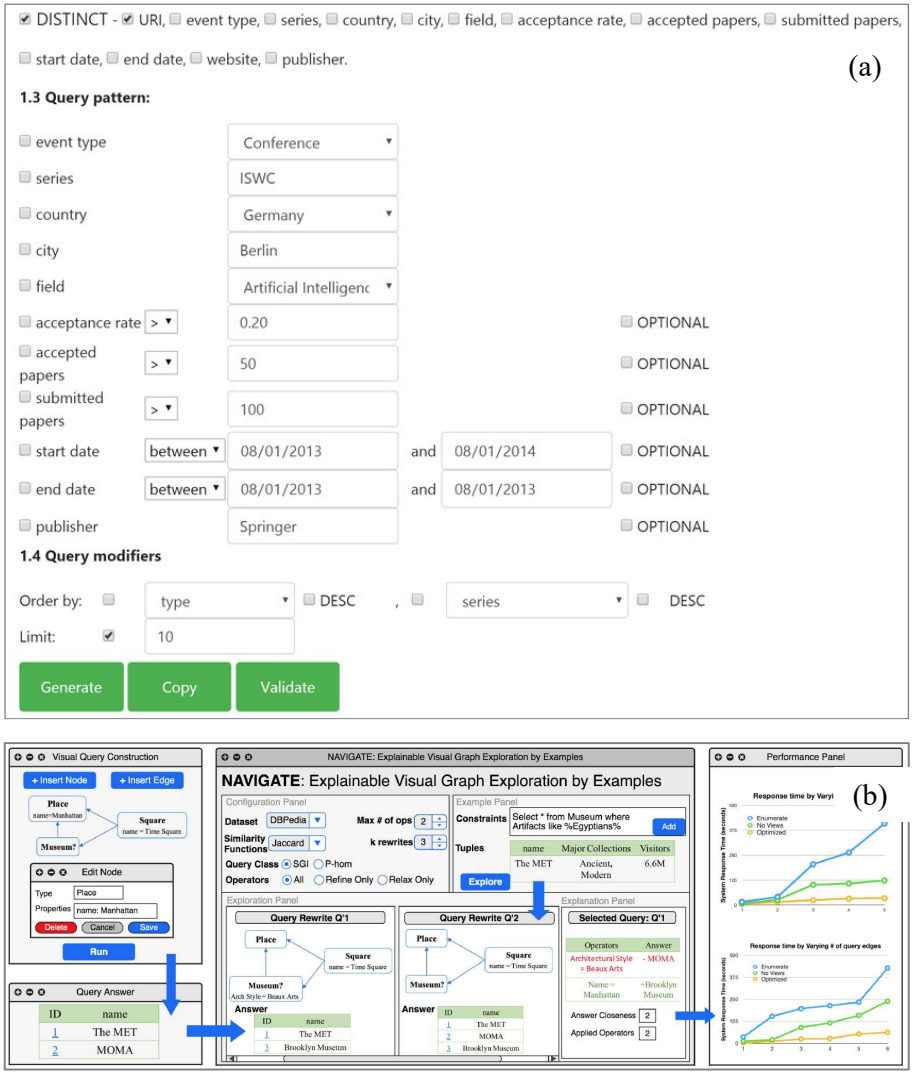
1.1. Vizuālo vaicājumu rīku vēsture

Datu bāzēm domāto vizuālo vaicājumu rīku vēsture ir sākusies aptuveni tajā pat laikā, kad tika aktīvi attīstītas SQL datu bāzes, par pirmo tiek uzskatīts IBM kompānijas 1970.-os gados izstrādātais QBE (no angļ. *Query by example*) rīks [17], kas ir filtru sistēma, kura ļauj lietotājam ievadīt savas ierobežojošās vērtības un atlasīt atbilstošus ierakstus no relācijas datu bāzes [18]. Ja rīks ir domāts iepriekš zināmajiem datu bāzēm, tad šī pieeja ir izplatīta, tomēr galvenais trūkums ir nepieciešamība definēt visu lauku filtrus un iespējamas saites un atribūtu vērtības, kas ne-relācijas datu bāzēm ir grūts uzdevums, jo datu struktūrā var definēt ļoti daudz laukus, un ar vērtību būs tikai daži ieraksti. Tāpēc no uz formām balstītajiem rīkiem pētījumi novirzījās uz uz diagrammām balstītajiem, kā arī uz tādiem rīkiem, kas apstrādā cilvēku valodu, jeb dabisku valodu, un pārveido to par vaicājumu atbilstoši kādas vaicājumvalodas prasībām. Tomēr noteiktiem pielietojumiem ar iepriekš zināmo datu struktūru arī ne-relācijas datiem ir izveidoti uz formām balstītie rīki, piemēram, SPARQLViz rīks, kas veido vaicājumu aizpildot formas ar paskaidrojumiem cilvēkiem saprotamā valodā [19], M.Arenasa (*Marcelo Arenas*) grupas izstrādātais formas tipa rīks [20] vai jaunāki SPARQL-AG rīks [21] (1.1.att.(a)) un vairāk grafu izpētei domāts Navigate rīks, bet kuru funkcionalitātē ir arī vaicājumu formulēšana [22] (1.1.att. (b)).

Uz diagrammām balstīto rīku vēsture ir īsāka, rīki ir izstrādāti jau šajā – XXI – gadsimtā, ko nosaka pieprasījums pēc tāda veida risinājumiem un grafiskās vides izstrādes iespējas. Daži rīki ir izstrādāti 2006.-2007. gados, piemēram, iSPARQL rīks [23] un Semantic Crystal rīks [24]. Pēc izstrādes laika nākamais rīks ir Nitelight rīks, kas ir izstrādāts 2008. gadā kā tīmekļa lietotne [25], 2010. gadā tika publicēts darbs par vēl vienu rīku – SPARQLinG un atbilstošu RDF-GL vaicājumvalodu [26], bet 2011. gadā – Smeagol rīks [27]. Bet ņemot vērā, ka vēlāk tika izstrādāts cits rīks ar SPARQLinG rīkam līdzīgo nosaukumu, tad tālāk šis rīks tiks saukts pēc vaicājumvalodas.

Vizuālo vaicājumu rīku izstrāde intensīvi notika pēdējos 10 gados, daži no tiem ir specializētie rīki, kas ir domāti vienam pielietojumam, piemēram, AskOmics rīks priekš ar genomiem saistītās informācijas meklēšanas no speciālās datu bāzes [28], kas pēc ilga pārtraukuma pagājušajā gadā ir guvis pārveidoto versiju un tagad aktīvi attīstās [29]. Cits līdzīgs arī pēc pielietojuma nozares (medicīnas dati) rīks ir SPARQLGraph rīks, bet pēdējā šī rīka

versija ir izveidota 2014. gadā [30], tomēr 2018. gadā šo rīku savā prezentācijā apskatīja Boeing kompānijas pārstāvji, analizējot to potenciālu saviem pielietojumiem [31], kas liecina par rīka kvalitāti.



1.1.att. **Uz formām balstītie rīku saskarnes:**
(a) SPARQL-AG rīks [21], (b) Navigate rīks [22]

Citi rīki ir domāti patvaļīgo ontoloģiju apstrādei, tomēr daudzi no tiem pēdējā laikā nav mainīti, kas var liecināt par to, ka projekts vismaz kādu laiku nav attīstīts vai notiek nopietni labojumi, kas vēl nav pievienoti rīka versijai. Piemēram, pietiekami neparasts idejas līmeni ir SparqlBlocks rīks, kas atgādina Lego konstruktoru, rīkā katrai vaicājumvalodas konstrukcijai ir sava forma, un pēc savienošanas vietām var noteikt, kur tai jābūt. Bet pēdējās publikācijas par rīku paradījās 2017. gadā [32], tomēr github platformā izmaiņas projektā bijā arī pēdējā pusgada laikā, bet tās bija saistītas ar izmantoto programmatūru atjauninājumiem [33], kas

liecina par to, ka rīks nav pilnīgi pamests, un, iespējams, līdzīgi AskOmics rīkam tiks attīstīts tālāk. Tajā pašā laikā tika izstrādāts Visual SPARQL Builder rīks, kas pēdējo reizi tika atjaunināts 2015. gadā [34], bet šis rīks arī tika minēts Boeing kompānijas prezentācija [31]. Vēl viens uz diagrammām balstīts rīks ir GraphTQL grafiskās vaicājumvalodas realizācija, kas paradījās 2015. gadā [35], arī tai nav novērota attīstība [36].

Līdzīgā situācijā ir QueryVOWL rīks, bet interesanti pieminēt, ka daļa no to izstrādātājiem iepriekš darbojas arī ar SparqlFilterFlow rīku [37], un tālākie darbi ir saistīti ar OWL ontoloģiju vizualizācijas rīku [38]. Iespējams, QueryVOWL rīks tiks tālāk izmantots kā daļa no cita projekta, pievienojot tā iespējas citam rīkam.

Viens no veiksmīgākajiem vizuālo vaicājumu rīkiem, kas domāti SPARQL vaicājumvalodai, ir OptiqueVQS (*Optique Visual Query System*, tāpēc izmanto arī saīsinātu nosaukumu - Optique) rīks, kas ir izstrādāts Eiropas komisijas FP7 integrētā projekta ietvaros (*European Commision 7th Framework Programme Integrated Project*). Ja iepriekšminētie rīki parasti ir saistīti ar zinātnisko institūciju, tad Optique rīka izstrāde piedalījās ne tikai sešas universitātes, bet arī četri uzņēmumi, ieskaitot labi pazīstamu *Siemens AG* un *Statoil ASA* uzņēmumus [39]. Aktivitātes saistība ar projektu ir notikušas līdz 2018. gadam, t.i. 2 gadus pēc projekta beigām [40], pašu OptiqueVQS rīku turpina attīstīt [41], ka arī ir izveidota vienkāršāka rīka versija ar nosaukumu kg-interface [42]. Daļa no zinātniekiem, kas bija iesaistīti Optique rīka izstrādē, turpināja tēmu ar cita rīka izstrādi, kas balstoties uz OWL2 ontoloģijām veido navigācijas grafus (*navigation graphs*) lai tālāk izmantot tos vieglākai vaicājumu formulēšanai [43].

Pēdējos gados ir izveidots vēl daži rīki, piemēram, 2018. gadā SPARQLing rīks, kas izmanto ontoloģijas attēlošanas valodu GRAPHOL lai izveidot vaicājumu, [44] tas pēdējā laikā nav atjaunināts [45], un RDF Explorer rīks, spriežot pēc github platformas datiem, rīku aktīvi attīstīja 2018.-2019. gados, bet tagad to izstrādē ir pauze [46]. Autori savā rakstā izcēla rīku par to, ka tam ir automātiskā teksta pabeigšana (*autocomplete*), tas spēj dinamiski reaģēt uz izmaiņām vaicājumā, neļauj izveidot vaicājumus, kuriem nav rezultātu ar vērtībām (*non-empty results*), un ir pieejama uz paraugiem balstītā lietotāju apmācība, un tāds īpašību komplekts lielākai vaicājumu rīku daļai nav [47]. Par pašu salīdzinājumu vairāk tiks stāstīts 1.3. apakšnodaļā, bet salīdzinājuma atbilstība esošais situācijai ViziQuer rīkam tiks analizēta 2.1. un 2.2. apakšnodaļās.

Apskatot rīku vēsturi kopumā var secināt, ka mūsdienas nav pieejams rīks, kas tiek uzskatīts par etalonu. Daudzi rīki ir zinātnieku mēģinājumi atrast ceļu, kas ved pie informācijas pieejamības jautājumu atrisināšanas, bet nav komerciāli vai brīvi pieejami potenciālajiem

lietotājiem pat ja lietotāju loks ir noteikts kā "bez iepriekšējām zināšanām". Pagaidām lielāka aktivitāte šajā nozarē bija ap 2015.-2018. gadiem, un šo novērojumu apliecina arī Zaifera (*Philipp Seifer*) ar līdzautoriem raksts, kurā ir izpētīts github platformas aktivitātes vizuālo vaicājumu jomā vairākām vaicājumvalodām, ierobežojoties ar publiski pieejamajiem Java programmēšanas valodā rakstītajiem risinājumiem, kuru rezultāti rāda, ka lokālais maksimums aktivitātēs tika sasniegts 2016.-2018. gados [48].

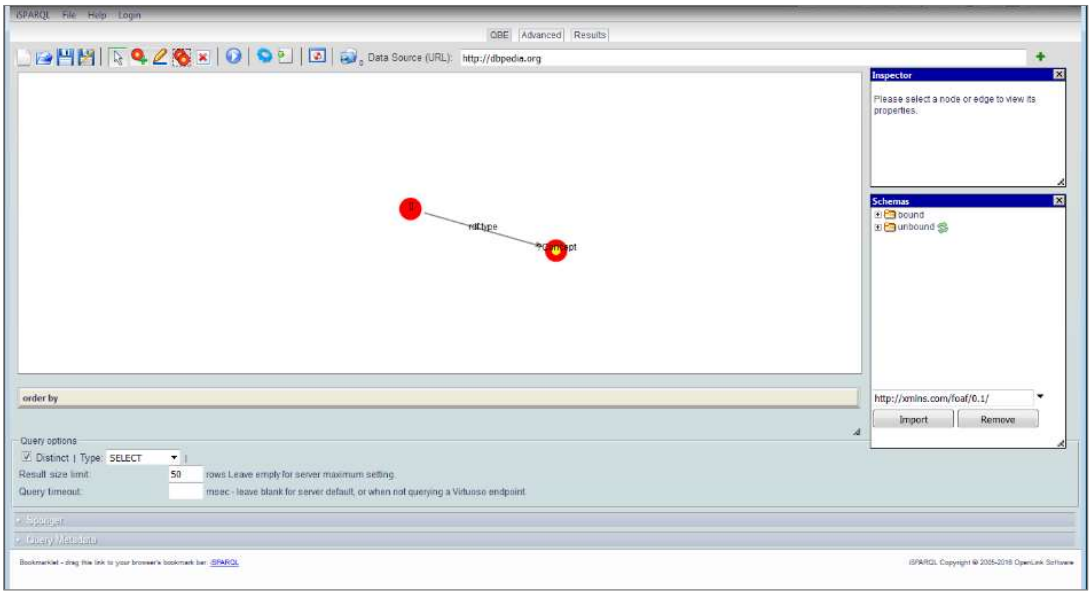
Tomēr mainoties situācijai un palielinoties pieprasījumam pēc ne-tekstuāla rīka [49], tiks izstrādāti jaunie risinājumi vai modificēti vecie, lai atbilstu esošai situācijai. Par pieprasījuma esamību arī no datu turētājiem liecina vairāku vietņu pievienota iespēja veikt meklēšanu izmantojot SPARQL vaicājumvalodu, piemēra, jau minētais Wikidata piemērs [7], ka arī Eiropas datu portāls [50] un tādi projekti kā Europeana projekts [51].

1.2. Uz diagrammām balstīto rīku lietotāja saskarnes risinājumu pārskats

Lai novērtēt esošos risinājumus un noteikt stiprās un vājās vietas, kurus būtu nepieciešams uzlabot, tiek veikti lietotāju testi, kad rīkus izmanto kāda potenciālo lietotāju grupa, veicot uzdevumus un novērtējot savu pieredzi. Mūsdienās nav definēti parametri, pēc kuriem novērtēt rīku, kā arī nav vaicājumu komplektu rīku novērtēšanai, tāpēc izstrādātāji izvēlās tādus vaicājumus, kas ļauj novērtēt viņu interesējošo īpašību realizāciju. Viens no bieži lietojamiem parametriem ir lietojamība (*usability*): "sistēmas īpašība, kas raksturo, cik viegli lietotājs var apgūt tās izmantošanu, sagatavot tai ieejas datus un interpretēt tās izejas datus" [52]. Šai definīcijai zinātnieki pievieno dažus raksturojošus parametrus, piemēram, vizuālo elementu pievilcību (*attractiveness*) [53], citi sadala to smalkās daļās, piemēram, Čens (*Yu-Hui Chen*) salīdzinot divas grupas ir ieviesis 11 novērtēšanas parametrus. Apskatot tik smalku sadalījumu rodas jautājums, kā noteikt robežu, jo, piemēram, apgūstamība (*Learnability*) ir cieši saistīta ar saskarnes dizainu, un šī nenoteiktība parādās Čena (*Yu-Hui Chen*) darbā, jo viena grupa (bibliotekāri) augsti novērtēja saskarni un dizainu, ievietojot 4. vietā, bet no zinātniskām publikācijām šo parametru ievietoja tikai 7. vietā [54].

Šajā darbā tiek uzskatīts, ka saskarnes un rīka kopējais dizains ir svarīgs punkts, kas ietekmē lietotāja pieredzi, rīka apgūstamību, spēju atcerēties pareizu darbību secību, lai sasniegt kādu mērķi, un lietotāja darba produktivitāti, veicot datu meklēšanas uzdevumus, tāpēc tika veikta esošo rīku saskarnes risinājumu analīze pēc publikācijām un palaižot rīka demonstrācijas versiju, ja tā bija iespējama. Šī informācija tiks tālāk izmantota analizējot ViziQuer rīka esošo stāvokli un iespējamus rīka attīstības virzienus.

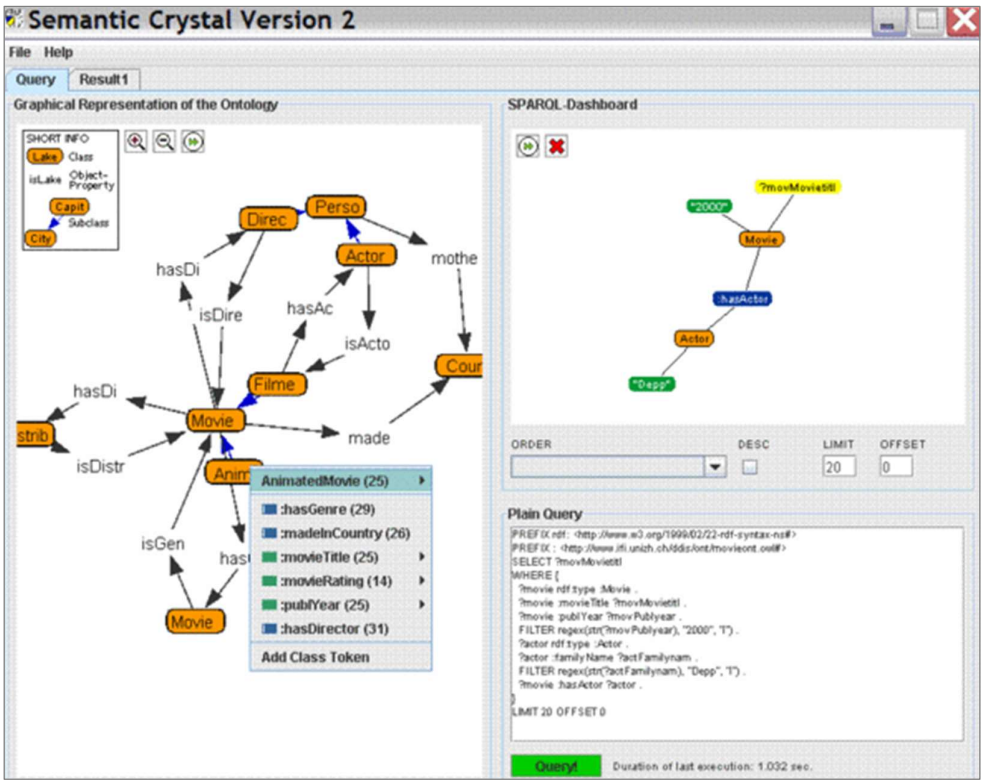
Tālāk netiek analizēti risinājumi tādos specializētajos rīkos kā AskOmics un SPARQLGraph rīki, jo tie darbojas ar iepriekš definēto īpatnējo – medicīnas – datu kopu un nav paredzēti lietošanai ar citiem datiem, un SPARQLBlocks rīks tā risinājuma idejas īpatnību dēļ. Arī Smeagol un GraphTQL realizācijas rīki netika analizēts, jo no rakstos piedāvātajiem materiāliem nav skaidri to darbības principi no saskarnes skatu punkta, bet demonstrēšanai paredzētās lapas nav pieejamas. Pārējiem rīkiem tiks veikta īsā saskarnes elementu analīze, balstoties uz literatūras avotiem un demonstrēšanai izveidotajām lapām, ja tādas ir pieejamas. Jo ne visiem rīkiem pastāv iespēja pamēģināt to darbībā, tad šī analīze nav izsmeļoša un var nesaturēt daļu no informācijas.



1.2.att. **iSPARQL rīka saskarne [23]**

Pirmajiem rīkiem – iSPARQL rīks, kas pieejams tiešsaistē [23], un Semantic Crystal rīks – ir raksturīgs vienkāršs dizains gan no grafisko risinājumu viedokļa, gan arī no pieejamo funkcionalitāšu klāsta. iSPARQL rīks ir vienkāršāks (1.2.att.), tam nav informācijas par pieejamām klasēm un saitēm, objektu attēlojumi ir ne līdz galam pārdomāti, skatoties no mūsdienu lietotāja viedokļa, piemēram, nav iespējams palielināt klases objektu, lai tajā varētu ieiet nosaukums. Pētot rīka funkcionalitāti tika pamanīti daži nepārdomāti risinājumi, piemēram, poga elementu pievienošanai darbojās uz katru peles klikšķi uz zīmēšanas lauka līdz brīdim, kad ir ieslēgta cita funkcija. Dzēšanas gadījumā ir nepieciešams apstiprināt savu darbību. Daudzi lauki ir aizpildāmi ar roku, kas pat zinot ontoloģiju un objektu vārdus ir kļūdu avots, jo netika pamanīta nekāda sarakstīto simbolu pārbaude.

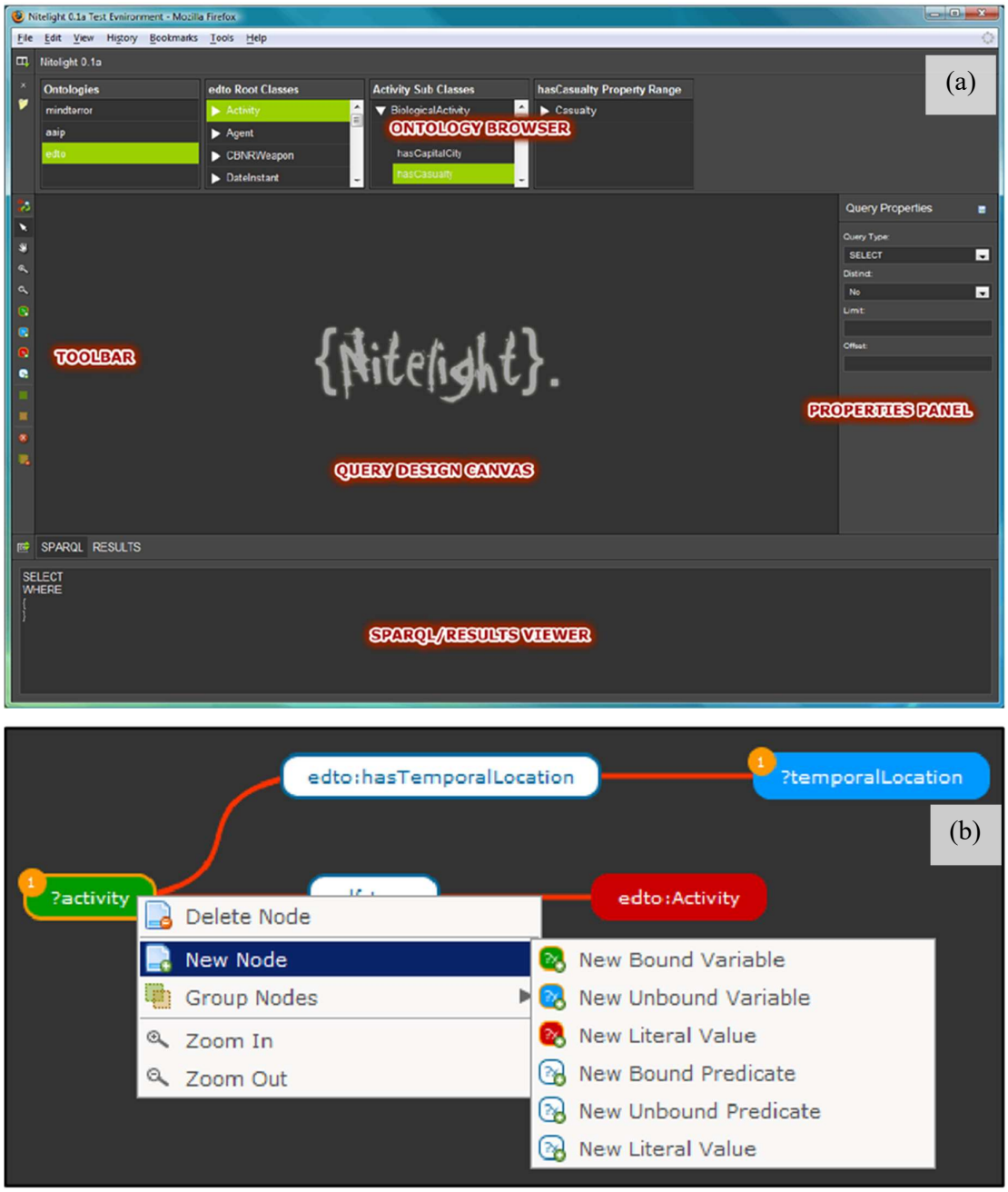
Semantic Crystal rīkam ir vairāk informācijas (1.3.att.), kas ir prezentēta grafiski: izmantojot krāsas ir izceltas klases, saites un atribūtu vērtības; gara nosaukuma jautājumu atrisina vienkāršākajā veidā, ierobežojot simbolu skaitu. Pats rīks ir sadalīts vairākās daļās, ieskaitot rezultātu cilni un laukus vaicājuma grafiskai un tekstuālai formai. Rīkam ir arī grafiski attēlota ontoloģija, par ko tas ir pelnījis nozares ekspertu un citu lietotāju atzinību vairākos pētījumos, jo nelielām ontoloģijām tāda vizualizācija atviegloja vaicājuma definēšanu [55, 56]. Semantic Crystal rīkam ir īsinājumizvēlne, kas, spriežot pēc attēla, ļauj noteikt klasei atbilstošus saišu un atribūtu vārdus, pie tam norādīts atbilstošu objektu skaits.



1.3.att. **Semantic Crystal rīka saskarne [24]**

Nitelight rīkam arī ir lakonisks dizains, kas ar krāsu palīdzību izceļ dažus elementus – izvēlēto objektu vai pašu vaicājumu (1.4.att.), tas pēc izstrādātāju domām sastāv no 5 daļām:

- Ontoloģiju daļa (*Ontology browser*), kurā ir pieejama ontoloģijas (lietotājam ir iespēja pievienot tos), klases (*Root class*), saites (*Activity sub class*) un atribūtu (*Property range*) izvēlei domāti saraksti;
- Vaicājuma veidošanas logs (*Query design canvas*);
- Rīkjosla (*Toolbar*), kurā atrodami vaicājuma grafisko elementu ievietošanas un pārveidošanas rīki;



1.4.att. **Nitelight rīka saskarne [25]:**

(a) kopskats, (b) vaicājuma veidošanas process ar īsinājumizvēlni

- Īpašību panelis (*Property panel*), kurā ir iespējams modificēt izvēlēto elementu vai, ja neviens elements nav izvēlēts, visu vaicājumu parametru, piemēram, ierobežojot atlasīto elementu skaitu;
- Vaicājuma vai rezultātu skatītājs (*SPARQL/Results viewer*) (1.4.att.(a)).

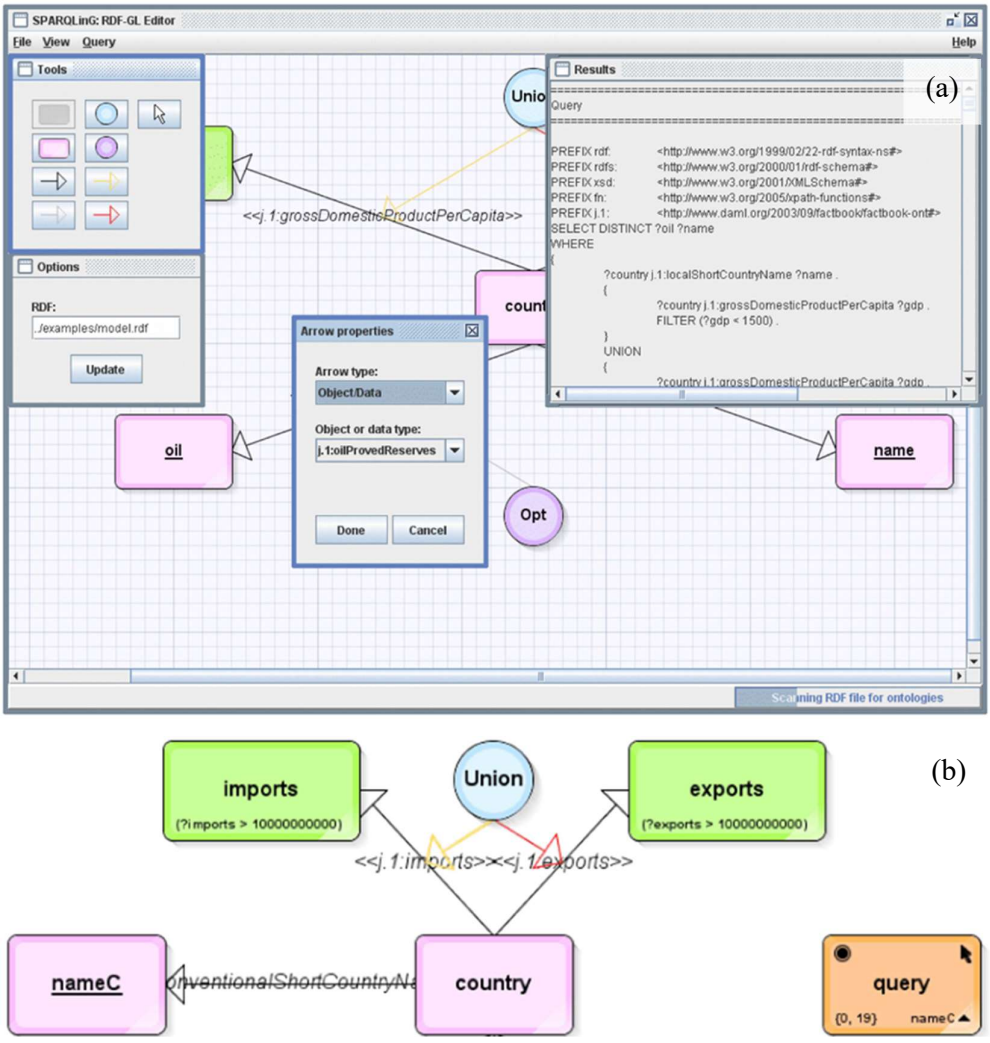
Rakstā autori piemin, ka ontoloģijas daļā izvēloties klasi paradās jauns lauks ar dotas klases apakšklasēm kā vēl viens saraksts; iespējams, šī īpašība deva nosaukumu "saknes klase", apzīmējot virsklases; pamanītas arī citas atšķirības terminoloģijā. Nav izdevies atrast informāciju, vai ir iespējams uzreiz atrast klasi, ja tā ir apakšklase, vai ir nepieciešams sākt ar virsklasi. Otrajā gadījumā vajadzīgas klases meklēšana var būt saistīta ar vairākiem soļiem, kas apgrūtina vaicājuma sastādīšanu un prasa ontoloģijas pārzināšanu.

Rīkam ir īsinājumizvēlne (1.4.att.(b)), kas pēc apraksta nodrošina iespējas pievienot elementus un modificēt tos, izmantojot tādas iespējas kā filtrēšana, kārtošana un grupēšana, daļēji atkārtojot rīkjoslas funkcijas. Vēl viena lietotāja saskarnes funkcionalitāte ir iespēja pārnest objektu no ontoloģijas daļas uz vaicājuma logu pēc vilkšana un nomešana principa (*drag-and-drop*). Spriežot pēc īpašību loga apraksta, vaicājuma veidošanas logs var saturēt tikai vienu vaicājumu [25].

RDF-GL rīka saskarne (1.5.att.(a)) atgādina attēlu apstrādei domātas programmas, jo līdzīgas iespējas ir apvienotas viena atsevišķajā logā: objektu zīmēšanas logs (*Tools*), objekta īpašību logs (*Arrow properties*) un t.t. Spriežot pēc attēla, logi novietoti virs diagrammas, un pie neliela rīka loga izmēra aizņem lielāko daļu no zīmēšanai paredzētas vietas, kas lielām diagrammām var izraisīt salasāmības problēmas. Iespējams, tos var samazināt vai citādi paslēpt, tomēr pieejamajā aprakstā par to nav informācijas.

Rīks piedāvā interesantu saites īpašības definīciju (1.5.att.(b)): apvienošana (*Union*) un neobligātas vērtības (*Opt,* no angļ. *optional*) īpašības ir atsevišķi diagrammas objekti, kas ar bultām savienoti ar pašu objektu vai objektiem. Tomēr ir apšaubāmi, ka tāda pieeja atvieglo lietotāja izpratni par darbībām, jo palielinās objektu skaits vizuālajā vaicājumā. Objektu novietojums un izmērs ir maināmi, izņemot bultas, kurām ir uzlikts ierobežojums uz sākuma un beigu elementu esamību un veikta automātiska izmēra pielāgošana. Objektu īpašību panelis paradās izvēloties kādu objektu, ļaujot, pēc rīka izstrādātāju domām, intuitīvi uzstādīt vajadzīgas īpašības; to izskats bultu gadījumā ļauj apgalvot, ka vērtības ir nolasītas no ontoloģijas un tos nav nepieciešams ievadīt ar roku [26]. RDF-GL rīks ir hronoloģiski pirmais no apskatītajiem, kas ievieto nosacījumus objekta attēlojumā, tātad nosacījums ir pamanāms uz

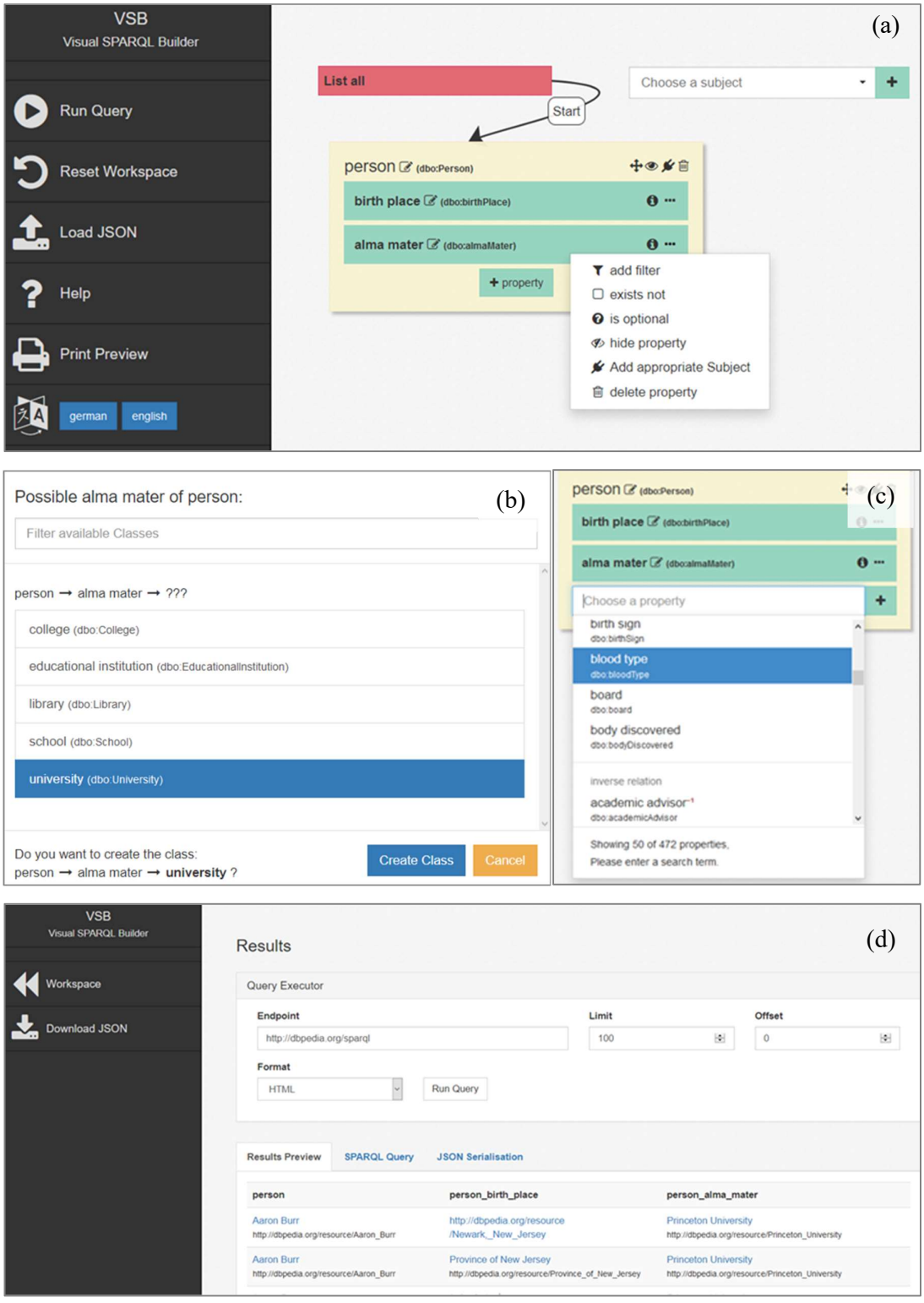
diagrammas, bet nav atdalāms no objekta (atribūta). Cits interesants saskarnes risinājums ir vaicājuma parametru iznešana speciālajā objektā (*query*), tajā atrodas informācija par atlasāmo ierakstu skaitu un kārtošanas nosacījumiem (1.5.att.(b)).



1.5.att. **RDF-GL rīks[26]:**

(a) logs, (b) vaicājuma piemēra daļa

Visual SPARQL Builder rīks ir vēsturiski nākamais un pirmais no apskatītajiem rīkiem, kas ievietoja atribūtus klases grafiskajā objektā (1.6.att.(a)). Kreisajā loga pusē atrodas rīka funkcijas, kas ļauj veikt darbības, kas nav tieši saistītas ar vaicājuma zīmēšanu, piemēram, palaist vaicājumu un atcelt izmaiņas. Apakšā (nav uz attēla) ir poga, kas ļauj paslēpt tekstuālo izvēlnes daļu un samazināt to līdz rīkjoslai, atbrīvojot vairāk vietas vaicājuma veidošanai. Vaicājuma daļā ir divi objekti: vaicājums un klases pievienošanas rīks. Vaicājuma sākuma



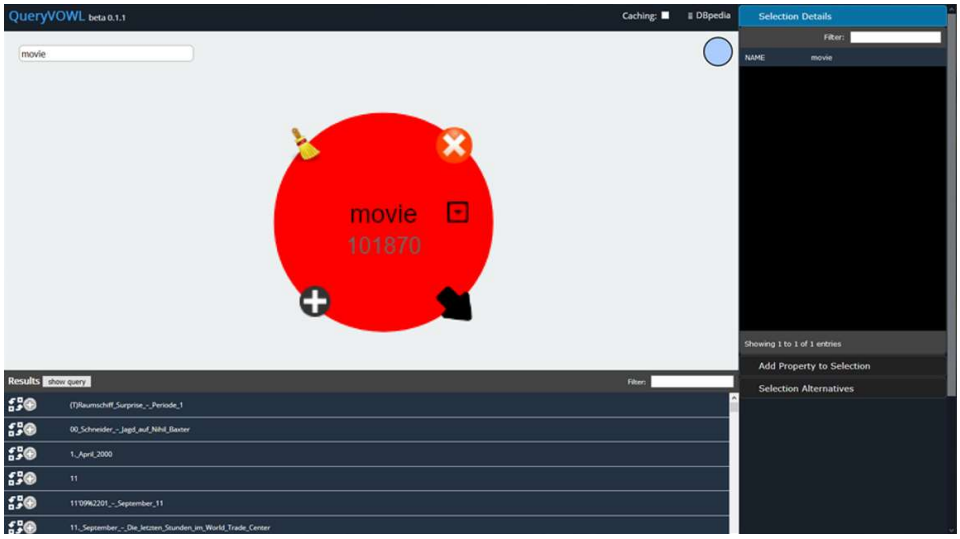
1.6.att. **Visual SPARQL Builder rīka saskarne [57]:**

(a) ar ieslēgto alma mater atribūta konfigurēšanas izvēlni, (b) Add appropriate Subject funkcionalitātes logs, (c) +property pogas funkcionalitāte, (d) un vaicājuma rezultātu logs

elements (sarkans taisnstūris) demonstrēšanas versijā ir ievietots no sākuma un nav maināms, kas neļauj novērtēt to funkcionalitāti.

Klases pievienošana var notikt divos veidos: (1) no labajā augšējā stūrī novietota nolaižamā kombinētā lodziņa ar iespēju ievadīt tekstu, lai atlasīt to saturošus elementus; (2) no klases objekta atribūtu loga ar Add appropriate Subject "dakšas" palīdzību ("dakša" no klases loga ļauj savienot diagrammā jau ievietotas klases). Otrā iespēja ir pieejama, ja atribūtam ir atbilstošais objekts; ja atribūts ir ar fiksēto – skaitlisko, tekstuālo vai citu – vērtību, tad ir iespējams izvēlēties formātu, piemēram, datumu vai simbolu virkni. Atbilstošā objekta gadījumā pievienošanas poga atver jaunu logu (1.6.att.(b)), kurā ir meklēšana, RDF trijnieka divas daļas un iespējamas trešās daļas vērtības, kā arī pogas darbības apstiprināšanai un atcelšanai. Atribūtu pievienošanai klases objektā atrodas speciāla "+ property" poga, kas atver nolaižamo kombinēto lodziņu ar meklēšanas iespēju pēc ievadīta teksta un pogu izvēles apstiprināšanai ("+" poga) (1.6.att.(c)).

Vaicājuma rezultāti aizvieto vaicājuma definēšanas logu (1.6.att.(d)), tikai šajā brīdī ir iespējams mainīt pieslēgšanas galapunktu (*Endpoint*), vaicājuma parametrus, kas saistīti ar rezultātu attēlošanu (rezultātu skaitu un nobīde), ka arī apskatīties rezultātus un pašu vaicājumu tekstuālā formā. Kreisajā pusē ir pieejams rīks rezultātu lejupielādei JSON formātā un atgriešana pie vaicājuma veidošanas.

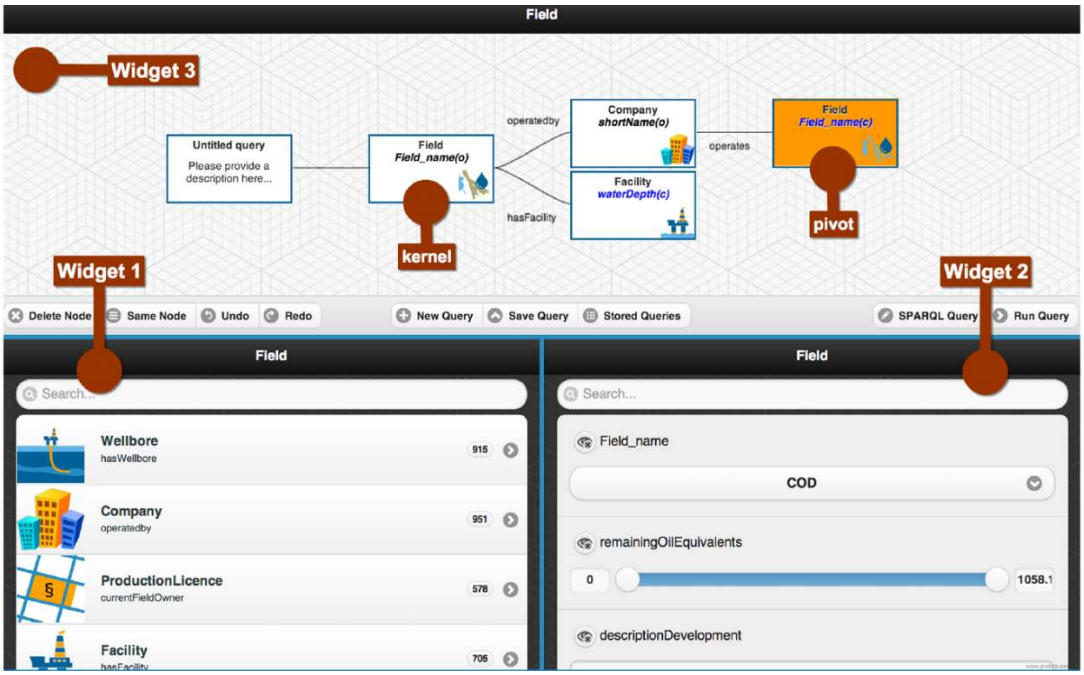


1.7.att. **QueryVOWL rīks [59]**

QueryVOWL rīks atverot to lapu ir pārāk lakonisks, gandrīz nav paskaidrojoša teksta, kas apgrūtina to apguvi, pēc tam, kad diagrammā ir ievietoti objekti, logi aizpildās ar informāciju (1.7.att.). Saskarne sastāv no meklēšanas loga, kas ļauj pievienot klasi, klases zīmēšanas rīka

(zils aplis), vērā ņemama daļa ir atvēlēta objekta informācijai (pa labi) un rezultātiem (apakšā). Rīka darbības īpatnības tika detalizēti apskatīti iepriekš [56], jo rīkā netika veiktas izmaiņas, tad ir vērts pieminēt tikai dažas no tām. QueryVOWL rīkam ir raksturīga pietiekami tehniska informācija URI formātā, īpatnēja meklēšanas rezultātu prioritātes noteikšana, kad rezultātus ne vienmēr iegūst uzreiz. Tas brīžiem strādā lēni, jo ir realizēta dinamiskā rezultātu pārrēķināšana pēc katras izmaiņas.

No interesantajām realizācijām ir vērts pieminēt to, ka dažas funkcijas vaicājuma būvēšanai ir realizētas kā interaktīvi elementi uz klases objekta; lai to būtu vieglāk pamanīt, objekts ir ievietots attēla palielinātajā izmērā ar attēlu apstrādes rīku palīdzību. Cits interesants risinājums ir rezultātu skaita ievietošana elementā, kas ļauj atrast to vietu vaicājumā, kas noveda pie tā, ka ir tukšs rezultāts, tomēr šis lauks ir viens no lēnas darbības cēloņiem, tāpēc nav viennozīmīgi novērtējams kā labs.



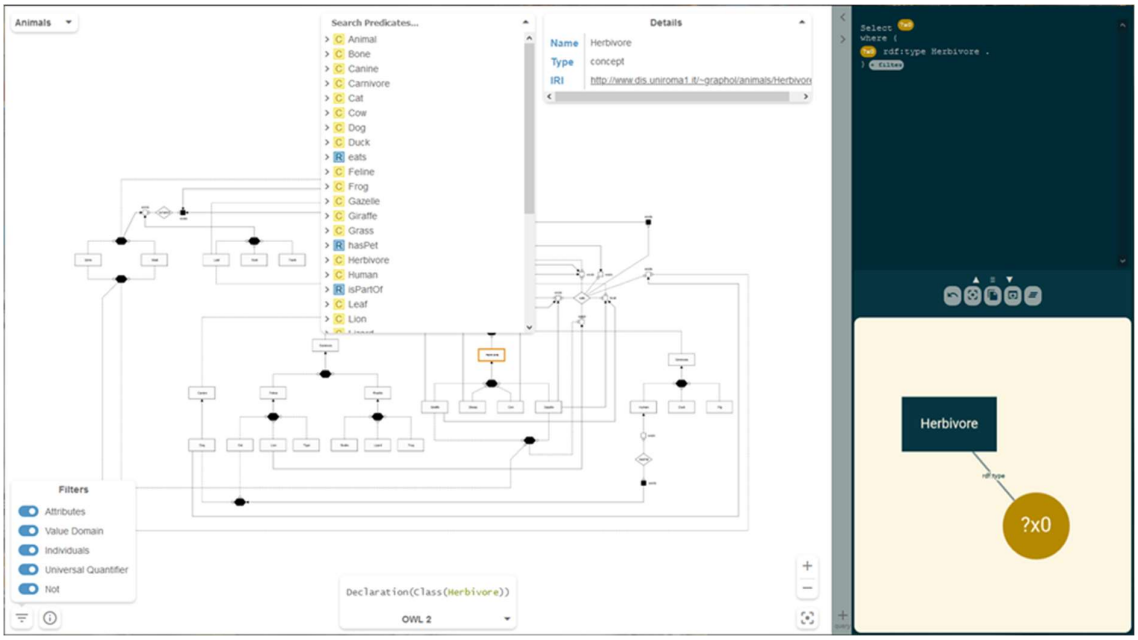
1.8.att. **OptiqueVQS rīka saskarne [60]**

Vēl viens rīks, kas arī iepriekš detalizēti apskatīts un izanalizēts [56], ir OptiqueVQS rīks (1.8.att.). Šī darba ietvaros ir vērts pieminēt dažas no tā saskarnes risinājumiem:

- Lietotājs vienlaikus darbojas tikai ar vienu vaicājumu;
- Ir pieejams klašu saraksts ar meklēšanu;
- Izvēloties klasi var izstādīt atribūtu parametrus: uzlikt nosacījumus uz vērtībām un noteikt, vai atribūts jāizvada rezultātos;

- Atribūtus var meklēt;
- Pēc pieprasījuma (atbilstošas pogas nospiešanas) ir pieejami
  - vaicājums tekstuālā formā;
  - vaicājuma rezultāti;
  - vaicājuma veidošanas funkcijas (jauns vaicājums, saglabāt un apskatīties saglabātus vaicājumus);
  - vaicājuma elementu rediģēšanas funkcijas, piemēram, atsaukt (*Undo*) vai atcelt atsaukšanu (*Redo*).

Novietojot vairākas biežāk izmantotas funkcijas rīka izstrādātājiem izdevās nepārslogot logu un saglabāt uzsvaru uz vaicājuma veidošanai paredzētajiem elementiem, salīdzinot, piemēram, ar RDF-GL rīku. Optique rīkam ir viens interesants risinājums, kas ļauj lietotājiem vaicājumu tekstuālā formā uzklikšķināt uz objekta un pāriet uz diagrammas formu ar fokusu uz izvēlētā objekta, kas atvieglo rediģēšanu. No rīka trūkumiem vai nepilnībām var minēt tādu faktu, ka diagrammā atribūta attēlošana vaicājuma rezultātos un nosacījums uz šo atribūtu atšķiras tikai ar krāsu un vienu burtu iekavās, t.i. nosacījuma formulējums nav redzams diagrammā.



1.9.att. **SPARQLing rīka saskarne [61]**

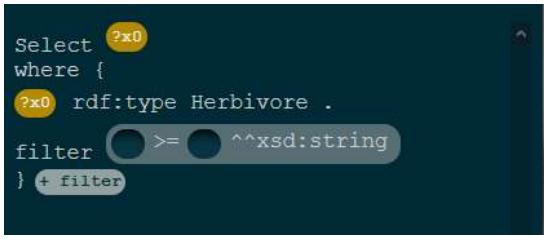
SPARQLing rīks (1.9.att.) ir pieejams tiešsaistē, tā autori piedāvā dažas pazīstamas demonstrējumiem paredzētas vienkāršas ontoloģijas, piemēram, *Pizza* un *Animals* [61]. Rīks sastāv no 3 daļām: (1) ontoloģijas attēlošanai paredzēta kreisā daļa; (2) šaurā pelēka vertikāla josla (tālāk – rīkjosla); (3) un labajā daļā atrodams vaicājums tekstuālā (augšā) un grafiskajā

(apakšā) formā. Ar "<" un ">" pogām rīkjoslā vaicājuma daļu var paslēpt pilnībā vai palielināt līdz 75% no ekrāna platuma; palaižot rīku vaicājuma daļa ir paslēpta, kas pirmajā brīdī izraisa neizpratni par rīka darbību, ja nav veikta apmācība.

Ontoloģijas daļā ir pieejamas dažas pogas, kas atver izkrītošos nolaižamos sarakstlodziņus ar funkcionalitātei atbilstošu sarakstu:

- Augšā loga ir brīvi pārvietojami objekti ar iespēju atstāt tikai vienu aprakstošo rindu
  - pa kreisi ir ontoloģijas nosaukums;
  - vidū ir predikātu saraksts;
  - pa labi ir izvēlētā objekta raksturojums (vārds, tips un IRI adrese);
- Apakšā loga ir fiksēti objekti un to grupas:
  - kreisajā stūrī ir divas pogas: filtri ontoloģijas attēlošanas pielāgošanai un ontoloģijas īss raksturojums;
  - loga vidū ir izvēlēta elementa raksturojums no ontoloģijas skatu punkta (piemēram, klases definīcija (*declaration*) vai atdalīto (*disjoint*) klašu mezgls ar atbilstošu klašu pārskaitījumu);
  - labajā stūrī ir ontoloģijas izskata izmēra un fokusēšanas uz izvēlētā objekta pogas.

Predikātu saraksts satur klases un attiecības alfabētiskajā secībā, objektiem ir marķējums pirms nosaukuma ("C" burts klasēm un "R" burts saitēm jeb attiecībām starp klasēm). Sarakstam ir pieejama meklēšana, kas atlasa rezultātus pēc katra jaunā simbola ievadīšanas.



1.10.att. **SPARQLing rīka nosacījuma pievienošana [61]**

Rīka autori pasvītro, ka viena no rīka īpašībām ir Graphol ontoloģijas attēlošanas valodas izmantošana vaicājuma zīmēšanai [44], tāpēc no ontoloģijas attēla ir iespējams pievienot atbilstošu klasi, izvēloties to un uzklikšķinot uz pelēkajā šaurajā joslā apakša atrodamas "+" pogas. Funkcija ir šķietami ērta, bet nezinot par to ir slikti pamanāma, citu – piemēram, vilkšana un nomešana – iespēju pievienot klasi vaicājumam nav. Pēc klases pievienošanas tekstuālais vaicājums mainās atbilstoši darbībai, definējot jauno mainīgo. Interesanti realizēta nosacījuma definēšana vaicājumā: teksta beigās ir "+ filter" poga, pēc pogas nospiešanas paradās šablons,

kurā ar vilkšanas palīdzību ievieto atbilstošu mainīgo (1.10.att.). Tas atgādina SPARQLBlocks rīka darbības principu, tajā skaitā vizuāli.

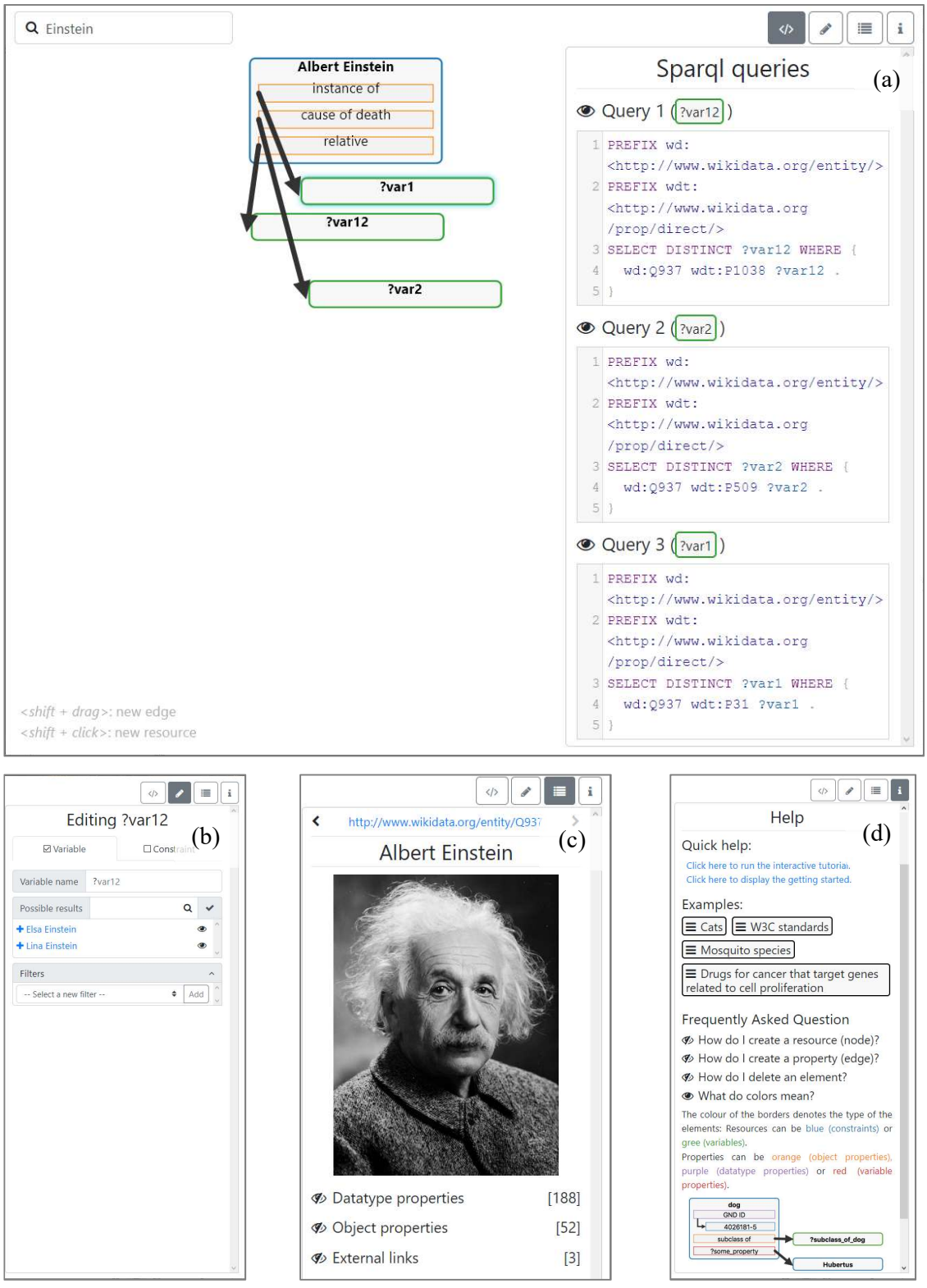
Tomēr SPARQLing rīkam pagaidām ir dažas funkcionalitātes vai to trūkums, kuri izraisa sajūtu, ka rīkam nepieciešama uzlabošana, piemēram, pietrūkst klases ievietošana diagrammā ar vilkšanas un nomešanas metodes atbalstu vai ar dubultklikšķa palīdzību. Arī nav skaidrs, cik labi rīks darbosies ar lielām ontoloģijām, kuru attēlošana ir apgrūtināta; iespējams, to atrisinās ontoloģijas filtrēšana un meklēšana, bet uz pieejamajām nelielajām ontoloģijām filtru efektivitāti un meklēšanas ātrdarbību neizdevās novērtēt.

Jaunākais RDF Explorer rīks ir pieejams tiešsaistē un darbojas ar Wikidata datiem [62]. Rīka izstrādātāji apgalvo, ka viņu mērķis ir rīks, kas ļauj definēt vaicājumu lietotājiem bez iepriekšējām zināšanām par SPARQL vaicājumvalodu un semantiskā tīmekļa jēdzienu; tomēr lasot publikāciju nācās konstatēt, ka testu grupai bija paredzama vismaz 5 minūšu apmācība [47]. Arī pētot rīka saskarni tika pamanīts, ka vaicājumos izmanto tehnisko pierakstu, kas definētajam lietotājam nevar būt labi saprotams, tomēr iz diagrammas daļas nosaukumi ir vairāk dabiskā valodā (ja nav izmantoti termini) (1.11.att.(a)).

Saskarnes kreisajā augšējā stūrī atrodams meklēšanas logs, pēc vērtības ievadīšanas vai mainīšanas rezultātu saraksts tiek atjaunots; izvēlētais rezultāts ar vilkšanas un nomešanas metodi tiek ievietots no saraksta diagrammā. Labajā augšējā stūrī atrodamas 4 pogas, kas ļauj pārslēgties starp vaicājumu, mainīgo, objekta un palīdzības logiem (1.11.att.(a)-(d)). Kreisajā apakšējā stūrī atrodams atgādinājums par iespējamām taustiņu kombinācijām. No apskatītas saskarnes ir viegli pamanīt, ka no apskatītājiem rīkiem RDF Explorer rīkam ir pēc elementu skaita viena no nabadzīgākām sakarnēm. Parastiem lietotājiem, kas nav IT specialisti, tas varētu atvieglot darbu, jo samazina darbību skaitu, ļaujot vieglāk atcerēties pareizo secību vaicājuma sastādīšanai.

Vaicājuma logā (1.11.att.(a)) katram mainīgajam ir savs vaicājums, tos var samazināt līdz virsrakstam, nospiežot "acs" ikonu. Vaicājuma teksts ir rediģējams, bet izmaiņas netiek saglabātas, arī nav nekādu iespēju apvienot vairākus vaicājumus, kas ir saistīti ar vienu objektu. Iespējams, tālāk ar rediģēšanu tiks saistīta kāda funkcionalitāte, bet dotajā brīdi tas vairāk traucē. Mainīgā vārds ir interaktīvs, nospiežot to, vaicājuma logs ir aizvietots ar mainīgā rediģēšanas logu; atgriezties atpakaļ pie vaicājumiem var, nospiežot atbilstošu pogu labajā stūrī.

Mainīgā rediģēšanas logā (1.11.att.(b)) ir pieejamas divas cilnes, viena ļauj nomainīt mainīgā vārdu, apskatīties iespējamus rezultātus un uzlikt filtru, otrā paredzēta nosacījumu veidošanai. Pirmā cilne ir saprotama un vienkārša, tai pievienota meklēšana iespējamos rezultātos, otrā prasa ievadīt URI, kas lietotājam bez iepriekšējām zināšanām var rast problēmas



1.11.att. **RDF Explorer rīka saskarne [6]:**

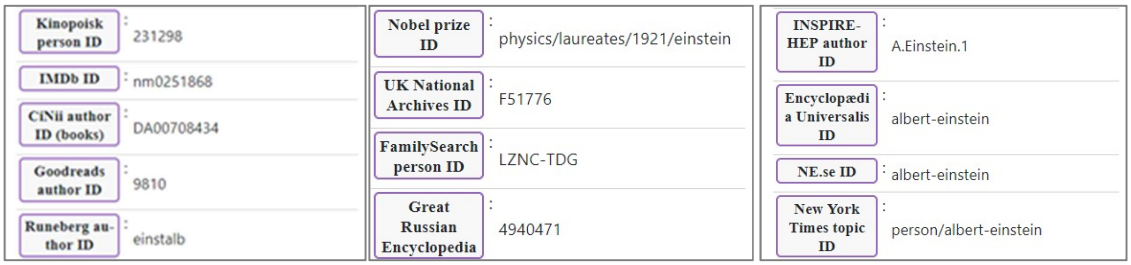
(a) saskarnes kopskats ar vaicājumiem tekstuālā formā, (b) mainīgā rediģēšanas logs, (c) objekta aprakstošais logs, (d) palīdzības logs

pat mēģinot izprast, kas ir jādara. Iespējas ievietot URI no saraksta nav, tikai kopējot (ja ir zināms, kur to var atrast) vai ierakstot ar roku, kas ir papildus kļūdu cēlonis.

Objekta aprakstošais logs (1.11.att.(c)) piedāvā pievienot objektam īpašības no saraksta, veidojot jau ar saiti savienotu objektu. Īpašības ir sadalītas kategorijās: datu tipa īpašības, objekta īpašības un ārējās saites. Pirmā tipa īpašībām ir vērtības (skaitļi, simbolu virknes), otrā tipa īpašības ir jaunie objekti uz diagrammas, t.i. citas klases vai instances. Viens no šī loga trūkumiem ir meklēšanas neesamība, kā arī kārtošana pēc nezināma principa, kas, ņemot vērā lielu vērtību skaitu, būtiski apgrūtina uzdevumu atrast nepieciešamo. Piemēram, Einšteina gadījumā:

- 188 lauki ar skaitliskām vērtībām, pie tam daudzi no tiem ir noteiktā sistēmā izmantotais identifikators (1.12.att.);
- 52 objekta īpašības, daudzām no tām pieejami vairāki precizējumi, piemēram, dzimums ir tikai viens, bet dažādas iestādes (viena kategorija), kurās Einšteinam bija akadēmiskie amati, ir aptuveni 20.

Pat ja apskatīties katru kategoriju atsevišķi, saraksts nav labi pārskatāms.



1.12.att. **Einšteina objekta datu tipa īpašību fragments oriģinālajā secībā [62]**

Pētot rīka saskarnes iespējas, nācās secināt, ka rīkam ir daži citi trūkumi, kas pasliktina lietotāja pieredzi. Pirmais pamanītais trūkums ir saišu vilkšana starp diviem objektiem diagrammā: tam nepieciešams uznest vienu objektu virs otrā, tad izveidosies saite, bet pēc tam vēl nepieciešams patērēt laiku objektu telpiskai atdalīšanai un ērtākai novietošanai. Otrais būtiskais trūkums veidojas, ja ir izdzēsts objekts: rīks "atceras" pēdējo objektu un nospiežot objekta aprakstošā loga izsaukšanas pogu, neizvēloties objektu diagrammā, tiek parādītas izdzēsta objekta īpašības. Vēl viens risinājums, kas pasliktina vaicājum uztveri, ir mainīgā iznešana atsevišķajā diagrammas objektā, kas komplicētam vaicājumam būtiski palielinās diagrammas objektu skaitu.

Apkopojot lietotāja saskarnes risinājumus ir secināts, ka apskatīto rīku izstrādātāji piedāvā dažādus saskarnes dizainus, kas, viņuprāt, sekmētu vaicājumu sastādīšanu un pareizo uzdevuma risinājuma atrašanu. Tomēr daudziem risinājumiem ir kopīgi elementi:

- Ļaut lietotājam piekļūt datu shēmai tiešā (piemēram, diagrammas formā) vai apslēptā (klases vai atribūtu meklēšana) veidā;
    - Dažos rīkos izmanto vaicājuma pakāpenisko veidošanu, kad elementus var pievienot pie jau esoša elementa, izvēloties no atļauto elementu saraksta;
- Piedāvā ātru pieeju biežāk lietotām funkcijām vienā vai citā veidā, piemēram, īsinājumizvēlnē, rīkjoslā vai kā pogu;
- Izstrādā informatīvo elementu attēlošanu:
    - Krāsu un/vai formu izmantošana elementu kategorizēšanai;
    - Saistīto elementu apvienošana vienā grafiskā elementā, piemēram, klases atribūtu un nosacījumu ievietošana klases objektā;
- Vaicājuma pieejamība SPARQL vaicājumvalodā, dažreiz pievienojot rezultātus (kā kopējo rezultātu skaitu vai pilnā formā);

Daļa no risinājumiem samazina iespējamo kļūdu skaitu pareizrakstības vai precīza nosaukuma nezināšanas dēļ, citi atvieglo vaicājuma uztveri, tomēr visiem rīkiem ir kopīgs pēc iespējas mazāka SPARQL vaicājumvalodas konstrukciju izmantošana. Bet neskatoties uz rīku skaitu, nav izstrādāta vienota pieeja saskarnes dizaina jautājumam un nepieciešamo saskarnes elementu sarakstam, izskatam un novietojumam, kas ļauj veidot jauno rīku izmantojot pieredzi, veselu sapratu un savus priekšstatus par rīka funkcionēšanu.

1.3. Vizuālu vaicājumu rīku efektivitātes novērtējumi

Lai novērtēt jebkuru vaicājumu rīku neatkarīgi no to darbības principa, ir nepieciešams piedāvāt lietotājiem vaicājumus un novērtēt to objektīvus (patērētais laiks, rezultātu atbilstība uzdevumam) un subjektīvus (ērti, vienkārši, patīk) rādītājus. Lietotāji parasti pārstāv vienu no trim kategorijām:

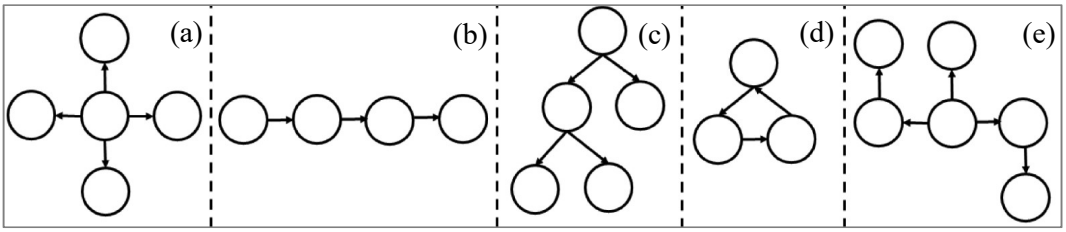
- nozares specialisti, kuriem ir liela pieredze vaicājumu rakstīšanā izmantojot kādu tekstuālo vaicājumvalodu;
- cilvēki ar labām zināšanām IT jomā, bet kas nav specializējušies informācijas iegūšanā ar vaicājumu palīdzību, parasti tas nozīmē, ka ir vai tiek iegūta augstāka izglītība datorzinātnes kādā no virzieniem;
- lietotāji, kam nav pieredzes vaicājumu jomā ne darbā, ne izglītības iegūšanas laikā.

Pirms novērtēšanas testa vai tā daļas veikšanas var notikt apmācība. Kategorijas izvēle un apmācības nepieciešamība atbilst rīka lietošanas scenārijam, kuru nosaka paši izstrādātāji.

Vēl viens svarīgs punkts ir vaicājumu sarežģītība, t.i. cik daudz un kāda koka formas ir RDF trijnieki (ja novērtēta ir SPARQL vaicājumvaloda), kas ir izmantoti vaicājumos. Vienā no rakstiem ir piedāvāta piecu tipu sistēma:

1. Zvaigzne: centrā ir atrodams meklētais objekts, bet visi citi apgalvojumi ir saistīti ar to;
2. Ķēde: apgalvojumi veido ķēdi bez atzariem;
3. Koks: ķēdei var būt atzari;
4. Cikls: apgalvojumi veido slēgtu struktūru;
5. Sarežģīta forma: iepriekšējo formu kombinācija (1.13.att.) [63].

Izpētot novērtēšanas testos piedāvātos vaicājumu ir secināts, ka tie pieder pirmiem 3 tipiem, neatkarīgi no lietotāju kategorijas; bieži vien trijnieku skaits nepārsniedz divus apgalvojumus (atlases kritērijus, kar ir klases atribūti, netiek skaitīti par atsevišķu apgalvojumu, ja skaitīt tos, tad skaitlis ir nedaudz lielāks, bet tipiski tas nepārsiedz 5). Arī pārrunājot tipiskus vaicājumus ar nozares specialistiem un apskatot dažādos apmācību avotos piedāvātus piemērus (piemēram, Wikidata [7] vai Europeana vaicājumu sistēmas [64]) tika secināts, ka tā ir tipiskāka vaicājumu uzbūve.



1.13.att. **Vaicājumu pamata tipi [63]:**
(a) zvaigzne, (b) ķēde, (c) koks, (d) cikls, (e) sarežģītas formas

Analizējot vizuālu vaicājumu rīku dažādus novērtējumus netika atrasts visaptverošais pārskats par vismaz kādā brīdī pieejamo rīkiem. Sastaptus salīdzinājumus var sadalīt pēc pētījuma mērķa un sarežģītības:

- Vienkāršs, kur lietotāju rezultāti izmantojot vizuālu vaicājumu rīku ir salīdzināti ar tekstuālo vaicājumvalodu;
- Vairāku riku salīdzinājums uz vienādiem vaicājumiem;
- Literatūras analīze, kurā autori salīdzina literatūrā dotos datus pēc funkcionalitātes pieejamības vai citiem kritērijiem, t.i. nav veikta analīze balstoties uz vienādiem vaicājumiem.

Vienkāršā pētījuma piemērs ir Katarci (*Tiziana Catarci*) un Santuči (*Giuseppe Santucci*) publikācijā aprakstītais eksperiments, kurā piedalījās 102 cilvēki, kas tika sadalīti sešās grupās pēc pieredzes vaicājuma sastādīšanas jomā un uzdevumu tipa, kurus lietotāji veica eksperimenta laikā. Rezultāti rāda, ka SQL vaicājumvalodas gadījumā vizuāls vaicājuma rīks ļāva ātrāk un/vai precīzāk izpildīt uzdevumus [65].

Divu vai vairāk rīku salīdzinājumiem bieži vien par mērķi ir nosaukts labākas pieejas atrašana, un tādēļ autori apskata pēc savas būtības dažādus rīkus. Piemēram, Kaufmane (*Esther Kaufmann*) un Bernšteins (*Abraham Bernstein*) salīdzināja 4 rīkus, no kuriem divi bija uz dabiskas valodas balstīti, viens – uz kontrolētas dabiskas valodas un viens uz diagrammām balstīts, piedāvājot testa personām no dažādām nozarēm (kopskaitā 48) atbildēt uz vienkāršiem jautājumiem ar vaicājumu palīdzību bez iepriekšējas apmācības. Sagaidāmi, ka lietotājiem pie tādiem nosacījumiem labāki sasniegumi bija ar uz dabiskas valodas balstītiem rīkiem [55].

Līdzīgs pētījums, bet piedāvājot apmācību un apskatot arī ekspertus, sasniedza citus rezultātus: starp uz dabiskas valodas, uz kontrolētas dabiskas valodas, uz formām un uz diagrammām balstītajiem rīkiem veidojot nedaudz komplicētākus vaicājumus lietotāji zemāk novērtēja tieši uz dabiskas valodas balstītus rīkus. Par vienu no iemesliem tam uzskata uz dabiskas valodas balstītu rīku nespēju darboties ar sinonīmiem; ekspertu vidū par atvieglojošo faktoru nosauca iespēju redzēt vaicājumu tekstuālā formā. Viens no faktoriem, kuru lietotāji nosauca par iemeslu vieglākai vaicājuma sastādīšanai ir pieeja ontoloģijas datiem [56].

Salīdzinot uz formām balstīto PepeSearch un uz diagrammām balstīto OptiqueVQS rīkus Vega-Gorgojo (*Guillermo Vega-Gorgojo*) ar līdzautoriem arī novēroja rīka novērtējuma atkarību no lietotāju pieredzes. PepeSearch rīks tika uzskatīts par vieglāk apgūstāmu un saprotamāku, bet Optique rīks pēc lietotāju domām ļāva sasniegs precīzākus rezultātus un, kas ir diezgan negaidīti, labāk koncentrēties uz uzdevuma risināšanas. Izvēloties no diviem rīkiem, nozares specialisti viennozīmīgi (70%) izvēlējas Optique rīku, bet ne-specialistu viedokļi sadalījās aptuveni vienādas daļas, dodot nelielu – viena balss lielu – priekšroku PepeSearch rīkam [66].

Viens no pēc rīku skaita apjomīgākajiem darbiem, kas tika sastapts literatūrā, ir Vargasa (*Hernán Vargas*) ar līdzautoriem publikācija, kurā zinātnieku grupa prezentē savu izstrādājumu – RDF Explorer rīku. Tajā ir salīdzināt 13 uz dažādiem principiem balstītie rīki, un 8 no tiem ir uz diagrammām balstītie; tomēr pašai rīku analīzei autori nepievērš daudz uzmanības. Autori izvēlējas četrus kritērijus, pēc kuriem salīdzināja rīkus:

- Automātiskā pabeigšana (*Autocomplete*) – rīka spēja piedāvāt ievadīta teksta saturošus rezultātus, kas atbilst pieejamajiem datiem;
- Vaicāšana balstoties uz piemēra (*Example-based Querying*) – iespēja izmantot gatavu vaicājumu un uz tā bāzes veido sev nepieciešamo;
- Dinamiski rezultāti (*Dynamic results*) – rezultātu pārrēķināšana lai ļaut lietotajam novērtēt līdz dotam brīdim paveikto;
- Netukšie rezultāti (*Non-Empty Results*) – neļaut izveidot tādu vaicājumu, kuram nav rezultātu [47].

Vargass ar līdzautoriem neslēpj, ka kritēriju izvēle nav nejauša, bet virzīta uz sava produkta galveno īpašību izpēti, kuru saraksts, savukārt, ir radies no cilvēka un datora mijiedarbības virziena pētījuma par prasībām, kas tiek izvirzīti no šī skatu punkta [67].

1.1. tabula

Uz diagrammām balstītu vizuālu vaicājumu rīku salīdzinājums [47]
Apzīmējumi: AP – automātiskā pabeigšana, PA – paraugu un/vai apmācība rīkā; DR – dinamiskie rezultāti; NtR – netukšie rezultāti

Npk	*Rīks*	*Gads*	*Īpašība*				*Lietotāju skaits*	*Avots*
			AP	*PA*	*DR*	*NtR*		
1	Nitelight	2008.	-	-	-	-	-	[25]
2	RDF-GL	2010.	-	-	-	-	5	[26]
3	Smeagol	2011.	X	X	X	X	43	[27]
4	QueryVOWL	2015.	X	-	-	-	6	[68]
5	OptiqueVQS	2018.	X	-	-	-	10	[69]
6	SPARQLing	2018.	-	-	-	-	-	[44]
7	ViziQuer	2018.	-	-	-	-	14	[70]
8	RDF Explorer	2019.	X	X	X	X	28	[47]

Literatūras analīzes rezultātā tikai viens rīks ir sasniedzis tik pat labus rezultātus kā RDF Explorer rīks (1.1.tab.) [47]. Publikācijas autori nepaceļ jautājumu par to, vai kādas īpašības neesamība vairākos rīkos liecina par to, ka tā nav vajadzīga vai pat kaitnieciska, par ko nedaudz detalizētāk tikt pateikts analizējot ViziQuer rīka saskarni 2.1. apakšnodaļā. Arī nav saprotams, kāpēc autori uzskata, ka QueryVOWL rīkam nav dinamisko rezultātu, kas, pārbaudot rīka tiešsaistes versiju, tika novērots. Iespējams, ka analīze ir balstīta tikai uz literatūras avotiem, un

Vargass ar līdzautoriem nav pamēģinājuši šos rīkus, bet atbilstošo rīku izstrādātāji neuzskatīja šo īpašību pieminēšanu rakstos par nepieciešamu.

Pēc literatūras avotos aprakstītiem novērtēšanas testiem vizuālo vaicājumu rīki līdzi ar uz dabiskas valodas balstītajiem pārspēj tekstuālos vaicājumu rīkus, bet starp lietotājiem nav vienota viedokļa, kāda pieeja ir labāka. Lielu lomu rīku novērtēšanā spēlē izstrādātāju noteikti rīku izmantošanas parametri (potenciālo lietotāju pieredze, terminoloģijas izmantošana, apmācība), kurus jāņem vērā gan sastādot vaicājumus, gan izvēloties potenciālus lietotājus novērtēšanas testu veikšanai.

2. VIZIQUER VIZUĀLAIS VAICĀJUMU RĪKS

ViziQuer vizuālu vaicājumu rīku izstrādāja un turpina attīstīt LU MII pētnieku grupa K.Čerāna vadībā, tas ļauj vaicāt pār RDF formātā uzdotiem datiem. Rīks atbalsta vairākas SPARQL vaicājumvalodas konstrukcijas, un tā saskarnē ir izmantoti daži termini, kas apgrūtina darbu cilvēkam bez iepriekšējas pieredzes vai zināšanām par vaicājumiem, kas samazina potenciālo lietotāju loku, bet ļauj paplašināt rīka iespējas. Tā ir viena no būtiskākām ViziQuer rīka atšķirībām no citiem vizuālajiem vaicājumu rīkiem jau idejas līmenī: no sākuma ir definēts mērķis izveidot tādu rīku, kas ir vērsts ne uz vienkāršākiem pielietojumiem, bet uz tādiem, kur ir nepieciešama sarežģītu vaicājumvalodas konstrukciju izmantošana, atbalstot negācija, agregācijas un apakšvaicājumus. Bagātīgu (*rich*) vaicājumu definēšana lietotājiem bez labām SPARQL vaicājumvalodas zināšanām, bet ar priekšstatiem par vaicājumiem kā tādiem, nosaka rīka sarežģītību visos aspektos, sākot ar uz darba uzdevumiem vērstiem diagrammu sadalījumiem un beidzot ar nelielām funkcionalitātēm, kas ir pievienotas saskarnei. Uz darba izstrādes laiku nav zināms neviens cits rīks, kas atbalstītu tik pat plašu SPARQL vaicājumvalodas konstrukciju skaitu, vai pat izvirzītu par mērķi darbu ar bagātīgiem vaicājumiem.

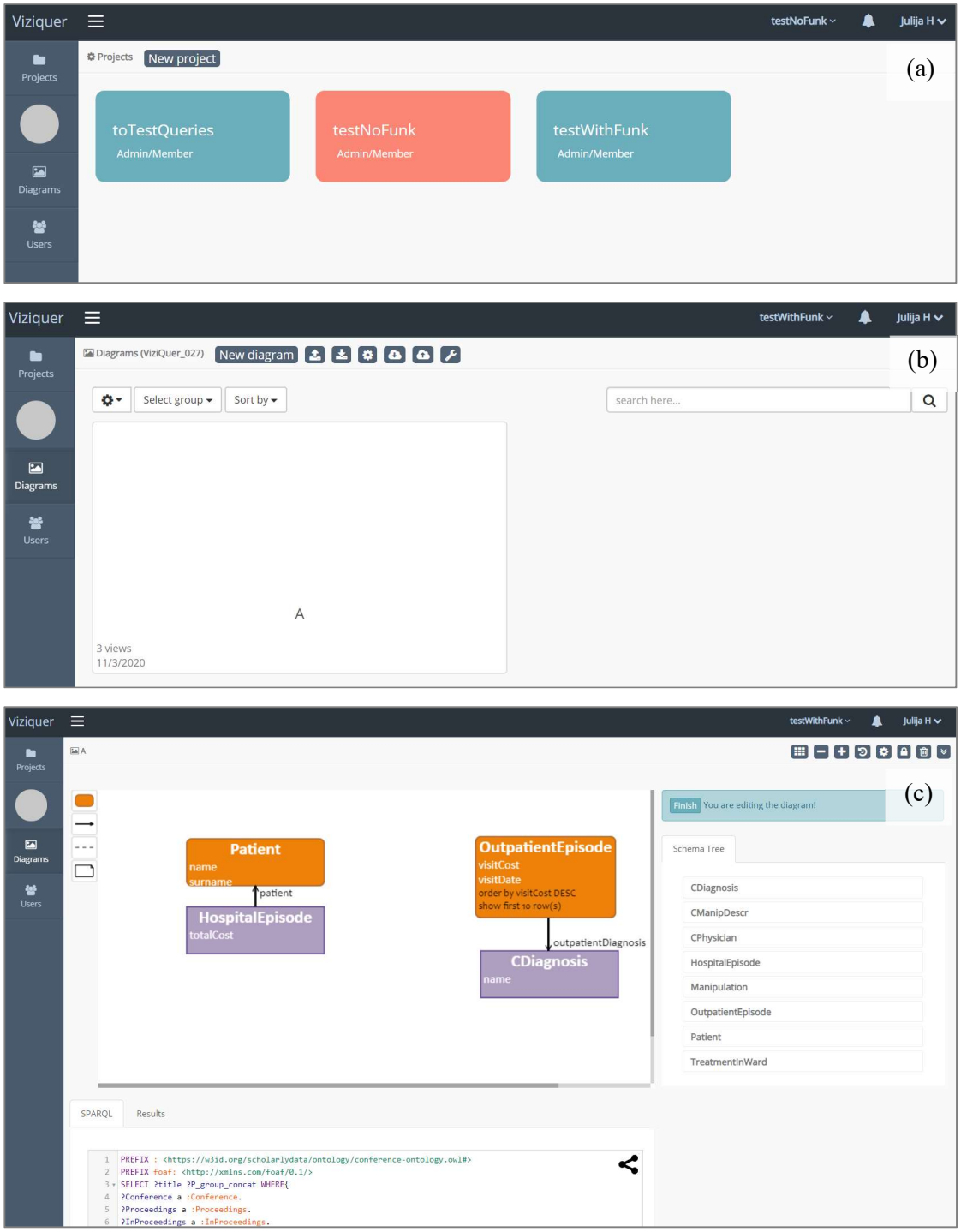
ViziQuer rīka pirmā versija tika izstrādāta kā darbvirsmas programma [71 – 73], vēlāk tika īstenota rīka pāreja uz tiešsaistes versiju, kad rīks ir pieejams kā tīmekļa lapa [57, 74-76]. Šī pāreja ļāva atrisināt vairākus jautājumus: piedāvāt potenciālajiem lietotājiem iespēju pārbaudīt rīku darbībā bez nepieciešamības konfigurēt to pašiem; piedāvāt rīku lielākam lietotāju skaitam, atvieglot rīka atjauninājumu vai speciālo konfigurāciju piedāvāšanu lietotājiem atbilstoši prasībām; kā arī pieslēgties pie dažādām datu bāzēm un darboties ar tām. Darbvirsmas rīks ir joprojām pieejams, bet to pēdējā versija ir izstrādāta 2016. gadā [77], bet tiešsaistes rīka versija ir aktīvi attīstīta [78].

2.1. Lietotāja saskarnes risinājumu analīze

ViziQuer rīks ir paredzēts darbam ar patvaļīgo datu avotu, kas atbilst noteiktai struktūrai, tāpēc izstrādājot rīku tika pieņemts lēmums sadalīt vaicājumus projektos: apvienojot vaicājumus, kas darbojas ar vieniem un tiem pašiem datiem, vienā kategorijā (2.1.att.(a)). Katrs projekts – ja tas nav tukšs – satur vienu vai vairākas diagrammas (2.1.att.(b)), kas, savukārt, satur vaicājumus (2.1.att.(c)). Diagrammu jēdziena ieviešana ļauj sadalīt vaicājumus smalkāk, kas no praktiskā pielietojuma skatu punkta ļauj izveidot, piemēram, vaicājumu grupu pēc kopēja temata: kvartāla atskaite, ārstu statistika, uz noteikto datumu sagatavoti vaicājumi un tml. No iepriekš apskatītajiem rīkiem neviens nepiedāvāja tik smalku dalījumu, daļa no rīkiem ļauj veidot tikai vienu vaicājumu uz ekrāna, citiem to skaits nav ierobežots, bet vaicājumu apvienošana grupā netika pamanīta nevienā rīkā. Vairāku vaicājumu esamība uz viena ekrāna var sekmēt vaicājuma sastādīšanu pēc līdzības principa, izmantojot jau gatavu vaicājumu kā šablonu, tomēr sākumā tas var izraisīt noteikto neizpratni, kā pareizi sastādīt vaicājumu, tāpēc lietotājiem ieteicama vismaz neliela apmācība, kas atvieglo pirmos soļus.

Visiem logiem ir kopējā daļa kreisajā pusē - rīkjosla, kas ļauj pārslēgties starp dažādiem skatiem: projektu sarakstu, diagrammas skatu, lietotāja informāciju, bet izstrādātājiem – arī paskatīties rīka konfigurāciju, kurā var modificēt pieejamas funkcijas, elementu attēlojumu detaļas, piemēram, formu vai krāsu. Rīkjoslu var paslēpt, izmantojot augšējā daļā novietotu simbolu no trim horizontālām līnijām, palielinot galvenajai daļai pieejamu vietu. Rīkjoslai nav tikai ikonu režīma, ka ir Visual SPARQL Builder rīkam, jo paskaidrojošais teksts ir mazāks pēc izmēra un novietots zem attēla, un rīkjoslas platums, pretēji SPARQLing rīkam, ir fiksēts, jo tajā nav maināma vai nezināma izmēra elementu.

Projekta skatā ir ekrāna augšējā daļā pieejamas vairākas funkcijas, kas ir nepieciešamas lietotājiem: izveidot diagrammu, pievienot vai izeksportēt datu shēmu un pašu projektu, apskatīt un rediģēt iestatījumus, kā arī nomainīt izmantoto konfigurāciju, kas atvieglo pāreju uz jauno rīka versiju, ja tajā ir kļuva pieejamas jaunas funkcionalitātes. ViziQuer rīka realizēta darba eksportēšana kā JSON formātā datne pēc savas būtības aizvieto saglabāšanu, kas ir sastopama dažos apskatītajos rīkos; administratora tiesības ļauj līdzīgā veidā darboties arī ar konfigurāciju, tādējādi ļaujot pielāgot rīku lietotāju grupai. Par iespēju konfigurēt rīku citos darbos netika minēts, iespējams, saglabājot to izstrādātāju ziņā un neļaujot lietotājiem nejauši sabojāt rīku. ViziQuer rīkam šī potenciālā problēma ir novērsta ar to, ka var saglabāt gan pašu

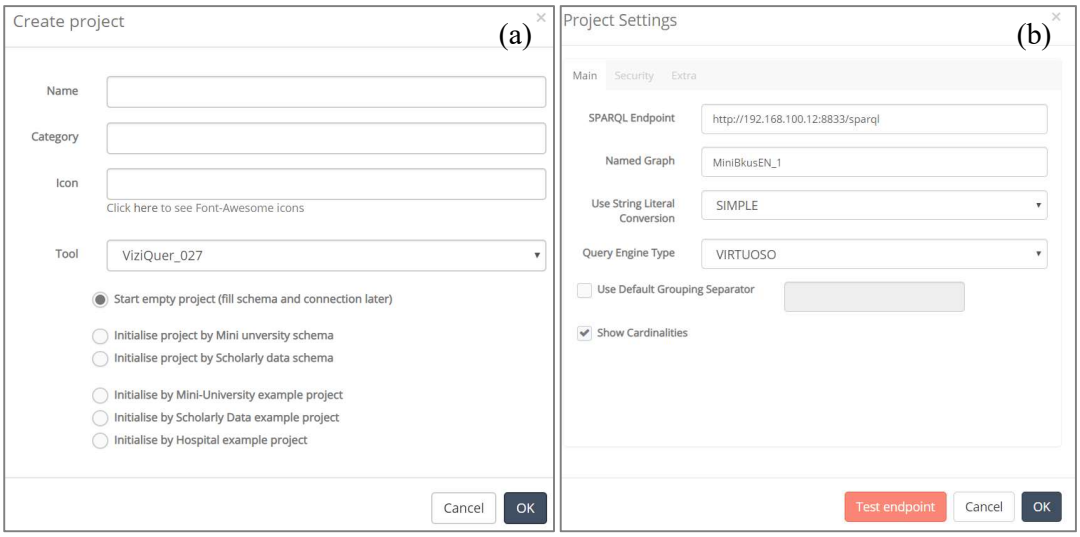


2.1.att. **ViziQuer rīka tiešsaistes versijas saskarne:**

(a) projektu, (b) projekta diagrammu, (c) diagrammas līmeņos

projektu, gan arī konfigurāciju, pirms veikt kādas izmaiņas, papildus tam tiesības darboties ar konfigurāciju piešķir tikai daļai no lietotājiem.

Projekta izveides logā ir iespējams pievienot jau izveidotā projekta datus (2.2.att.(a)), kas ļauj sākt ar darbam gatavu iestatījumu komplektu. Cits no apskatītajiem rīkiem atšķirīgs loga punkts ir iespēja izvēlēties rīka konfigurāciju (*Tool*), pielāgojot rīku savām vajadzībām vai darba prasībām. Citādi logam ir tipiski lauki, piemēram, projekta nosaukums (*Name*), jo tehniskie projekta parametri ir izdalīti atsevišķajā logā: iestatījumu logā (2.2.att.(b)), piemēram, var pievienot galapunktu, kur glabājas dati, un nomainīt diagrammās attēloto informāciju, piemēram, saites kardinalitātes attēlošanu. Šie parametri tiks izmantoti visās projekta diagrammās.



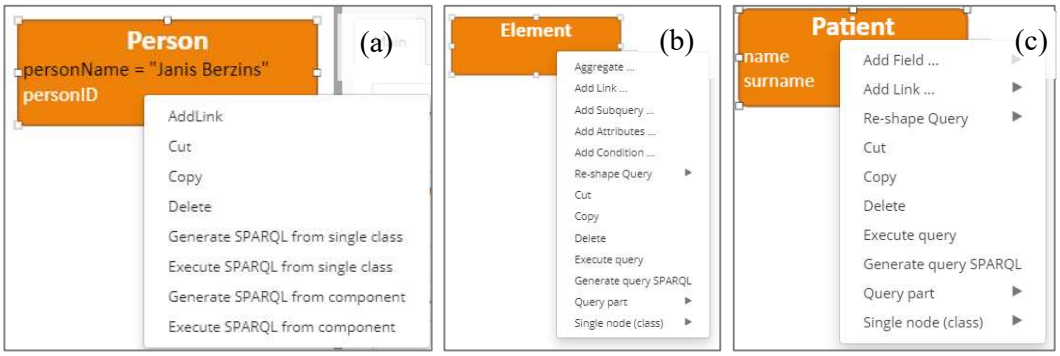
2.2.att. **ViziQuer rīka projekta logi:**
(a) izveides, (b) iestatījumu

Diagrammas izveides logs ir lakonisks, ļaujot ievadīt to nosaukumu, jo diagrammas parametrus nosaka projekts. Diagrammas logu (2.1.att.(c)) nosacīti var sadalīt 5 daļās:

- Diagrammas loga augšējā daļā ir josla, kuras kreisajā pusē ir diagrammas vārds, bet labajā – funkciju komplekts no:
  - režģa ieslēgšanas/izslēgšanas pogas ērtākai zīmēšanai un novietošanai;
  - pogām tālummaiņai,
  - aprakstošas informācijas par diagrammu un pieejas tiesībām izsaukšanu;
  - izdzēšanas pogu un zīmēšanas daļas paslēpšanu, atstājot tikai SPARQL vaicājumvalodā uzrakstīto vaicājumu;

- Kreisajā daļā ir rīkjosla ar zīmēšanai paredzētajiem rīkiem, bez vaicājuma elementiem ir iespējams pievienot arī komentāra objektus, atstājot savas piezīmes;
- Centrālajā daļā ir zīmēšanas lauks, tas aizņem lielāko daļu, ļaujot novietot papildus funkcijas apkārt, bet galveno daļu atstājot centrā, kas ir visiem rīkiem tipisks risinājums;
- Labā daļa ir funkcionāla, tā mainās atkarībā no lietotāja darbībām:
  - Ja nav izvēlēts neviens elements, tad tajā atrodas visas klases, kas ir nosaukti datu shēmā, ar dubultklikšķi tos var ievietot zīmēšanas daļā, kas ļauj lietotājam ātrāk pievienot klasi un nodrošina netiešu pieeju datu shēmai;
  - Ja ir izvēlēts kāds objekts diagrammā, tad ir pieejamas funkcionalitātes, kas atbilst dotajam objektam: vārdu maiņa, īpašību un nosacījumu pievienošana, objekta tipa maiņa un tml.;
- Apakšā atrodama daļa ar divām cilnēm, vienā ir izveidotais SPARQL vaicājumvalodā uzrakstītais vaicājums, bet otrajā – vaicājuma rezultāti, ja projektam ir definēta datu bāze.

No iepriekš apskatītajiem rīkiem tikai Optique rīks piedāvā pievienot vaicājumam papildus aprakstošo informāciju, šim rīkam vaicājuma pirmais elements satur tā nosaukumu un aprakstu. ViziQuer rīks piedāvā lietotājam ievietot patvaļīgo komentāru skaitu, tādējādi ļaujot izveidot arī diagrammu, kas piemērota apmācībai neklātienē, vai detalizēti aprakstītu definēto vaicājumu un tā daļas.

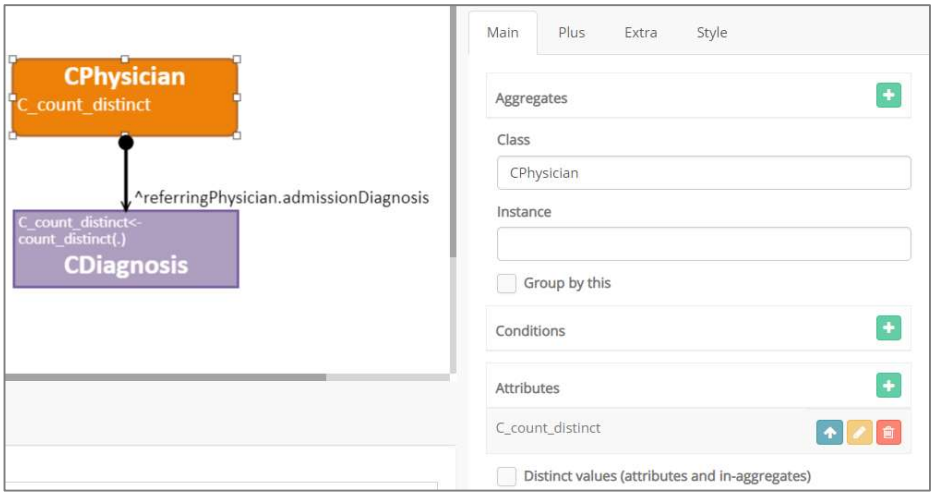


2.3.att. **ViziQuer rīka viena objekta īsinājumizvēlne:**
(a) pirms 2 gadiem [58], (b) pirms pusgada [79], (b) tagad

ViziQuer rīka saskarnē ir vēl viens elements, kas nav redzams uzreiz – īsinājumizvēlne, kuras saturs ir atkarīgs no konfigurācijas un izvēlēta elementa. Sākotnēji tā saturēja tikai kopēšanai un izdzēšanai nepieciešamas funkcijas, bet attīstot rīku radās nepieciešamība piedāvāt lietotājam jaunas bieži izmantotas iespējas, kas atvieglo vaicājuma veidošanu, ar minimālo darbību skaitu, tāpēc pakāpeniski tika pievienotas funkcionalitātes (2.3.att.(a)), un

kad to skaits palielinājās līdz pietiekami lielam, funkcionalitātes kļuva grūti caurskatāmas (2.3.att.(b)), un tika ievestas kategorijas, apvienojot daļu no funkcionalitātēm pēc līdzības principa (2.3.att.(c)). Kategorijas ir vizuāli sadalītas 3 daļās: vaicājuma veidošana, kopēšana-izdzēšana un ar vaicājuma izpildi saistītas darbības (izveidot vaicājumu SPARQL vaicājumvalodā un izpildīt to).

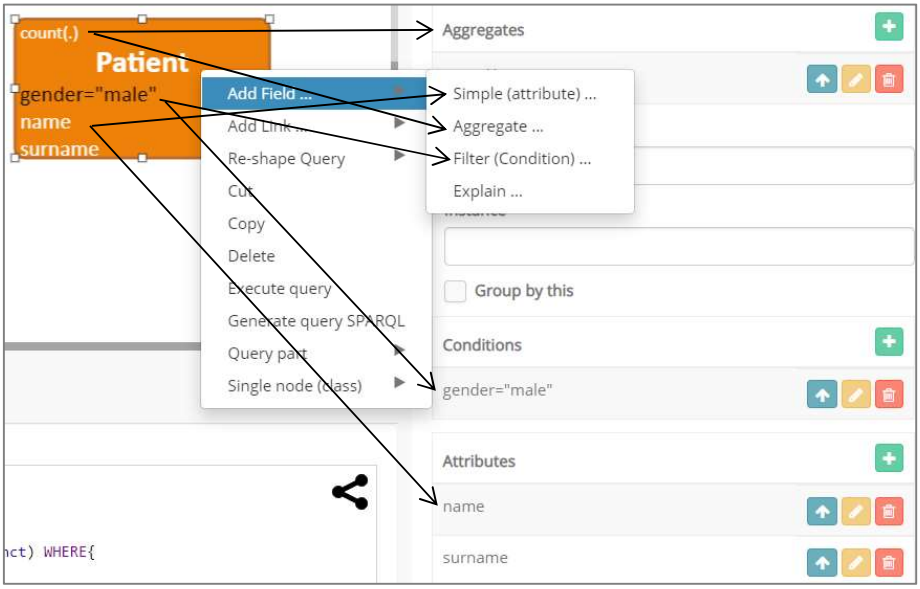
Kā jau bija minēts iepriekš, ViziQuer rīks – līdzīgi QueryVOWL rīkam – ļauj ievietot klases tipa elementu divos veidos, apvienojot zīmēšanas rīku (piemēram, RDF-GL rīks) paņēmienu, kad pirmajā solī diagrammā ir ievietots grafiskais element, kuram pēc tam definēti parametri, sākot ar klases nosaukumu, ar ievietošanu no saraksta, kā tas ir Nitelight, Visual SPARQL Builder un Optique rīkiem. Otrā metode ir ātrāka, bet lielām datu shēmām ar lietotājam nezināmo vai vāja zināmo struktūru tā ir grūtāka, jo ir nepieciešams atrast vajadzīgo klasi. Šajā gadījumā vieglāk ir ievietot objektu un īpašību definēšanai paredzētajā logā (2.4.att.) ievadīt daļu no klases nosaukuma, kas nostrādā kā meklēšana, atlasot tādus klases vārdus, kas satur ievadīto. Jo atlasīšana notiek pēc katra simbola ievadīšanas, tad lietotājam ir iespēja kontrolēt procesu un atrast vajadzīgo vārdu, pat ja tas ir saīsinājums no dabiskā valodā rakstīta nosaukuma.



2.4.att. **Vaicājums un daļa no klases objekta īpašību definēšanas loga**

Citu parametru ievadīšanai ir paredzēti citi īpašību definēšanas loga lauki, piemēram, nosacījumu lauks (*Conditions*), kas nosaka atlasīto instanču ierobežojošas vērtības, un rādāmo atribūtu sarakstu lauks (*Attributes*) (2.4.att.). Tās pašas darbības ir izsaucamas arī no īsinājumizvēlnes, kas ļauj lietotājam pāriet pie nepieciešamas darbības uzreiz, kā arī turpināt veikt darbības ekrāna centrā, nepārslēdzoties uz malām. Īpašību secība īsinājumizvēlnē atšķiras no īpašību definēšanas loga: pirmajā gadījumā tā vairāk atbilst lietotāja darbībām pēc tā

biežuma un vaicājuma izveides, piemēram, lasot vaicājumu dabiskā valodā lietotājs sākumā izvēlēsies to, kas ir izvadāms uz ekrāna, pēc tam definējot citus parametrus; otrajā gadījumā secība atbilst tam, kā parametri ir novietoti diagrammas elementā, kas atvieglo lietotājam meklēšanu, piemēram, ja ir nepieciešams veikt izmaiņas ievadītajā informācijā (to var izdarīt tikai īpašību definēšanas logā) (2.5.att.).



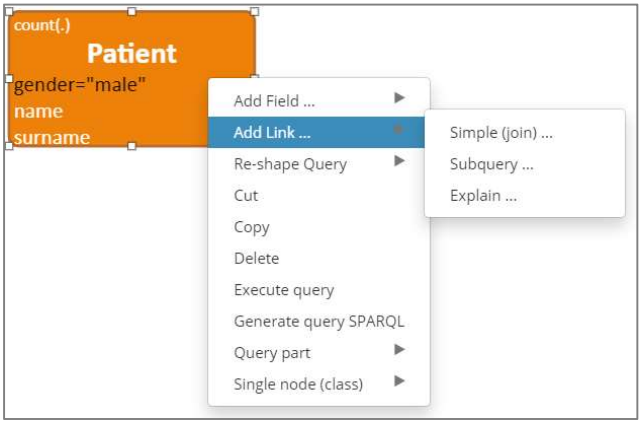
2.5.att. **Īsinājumizvēlnes un objekta īpašību definēšanas loga apakšpunktu novietojuma salīdzinājums**

Pēc tam, kad ir ievietots pirmais elements tālākai vaicājumu attīstīšanai, t.i. jauno objektu pievienošanai vaicājumam, ir iespējami divi veidi: uzzīmēt klases objektu un pēc tam savienot to ar saiti vai pievienot ar īsinājumizvēlnē pieejamu punktu Add Link (2.6.att.). Pirmajā gadījumā lietotājam ir nepieciešams atkārtoti iziet caur visiem posmiem:

- Uzzīmēt objektu;
- Noteikt klases vārdu un īpašības;
- Savienot objektus ar saites elementu (bultas formā);
- Noteikt saites vārdu.

Tāda pieeja ir tipiska rīkiem, kas atgādina zīmēšanas rīkus, bet šajā procesā nezinot datu shēmu ir iespējamas vairākas kļūdas, piemēram, var savienot divas klases, starp kurām nav definēta saite. Šī problēma tika adresēta, piedāvājot lietotājam priekšā teikšanu, kā arī funkciju Connect Classes, kas ļauj izveidot saiti starp divām klasēm, piedāvājot iespējamus ceļus, tajā skaitā tādus, kas satur citas klases.

Otrajā gadījumā lietotājam ir piedāvāts sarakst ar atļautām tieši savienotām klasēm un saitēm, funkcionalitāte pēc savas darbības ir līdzīga pieejai, kuru izmanto tādi rīki, kas piedāvā lietotājam pakāpenisko vaicājuma izveidi, piemēram, Optique rīks. Abu funkciju – Add Link un Connect Classes - pirmās versijas tika izveidotas izstrādājot bakalaura darbu [58]; šī darba ietvaros tās tika pilnveidotas, kas tiks aprakstīts detalizētāk 2.3. apakšnodaļā.



2.6.att. **Īsinājumizvēlnes vaicājuma tālākai būvēšanai paredzamas funkcijas**

Vēl viena sadaļa īsinājumizvēlnē (Re-shape Query) satur divas funkcijas: šī darba ietvaros izstrādāta Add Outer Query funkcija un bakalaura darba ietvaros izstrādāta Set as Main Class funkcija, kas ļauj izmainīt vaicājuma galveno klasi, ja vaicājuma struktūra atļauj tādu darbību [58].

2.2. Saskarnes risinājumu salīdzinājums

Salīdzinot ViziQuer rīka saskarnes risinājumus ar citiem rīkiem, var atzīmēt, ka daļa risinājumu ir kopīga, piemēram, 1.2. apakšnodaļā minētas:

- Piekļūšana datu shēmas datiem, ViziQuer rīkam tā ir realizēta vairākas vietās: klases ievietošanas, Add Link un Connect Classes funkcionalitātēs, kas tika izstrādātas šī darba ietvaros un detalizēt tiek apskatīta 2.3. apakšnodaļā;
- Rīkjoslu un īsinājumizvēlnes izmantošana ērtākai funkciju pieejamībai;
- Nozīmes piešķiršana krāsai un formai, kā arī saistīto elementu apvienošana (atribūti un nosacījumi kā klases grafiskā objekta daļa);
- SPARQL vaicājumvalodā rakstītais vaicājums un iespēja pārbaudīt to rezultātus.

ViziQuer rīks piedāvā arī divu pieeju apvienošanu, ļaujot lietotājam, kas ir pazīstams ar citu rīku, veidot vaicājumu sev pierastajā veidā: zīmējot vai pievienojot elementu no automātiski

izveidota datu shēmai atbilstošā saraksta. Tāda pieeja nav unikāla un no apskatītājiem rīkiem QueryVOWL un Nitelight rīkos ir apvienotas abas metodes, tomēr biežāk ir sastapta vienas metodes izmantošana.

Ja paņemt 2019. gadā publicēto rīku salīdzinājumu, ko veicis Vargass ar līdzautoriem pētot savas komandas produktu RDF Explorer rīku [47], tad no četriem parametriem, kas tikai izmantoti, divi ir pievienoti ViziQuer rīkam pēdējo divu gadu laikā (raksta autors ir salīdzinājis ar 2018. gada ViziQuer rīka versiju):

- Pievienota priekšā teikšana (Vargasam – automātiskā pabeigšana), kas ļauj izvēlēties no iespējamo variantu saraksta vajadzīgo, saraksts pielāgojas ievadītai simbolu virknei, datu shēmai un SPARQL vaicājumvalodas atļautām konstrukcijām;
- Paraugu vai apmācības esamība (Vargasam – vaicāšana balstoties uz piemēra), kas ir pieejama vairākās vietās: veidojot projektu kā jau gatava projekta (parauga) izmantošana, aprakstu pievienošana dažām funkcijām, piemēram, īsinājumizvēlnē Explain punkts.

Paraugu esamība šajā gadījumā ir nodrošināta tikai zināmām datu shēmām, ja kāds lietotājs izvēlas savu ontoloģiju, tad tāda apmācība nav pieejama. RDF Explorer rīkam nav iespējas izmantot citu datu shēmu un datus, kas ļauj izveidot dažādu tipisku vaicājumu piemērus.

Dinamiskie rezultāti (*Dynamic Results*) ir viena no funkcionalitātēm, kas prasa rūpīgo analīzi un vēl rūpīgāku ieviešanu, jo lielām datu shēmām vai datu bāzēm tā var izsaukt problēmas ar ātrdarbību, kas tika secinātas darbojoties ar QueryVOWL rīku. Tomēr daļēji, piemēram, rīka reakcija uz ievadītas simbolu virknes izmaiņām meklēšanas vai priekšā teikšanas gadījumos, dinamiskā rīka reaģēšana tika implementēta arī ViziQuer rīkā.

Tikai netukšo rezultātu (*Non-empty Results*) piedāvāšana arī nav viennozīmīgi labs risinājums, ja lietotājs ir nozares specialists, kas spēj interpretēt tukšo rezultātu (RDF Explorer un ViziQuer rīki atšķiras ar potenciālo lietotāju definīciju). Vēl viena atšķirība starp RDF Explorer un ViziQuer rīkiem ir otrā rīka spēja strādāt ar patvaļīgo datu shēmu, kuru izvēlas lietotājs, arī ja nav pieejami dati, kas būtu neiespējams, ja tiktu realizēts netukšu rezultātu aizliegums.

Vēl viena būtiska izmaiņa, kas tika ievesta pēdējā laikā, ir iespēja uz vienas saites noteikt garāku ceļu starp klasēm. Piemēram, CPhysician un CDiagnosis klases nav tieši saistītas, bet vaicājuma ir nepieciešamas tikai tās; ievestais pieraksts padara par iespējamu tos savienot pa tiešu (2.4.att.). Šī pieeja ļauj samazināt objektu skaitu diagrammā, jo nav nepieciešamas klašu-starpnieku un atbilstošo saišu ievietošana diagrammā; arī pats vaicājums kļūst pārskatāmāks.

Analizējot saskarnes risinājumu ir secināts, ka ViziQuer rīks piedāvā citiem rīkiem līdzīgus risinājumus, tomēr nevienā no rīkiem netika apvienots tik pat plašs risinājumu skaits, ļaujot lietotājam izvēlēties sev piemērotu, kā arī novērtējot risinājuma ietekmi uz rīka darbību. Tas ir sākotnēja mērķa – bagātīgu vaicājumu definēšanas atbalsts – rezultāts, jo lietotājam jāpiedāvā instrumenti, kas ir nepieciešami vaicājumu izveidei.

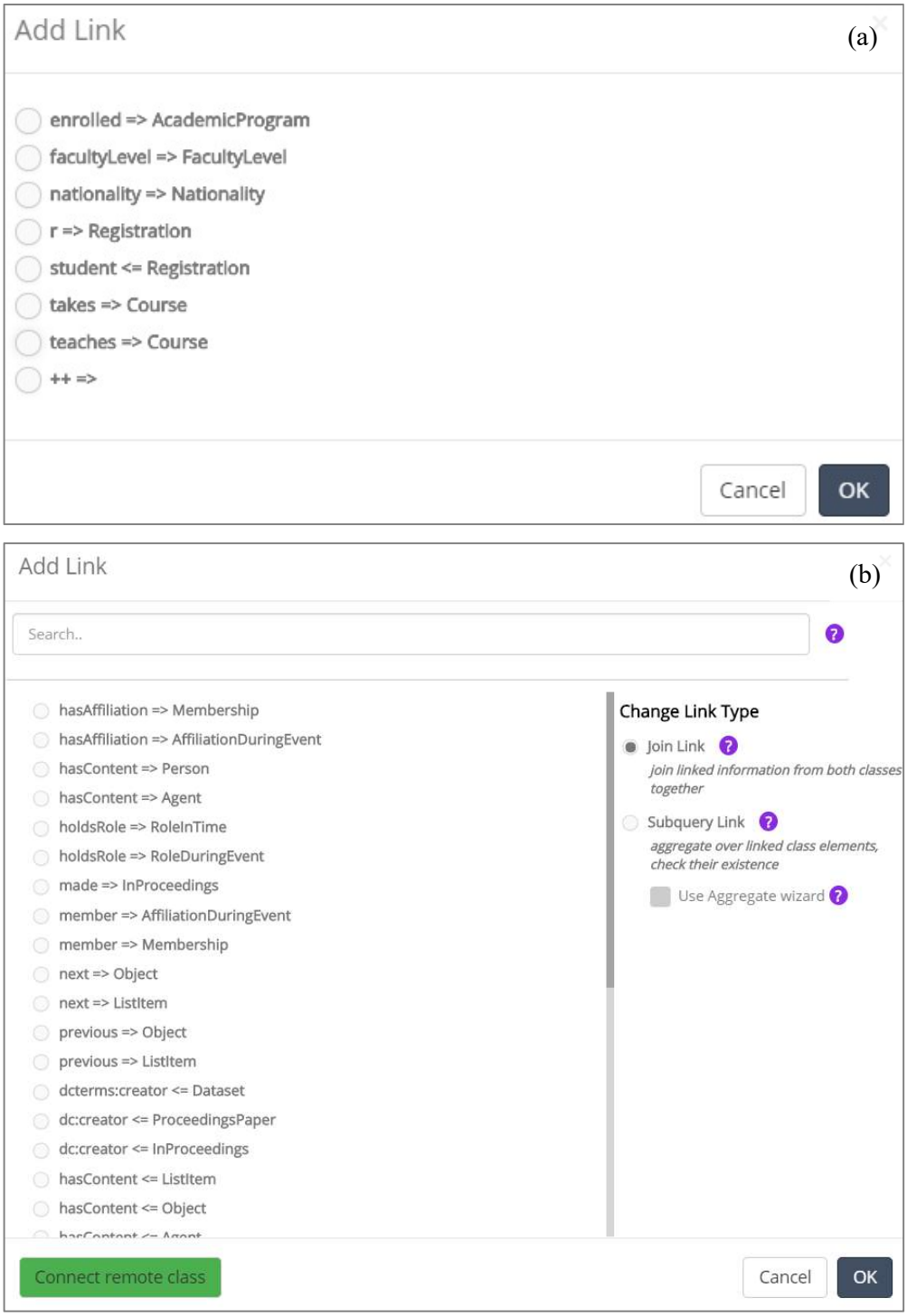
2.3. Implementēti risinājumi

Paplašinoties ViziQuer rīka lietotājiem pieejamo iespēju daudzumam arvien aktuālāks kļuva jautājums par lielām datu shēmām, saprotamu un lietotājam draudzīgo saskarni, kas ļautu piekļūt jaunām iespējām ērtākā veidā. Daži risinājumi, kas mazāk mijiedarbojas ar lietotāju, ir mazāk pakļauti šai problēmai, paceļot tikai novietošanas jautājumu; pieņemot, ka lietotājs saprot to darbību, tai nav nepieciešama informācija no lietotāja, izņemot pašu komandas izsaukšanu.

Tomēr vairākums funkcionalitāšu prasa lietotāja līdzdalību, un šī darba ietvaros uzmanība ir pievērsta funkcionalitātēm, kas tika idejas līmenī izdomātas un testa versijas līmenī realizētas bakalaura darba ietvaros, kas šī darba ietvaros tika pielāgotas jaunajiem apstākļiem, pievienojot būtiskus elementus, ieskaitot arī jaunās funkcionalitātes. Viens no tādiem piemēriem ir Aggregate Wizard funkcionalitāte, kas izradījās veiksmīgs risinājums, un tagad tā tiek izmantota arī ārpus sākotnēji plānotas vietas, papildinot funkcionalitāti un paskaidrojot lietotājiem nesaprotamākas vietas. Par tādu funkcionalitāšu nepieciešamību liecina arī veikta vizuālo vaicājumu rīku analīze, kas parāda, ka daudzos rīkos, ieskaitot pēdējā laikā izstrādātos, izmanto pakāpenisko vaicājuma definēšanu, pievienojot jau esošam objektam nākamo. Izstrādājot maģistra darbu tika sasniegta tāda funkcionalitāšu kvalitāte, ka tās tika iekļautas ViziQuer rīka publiski izplatāmajā versijā.

2.3.1. Add Link funkcionalitāte

Add Link funkcionalitāte pievieno vaicājuma klasei jaunu elementu pāri: klasi, ar kuru datu shēmā pastāv tieša saite, un pašu saiti, piešķirot abiem elementiem atbilstošus vārdus. Izmantojot šo funkcionalitāti lietotājam nav labi jāorientējas datu shēmā, jo sarakstā jau ir atlasīti tikai tādi pāri, kas atbilst tiešās saites esamības nosacījumam, tādējādi tiek nodrošināta netiešā pieeja datiem un samazināta kļūdu iespēja pareizrakstības dēļ, ka arī paātrināta



2.7.att. **Add Link funkcionalitātes loga izskats:**

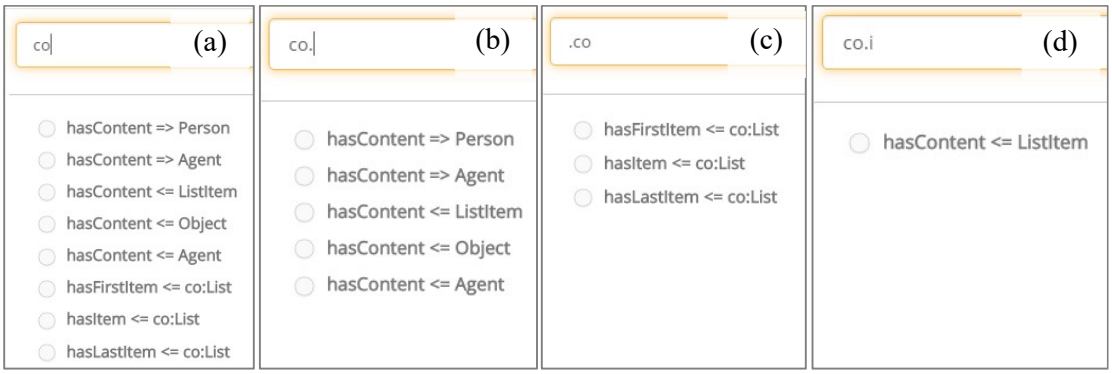
(a) pirmā versija [58], (b) tagad

vaicājuma izveide. Līdzīgo funkcionalitāti piedāvā Visual Query Builder, Optique un RDF Explorer rīki, bet Nitelight rīks nodrošina elementu (klašu un saišu) pievienošanu pa vienam.

Izstrādājot funkcionalitāti testa versijā saskarne bija ļoti vienkārša, saturot tikai pašu sarakstu (2.7.att.(a)) un 2 pogas izvēles pieņemšanai vai noraidīšanai, bet veicot pētījumus uz lielākām datu shēmām, pētot citus vizuālos rīkus un pārrunājot iespaidus izstrādātāju komandā un ar potenciāliem lietotājiem, tika apzināta nepieciešamība pilnveidot to, papildinot lietotājam piedāvāto iespēju klāstu ar dažām funkcionalitātēm. Tagad Add Link funkcijas logs satur papildus elementus:

- Meklēšanas logu;
- Saites īpašību uzstādīšanas logu;
- Connect Remote Classes pogu (2.7.att.(b)).

Connect Remote Classes pogas ieviešana atrisina situāciju, kad lietotājam ir nepieciešams savienot klases, starp kurām nav tiešās saites, bet nezinot datu shēmu tas tika apzināts tikai izsaucot Add Link funkciju; tā ļauj pa tiešo pāriet uz Connect Classes vienu no realizācijām (sk. 2.3.3. punktā).



2.8.att. **Add Link meklēšanas rezultāti atkarībā no punkta novietojuma:**
(a) meklējot starp klasēm un saitēm, (b) tikai starp saitēm, (c) tikai starp klasēm, (d) dažādu simbolu meklēšana katrā no daļām

Meklēšanas daļā atrodama forma simbolu ievadīšanai un aprakstošā loga izsaukšanai, izmantojot tipisko palīdzības apzīmējumu ar jautājuma zīmi violetā fonā. Palīdzības logs ir nepieciešams, lai paskaidrot meklēšanas īpatnību, t.i. speciālo simbolu (atstarpe, komats, punkts) nozīmi ViziQuer rīkam. Ja atstarpe un komats tiek izmantoti lai atdalīt dažādas meklējamas simbolu virknes, kā tas ir pieņemts vairākos citos rīkos, tad punktam piešķirta īpašā nozīme. Jo ar Add Link izveidotais saraksts katrā rindā satur divu veidu objektus (saite un klase), bet datu shēmās klasēm un saitēm mēdz būt līdzīgi vai pat sakrītoši vārdi, tad ir

iespējamas situācijas, kad, neskatoties uz veikto meklēšanu, rezultātu skaits joprojām ir liels un slikti pārskatāms, jo meklēta simbolu virkne sastopama abās daļās. Lai ierobežot meklēšanu tikai vienā no daļām tiek izmantots punkts:

- Ja punkta nav, tad simbolu virkne tiek meklētā gan saites, gan arī klases nosaukumos (2.8.att.(a));
- Ja meklēšanas logā ir ievietots punkts, tad pirms punkta atrodošo simbolu virknes tiek meklētas saites vārdos, bet pēc tā – klases vārdos (2.8.att.(b), (c)).
- Ja gan pirms punkta, gan arī pēc punkta ir ievadītas simbolu virknes, tad meklēšanas notiek tādā veidā, ka rezultāti satur tikai tādus pārus, kas saites vārds satur pirms punktā ievadīto simbolu virknes un klases vārdā – simbolu virknes pēc punkta (2.8.att.(d)).

Tāds risinājums bija nepieciešams, jo pētot citus rīkus tika pamanīts, ka, ja meklēšanas notiek visos pieejamos datos, reti ir novērojama iespēja izvēlēties, vai ir meklēta klase, saite vai pat instance, kas apgrūtina vaicājuma izveidi un palielina meklēšanas logā ievadīto simbolu skaitu līdz brīdim, kad saraksts kļūst pietiekami īss, lai varētu izvēlēties nepieciešamo, kas izplātītajiem vārdiem var notikt tikai ievadot visu vārdu. Tāda rakstura problēmas tika novērotas Visual SPARQL Builder, QueryVOWL un RDF Explorer rīkiem; citos rīkos risinājums ir atrasts ierobežojot laukus, kuros notiek meklēšana, piemēram, Optique rīkam, meklēšana no sākumā ir definēta tikai klasei. Izstrādājot šo saskarnes risinājumu ViziQuer rīkā tā problēma tika novērsta, bet paradījās nepieciešamība paziņot par to lietotājam, kas arī tika realizēts.

Vēl viena ar Add Link saistīta funkcionalitāte ir saites tipa konfigurēšana, kas atrodama labajā pusē (*Change Link Type*, 2.7.att.(b)). ViziQuer rīkā ir pieejamas 4 veidu saišu iestātījumi:

- Parasta saite, kas savieno divas klases;
- Neobligāta nosacījuma saite (*Optional*);
- Negācijas saite;
- Apakšvaicājuma saite (*Subquery*).

Parastas un apakšvaicājuma saites iestatījumi ir viena otro izslēdzošie, bet neobligāta nosacījuma un negācijas saites tips ir papildus parametri, kas neizslēdz neviena cita iestatījuma izmantošanu un tiek retāk izmantoti vaicājumos; negācija arī ir reti sastopama rīkos, kas apgrūtina pareizo rezultātu atrašanu [56]. Apakšvaicājumu izmantošana ir viena no ViziQuer rīka funkcionalitātēm, kas virzīta uz bagātīgo vaicājumu definēšanu, tāpēc tās izmantošanai ir pievērsta pietiekami liela uzmanība, bet tika secināts, ka ne vienmēr tā ir pareizi saprasta un korekti izmantota. Šī jautājuma risinājums detalizētāk ir apskatīts 2.3.2. punktā.

Veikto izmaiņu dēļ Add Link funkcionalitātes logs kļuva bagātāks ar elementiem, kuru mērķis ir atvieglot lietotājam vaicājuma sastādīšanu, ietaupot laiku un ievietojot diagrammā datu shēmai un SPARQL vaicājumvalodai atbilstošus elementu.

2.3.2. Aggregate Wizard funkcionalitāte

Apakšvaicājums ir tāda vaicājuma daļa, kuru izmanto esamības pārbaudei un apkopojumos (saskaitīt, sasummēt, atrast vidējo un tml.), un ir ļoti noderīga lietotājam. Tomēr pārrunas ar lietotājiem un lietotāju testi [58] rāda, ka neskatoties uz to, ka šī konstrukcija ir ļoti noderīga reālajos vaicājumos, tās izmantošana izraisa grūtības. Lai atrisināt šo jautājumu šī darba ietvaros tika realizēta papildus funkcionalitāte, kas ļauj lietotājam izmantot formu (vedni), definējot apkopojuma (*Aggregate*) parametrus. Ņemot vērā rīku specifiku, līdzīga pieeja ir sastapta SPARQLing rīkā pievienojot nosacījumu, kur lietotājam ir piedāvāta forma ar brīvām vietām lai pievienot atbilstošus parametrus (mainīgo, skaitlisko vērtību), bet tajā nav realizēts vairāku saistītu parametru apstrādes atbalsts vienā logā; pieeja arī atgādina uz filtriem balstītus rīkus, kur tā ir plaši izmantota.

Vednis sastāv no vairākiem laukiem apkopojuma izveidei:

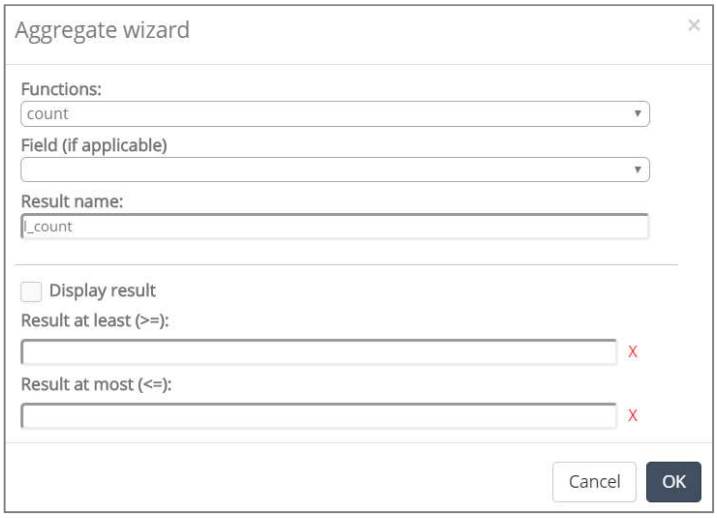
- Funkcija izvēlne, kas ļauj izvēlēties situācijai atbilstošu no zināmām funkcijām;
- Atribūts (*Field*), kuram funkcija tiks piemērota, ja tas ir iespējams;
- Mainīgā vārds, kas sākumā ir automātiski ģenerēts, bet lietotājs var ievest savu apzīmējumu;
- Funkcijas izmantošanas rezultātu pievienošana vaicājuma rezultātiem;
- Tipiskākie nosacījumi uz skaitļiem (2.9.att.).

Atribūti tiek atlasīti atbilstoši funkcijai, piemēram, atribūtu vērtību summēšana ir piemērojama tikai skaitliskām vērtībām, tāpēc izvēloties šo funkciju notiek atribūtu vērtību tipu salīdzināšana ar skaitļu tipiem.

Funkcionalitātes darbību kā soļu analīzi var veikt uz tipiska vaicājuma: atrast vārdu, uzvārdu un slimnīcas epizožu skaitu pacientiem, kam ir vismaz 5 slimnīcas epizodes. Pirmie soļi ir vienādi: ievietot klasi “Pacients” un pievienot tam atbilstošus atribūtus (vārdu, uzvārdu). Pieļaujot, ka lietotājiem ir pieejama un zināma Add Link funkcionalitāte ar apakšvaicājumu uzstādīšanas funkciju, tad tiek pievienota klase “Slimnīcas epizode”. Ja nav pieejams vednis, tad lietotājam jāizdara sekojošas darbības:

1. Pievienot agregācijas mainīgo klasei “Slimnīcas epizodes” – izsaukt logu un aizpildīt 2 laukus;
2. Pievienot nosacījumu uz agregācijas mainīgo (nosacījums “vismaz 5”) – izsaukt logu un ierakstīt tekstu, kas satur arī mainīgā vārdu;
3. Pievienot agregācijas mainīgo, lai tas tiktu attēlots – izsaukt logu un ierakstīt vārdu.

Šķietami vienkāršās darbībās tomēr ir daudz kļūdu avoti: lietotāji nepievieno agregācijas mainīgajam vārdu, nepareizajā vietā (citas klases objektā) pielieto agregāciju, attēlošanas pieprasījumu vai nosacījumu uz mainīgo; arī teksta ievadīšana ar roku palielina drukas kļūdas varbūtību.



2.9.att. **Aggregate Wizard funkcionalitātei atbilstošais vednis**

Pievienojot vedni ir:

- Pārbaudīta informācijas esamība, piemēram, mainīgā vārds;
  - ja lietotājs nepievērš uzmanību šim punktam, tad sākotnēji ir ievietots automātiski ģenerēts vārds;
- Pārbaudīta pareizrakstība, piedāvājot iepriekš ģenerēto sarakstu no funkciju vārdiem;
- Automatizēta pievienošana rādāmajām vaicājuma daļām (viena darbība - klikšķis);
- Automatizēta nosacījumu formulēšana (tikai robežvērtību ievadīšana).

Kā redzams, visi soļi, kas bez vedņa prasa vairākas darbības, ieskaitot teksta ievadīšanu, ar vedņa palīdzību prasa mazāk darbību un ir veicami vienā logā, rādot pārskatāmāku saskarni. Automātiskā parametru ievietošana ļauj novērst situāciju, kad lietotājs ievieto parametrus nepareizi, kas īpaši sākumā rīka izmantošanas posmā samazina kļūdu skaitu un parāda lietotājam pareizo vaicājuma parametru novietojumu.

Vairāku darbību automatizēšana atviegloja lietotājam agregācijas veidošanu un ļāva ātrāk sasniegt pareizo rezultātu. Pēc vedņa ieviešanas un testēšanas, risinājums tika pievienots visiem atbilstošajiem logiem un ir pieejams izsaucot apkopojuma veidošanas logu no klasei atbilstošās formas labajā pusē vai īsinājumizvēlnē (2.5.att.), ka arī no Add Link loga kā ieteiktais veids kā pareizi izveidot apkopojumu.

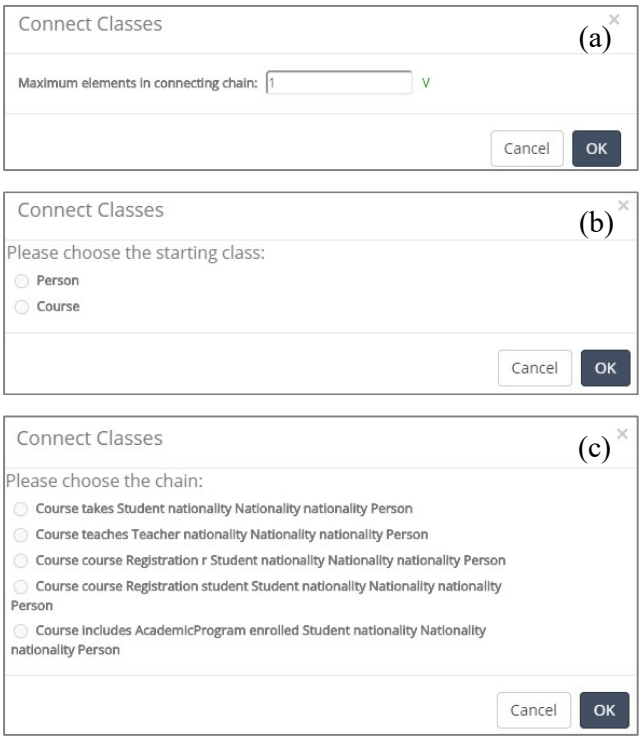
2.3.3. Connect Classes funkcionalitāte

Connect Classes funkcionalitāte ir loģisks turpinājums iepriekš apskatītajai Add Link funkcionalitātei, jo ļauj savienot klases, ja datu shēmā starp tiem ir patvaļīgo citu saišu un klašu skaits, tāpēc šī funkcionalitāte "manto" tādu labu īpašību kā netiešā pieeja datu shēmai, paplašinot no datu shēmas pieejamo apkārtni un nodrošinot lietotājam ātrāku pareiza risinājuma atrašanu. Sakontēji funkcionalitāte tika piedāvāta īsinājumizvēlnē vairāku diagrammas elementu izlasei, kurā ir tieši 2 klases, šī darba ietvaros tā tika pārveidotā tā, lai būtu izsaucama no īsinājumizvēlnes trijos gadījumos, ļaujot izmantot to priekšrocības saišu definēšanā.

Pirmā versija sastāvēja no 3 soļiem, kas tika realizēti kā secīgi radīti atsevišķi logi (2.10.att.). Jo garākas ķēdes ir retāk satopamas vaicājumos, tad lietotājiem tika piedāvāts izvēlēties maksimālo garumu (2.10.att.(a)); pēc tam vaicājuma virzienu, jo no izlases nevar precīzi pateikt, kāds virziens ir nepieciešams (2.10.att.(b)); pēdējā solī lietotājam piedāvā variantus, kas nesatur ciklus, t.i. katra klase var tikt sastapta ķēdē tikai vienu reizi (2.10.att.(c)). Šis risinājums sākumā tiek uzskatīts par optimālo, jo ļauj veikt secīgas darbības un kontrolēt pareizus ievadus, bet vēlāk tāda pieeja izradījās nepietiekami elastīga, lai ļautu lietotājiem mainīt savu izvēli bez visu soļu atkārtošanas. Šo situāciju varētu atrisināt divos veidos: ļaujot lietotājam atgriezties atpakaļ, pie iepriekšējā loga, vai apvienojot tos.

Attīstot funkcionalitāti tika pieņemts lēmums apvienot visus logus vienā, lai lietotājiem ir pieejami visi iestatījumi (2.11.att.(a)), ļaujot pielāgot tos savām vajadzībām, ja sākumā izvēlēti parametri neatbilst uzdevumam. Otrais nepieciešamais elements, kas tika pievienots logam, ir meklēšanas logs, kas Connect Classes funkcionalitātes gadījumā ir ne mazāk svarīgs, salīdzinot ar Add Link funkcionalitāti, jo pat vidēja izmēra datu shēmām piedāvāto ķēžu skaits ir liels. Abiem elementiem tika izvēlēts novietojums virs iespējamo variantu saraksta, lai nodrošināt to, ka lietotāja uzmanība ir pievērsta parametru uzstādīšanai (izstrādāta arī brīdinājumu sistēma, ja tomēr lietotājs nav izvēlējies kādu parametru). Tāds risinājums aizņem vērā ņemamu loga daļu, tāpēc tika piedāvāta iespēja paslēpt iestatījumu daļu, palielinot sarakstam pieejamo vietu (2.11.att.(b)). Šajā versijā tika atrisināts jautājums par ķēžu lasāmību,

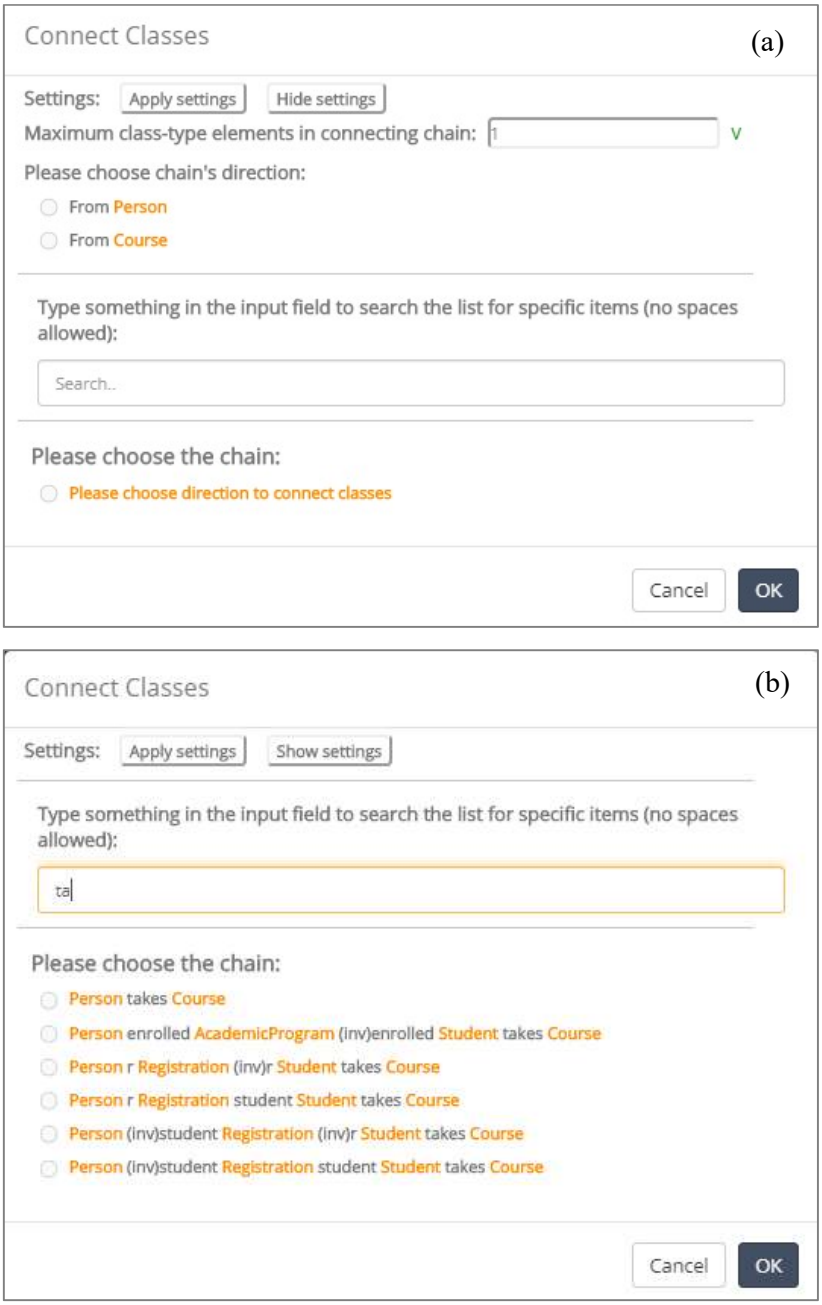
kad saites un klašu vārdi saplūst, kas tika pamanīts izstrādājot bakalaura darbu [58], izceļot klases ar krāsu, kas būtiski atviegloja uztveri; tika mēģināti citi izteiksmes līdzekļi (treknraksts, speciāli simboli), bet to efektivitāte bija mazāka.



2.10.att. **Connect Classes funkcionalitātes sākotnējās realizācijas saskarne [58]:**
(a) maksimālā ķēdes garuma noteikšana, (b) virziena noteikšana, (c) piedāvāto variantu saraksts

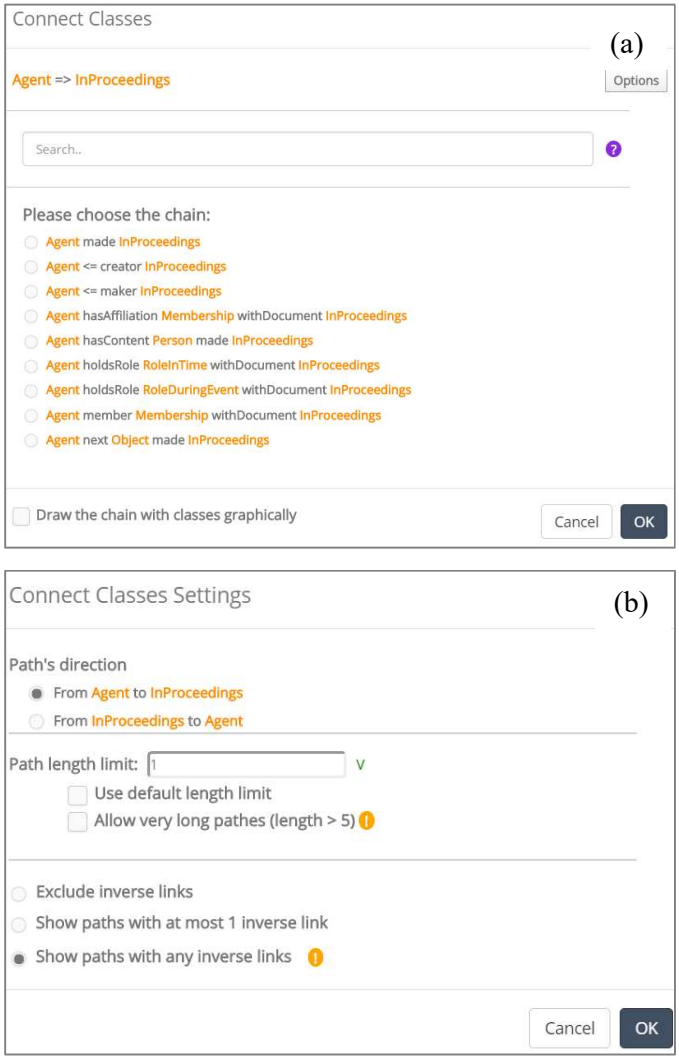
Attīstot rīku Connect Classes funkcionalitātei tika piedāvāta trešā funkcionalitātes versija (2.12.att.), tās pieejamo iestatījumu skaits ir pieaudzis, ieviešot papildus sekojošas iespējas:

- Ķēdes virziena noteikšana – lietotājs izvēlas to, kurā virzienā tiek būvēta ķēde;
- Maksimālā ķēdes garuma noteikšana ar apakšpunktiem
  - Logs, kurā var ievadīt skaitlisko vērtību;
  - Izvēles rūtiņa pēc noklusēšanas vērtības uzstādīšanai (balstoties uz tipiskākiem vaicājumiem – sk. 1.3. apakšnodaļā – tikai izvēlēts skaitlis 3);
  - Izvēlnes rūtiņa lai atļautu garus ceļus ar brīdinājumu, ka tas var ietekmēt rīka ātrdarbību;
- Inverso jeb apgriezto saišu skaita noteikšana:
  - Neradīt inverso saiti saturošas ķēdes;
  - Rādīt tikai ķēdes, kurās ir ne vairāk par vienu inverso saiti;
  - Radīt visas ķēdes.



2.11.att. **Connect Classes funkcionalitātes otrās realizācijas izskats [79]:**
(a) pie izsaukšanas, (b) paslēpjot parametru daļu un veicot meklēšanu

Maksimāla ceļa garuma ierobežojums tika pievienots palielinoties izmantojamu ontoloģiju izmēram, jo visu iespējamo ceļu starp diviem punktiem grafā atrašana (kas pēc būtības ir Connect Classes funkcionalitāte) prasa datora resursus un tādējādi ietekmē ViziQuer rīka ātrdarbību. Tāpēc bija nepieciešams pievērst lietotāja uzmanību šim faktam un pieprasīt rūpīgi

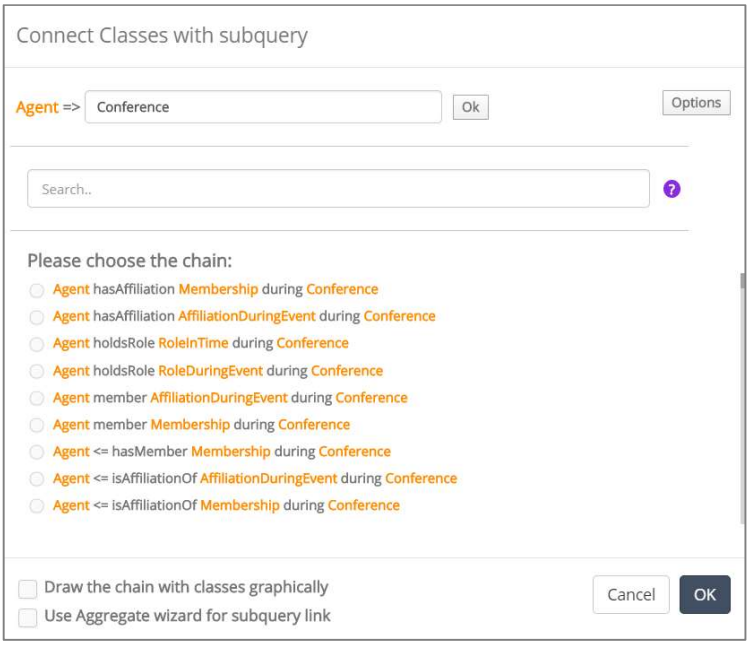


2.12.att. **Connect Classes funkcionalitātes saskarnes aktuālās versijas izskats:**
(a) funkcionalitātes, (b) iestatījumu logu

novērtēt nepieciešamību meklēt tik garus ceļus. Inverso saišu radīšanas ierobežojums ir saistīts ar to, ka ir iespējami gadījumi, kad ir paslēptas cilpas (apakšklašu un virsklašu attiecības dēļ) vai bezjēdzīgas daļas, kas neienes vaicājumā jauno informāciju, kas ir raksturīgi, ja inversai saitei ir kardinalitāte 1, kā arī grūti interpretējamas daļas. Tomēr pašas inversās saites ir

noderīgas un pilnīgā to atslēgšana būtiski ierobežotu vaicājumu definēšanas iespējas, tāpēc rīks paļaujas uz lietotāja izvēli; pie palaišanas rīks rāda visas iespējamas ķēdes.

Veikto izmaiņu rezultātā kopējais iestatījumu daļas izmērs kļuva pārāk liels lai savienot to ar ķēdēm. Šī saskarnes problēma tika atrisināt, piedāvājot tādu funkcionalitātes versiju, kas saglabāja iepriekšējas versijas labas īpašības (meklēšanas logu, krāsas izmantošanu klašu izcelšanai) (2.12.att.(a)), bet iestatījumus pārnesa atsevišķajā logā (2.12.att.(b)), kas ir pieejams no Connect Classes galvenā loga ar pogas palīdzību. Jo šajā risinājumā ķēdes būvēšanas virziens ir pamanāms tikai sarakstā un ir maināms iestatījumos, tad no iestatījumiem atbrīvotajā vietā pievienoja informāciju par šo parametru.



2.13.att. **Connect Classes funkcionalitātes logs, kas izsaucams no Add Link loga, ja ir izvēlēts apakšvaicājuma saites tips**

Atšķirībā no sākotnējas versijas, šo funkcionalitātes versiju pievienoja īsinājumizvēlnēm divos gadījumos: ja ir elementu izlase ar tieši divām klasēm, kā tas bija sākumā, un saites starp divām klasēm gadījumā. Pirmais gadījums ļauj bez saites zīmēšanas savienot diagrammai pievienotas klases, samazinot darbību skaitu, kuru nepieciešams veikt lietotājam. Otrajā gadījumā funkcionalitāte kalpo vārda piešķiršanai patvaļīga garuma saitei vai pilnā ceļa zīmēšanai, ievietojot diagrammā nepieciešamus klašu un saišu objektus un piešķirot tiem atbilstošus vārdus.

Pēc saīsināta ceļa pieraksta (*property path*) ieviešanas kļuva iespējams piešķirt saitei vārdu ne tikai tieši savienoto klašu gadījumos, bet arī ja starp klasēm nav definēta saite, atrodot

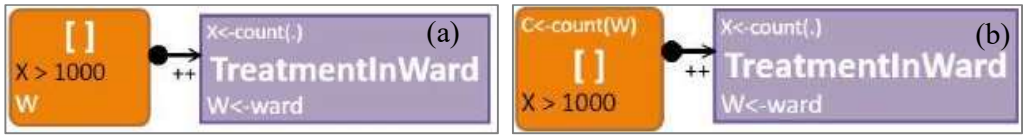
ceļu datu shēmā. Connect Classes funkcionalitātei tika veiktas izmaiņas, lai arī tā atbalstītu šo pierakstu, pievienojot saskarnei papildus parametru (Draw the chain with classes graphically izvēles rūtiņa (2.12.att.)), kas ļauj izdarīt izvēli starp visu objektu ievietošanu diagrammā un īso pierakstu. Šī izvēle ir nepieciešama, jo vaicājumos var būt atribūti un/vai nosacījumi tikai uz gala klasēm, kad vairāku neizmantotu objektu ievietošana diagrammā apgrūtina vaicājuma uztveri; bet ja vaicājumā ir nosacījumi, kas attiecās uz klasēm ceļa vidū, tad ir nepieciešams ievietot atbilstošo klasi diagrammā. Izvēles rūtiņa ļauj ievietot saskarnē (īsinājumizvēlnē) tikai vienu punktu, un vienlaikus atrisinot abus gadījumus un ļaujot lietotājam vieglāk atcerēties pareizas darbības, kas ir nepieciešamas rezultātu sasniegšanai.

Modificētā veidā Connect Classes funkcionalitāte ir izsaucama no Add Link loga ar Connect Remote Classes pogu, galvenā atšķirība ir nepieciešamība izvēlēties klasi, ar kuru izveidot saiti (2.13.att.). Šai Connect Classes funkcionalitātes versijai nav iespējams mainīt saites virzienu, šo īpatnību, kā arī iespēju izveidot apakšvaicājumu un īsā pieraksta gadījumā izmantot Aggregate Wizard vedni, tā manto no Add Link funkcionalitātes, t.i. ja Add Link logā lietotājs atzīmējis saites tipu kā apakšvaicājumu, tad šī izvēle ir saglabāta. Vedņa izsaukšana notiek tikai pie īsa pieraksta, lai novērstu gadījumus, kad lietotājs mēģinātu izveidot apakšvaicājumu ķēdes vidū.

Connect Classes funkcionalitāte tādā realizācijā ir unikāla, vismaz apskatīto rīku vidū neviens nepiedāvā veidot vaicājumu uzreiz ievietojot datu shēmai atbilstošu ķēdi, kaut gan strādājot pie tās liekas, ka tā ir acīm redzams nākamais solis, ko rīka izstrādātāji var piedāvāt lietotājiem. Darbojoties ar uzdevumiem iespēja noteikt ceļu starp klasēm bez ontoloģijas pētīšanas atvieglo vaicājuma izveidi, būtiski ietaupot laiku tādos gadījumos, kad starp klasēm minimālais ceļa garums ir 3-4 saites, bet uzdevumā nav definēti ceļa posmi. Izmantojot maksimālā garuma ierobežošanu, lietotājs spēj regulēt piedāvāto rezultātu skaitu, kas var sekmēt arī izpratni, vai meklēta klase atbilst uzdevumam, t.i. vai lietotājs ir izvēlējies pareizo klases nosaukumu. Tādā veidā tiek atrisināts jautājums par nepieciešamību labi pārzināt vai izpētīt datu shēmu, kā arī samazināts kļūdu skaits nepareizi ievadīto nosaukumu dēļ.

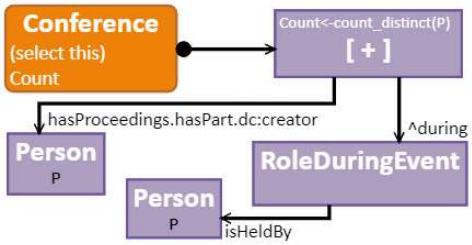
2.3.4. Add Outer Query un Add Union funkcionalitātes

Add Outer Query un Add Union funkcionalitātes ir līdzīgas pēc savas darbības tāpēc to raksturojumi ir apvienoti vienā punktā. Tās ir paredzētas speciālā klases tipa mezgla pievienošanai, un tās prasa tikai pašu atbilstošās komandas izsaukšanu. Tāpēc lietotāja saskarnes ziņā abas funkcionalitātes ir pārstāvētas ka īsinājumizvēlnes apakšpunkts.



2.14.att. **Add Outer Query funkcionalitāti izmantojošo vaicājumu piemēri [75]**

Add Outer Query funkcionalitāte ir būtisks elements lai atbalstītu bagātīgu vaicājumu izveidi un padarītu vaicājumu definēšanu ērtāku un no lietotāja darbību skatu punkta loģiskāku, jo lietotājiem nav dabiski sākt vaicājumu izveidi ar vadības klasi. Vadības klase neieiet datu shēmā un tai ir vairāk palīgierīces loma, kas ļauj izveidot noteiktas formas vaicājumus. Piemēram, izveidot sarakstu no slimnīcas nodaļām, kurās ir veiktas vairāk par 1000 medicīnisko aprūpju darbības, (2.14.att.(a)) vai saskaitīt tādas nodaļas (2.14.att.(b)). Abos gadījumos lietotājiem lasot uzdevumu pirmajā brīdi ir informācija tikai par vienu klasi, kuru, ja viņam nav pieredzes ar tāda tipa vaicājumiem, arī pievienos diagrammai. Izmantojot Add Outer Query funkcionalitāti lietotājam ir dots instruments, kas ļauj pievienot vadības klasi ar vienas funkcionalitātes izsaukšanu.



2.15.att. **Add Union funkcionalitāti izmantojošo vaicājumu piemēri [8]**

Add Union funkcionalitāte kalpo citas vadības klases pievienošanai, kas ir paredzēta nosacījumu apvienošanai ar loģisko operatoru VAI. Tāda uzdevuma piemērs ir sekojošs: saskaitīt katras konferences dalībniekus, kuri ir rakstu autori vai piedalījās konferences darbības organizēšanā. Atbilstošajā vaicājumā ir nepieciešams izmantot speciālo klasi "[+]", lai korekti pievienot vaicājumam abus nosacījumus (2.15.att.).

Abas izstrādātas funkcionalitātes neskatoties uz no lietotāja saskarnes skatu punkta vienkāršu realizāciju ir svarīgas, jo atvieglot lietotājam tiešo piekļuvi pie bagātīgā vaicājuma izveides līdzekļiem – vadības klasēm. Funkcionalitāte pēc darbības principa atgādina Add Link funkcionalitāti, tomēr ir nepieciešama kā atsevišķa, jo pievieno diagrammai pēc klasifikācijas cita tipa elementu.

2.4. Implementēto risinājumu novērtējums

Uz doto brīdī ViziQuer rīks ir viens no nedaudziem pēdējā laika izstrādātajiem vai pilnveidotajiem rīkiem, kas darbojas ar patvaļīgo datu shēmu un atbalsta plašu SPARQL vaicājumvalodas konstrukcijas, ļaujot definēt bagātīgus vaicājumus. Lai saglabāt rīka pozīciju ir nepieciešams sekot līdzi rīku izstrādes tendencēm, pievienot jaunas iespējas un modificēts jau esošas.

Viens no veidiem, kā iegūt rīka attīstībai nepieciešamu informāciju ir lietotāju testi, un tādi pētījumi tika veikti arī ViziQuer rīkam. Pirmajos pētījumos (2017. gadā) tika noskaidrots, vai ViziQuer rīks ļauj lietotājiem izveidot vaicājumus efektīvāk par SPARQL vaicājumvalodu, lietotāji bija studenti, kas pārstāv divas nozares: IT un medicīnas. No iegūtajiem rezultātiem tika secināts, ka rīks uzlabo rezultātus, ļaujot atrisināt vairāk uzdevumus pareizi [80].

No šī darba skatu punkta interesantāk ir apskatīt divus analoģiskus novērtēšanas testus, kas ir veikti ar viena gada starpību: 2018. gadā [81] un 2019. gadā [8], jo starp šiem testiem ir atrodam tas brīdis, kad tika pievienota Add Link un Aggregate Wizard funkcionalitātes, pie tam nav veikta citu būtisku funkcionalitāšu pievienošana, kas ļauj apgalvot, ka rezultāti lielā mērā atspoguļo tieši šo funkcionalitāšu ietekmi uz lietotāju pieredzi darbojoties ar ViziQuer rīku. Lietotājiem tika nodrošināta apmācība par rika iespējām un to, kā tas tiek darbināts un kā definē vaicājumus, piedāvājot arī piemērus, pēc apmācības tika piedāvāts vienāds vaicājumu komplekts un ierobežots laiks (70 minūtēs) to risināšanai ar ViziQuer rīka palīdzību.

Lietotāju testu rezultāti rāda, ka lietotāju rezultāti uzlabojas no vidēji 5.27 korektiem vaicājumiem uz vidēji 7.5 korektiem vaicājumiem. Aggregate Wizard vedņa ietekme ir atkarīga no vaicājuma sarežģītības, piemēram, sarežģītākam uzdevumam tas ļāvis pacelt korekti atrisināto vaicājumu ar agregācijām skaitu no 25% (2 no 8) uz 45% (9 no 20), kas ir vērā ņemams uzlabojums, bet tomēr vēl nav labs rezultāts, jo puse no lietotājiem nespēja izveidot korekto vaicājumu; mazāk sarežģītajiem vaicājumiem rezultāti bija pārliecinošāki, sasniedzot pat 100%, kas uzdevumu korekti atrisināja visi lietotāji [8].

Balstoties uz šiem rezultātiem var apgalvot, ka Add Link funkcionalitāte būtiski palielinā korekti atrisināto vaicājumu skaitu un paātrina vaicājumu definēšanu, bet Aggregate Wizard funkcionalitāte uzlabo rezultātus vaicājumiem, kas satur esamības pārbaudes vai apkopojumus.

2.1. tabula

Sagatavotu vaicājumu komplekti un testēšanas mērķa apraksts

Npk	*1. komplekts*	*2. komplekts*	*Apraksts*
1	Atrodiet 5 dārgākas slimnīcas epizodes un kā sauca pacientu.	Atrodiet 10 dārgāko poliklīnikas epizožu (OutpatientEpisode) ilgumu, izmaksas un atbilstošo diagnozi.	Triviāls variants Add Link funkcionalitātes izmantošanai. Vaicājumā minētas abas klases.
2	Atrodiet, cik daudz ir slimnīcas epizodes, kad vīrieši pavadīja slimnīcā vismaz 2 nedēļas.	Saskaitiet visas slimnīcas epizodes kungam ar uzvārdu Romero, kuras maksāja vismaz 500 naudas vienības.	Nedaudz komplicētāks vaicājums, klases ir minētas netiešā veidā.
3	Atrodiet tādu pacientu uzvārdus un personas kodus, kas tika ievietoti slimnīcā vairāk par 3 reizēm, un atbilstošu slimnīcas epizožu skaitu.	Atrodiet nodaļas (ward) un tām atbilstošu slimnīcas epizožu skaitu.	Add Link un Aggregate Wizard funkcionalitātes.
4	Atrodiet 10 visilgākās manipulācijas, kurā nodaļā (ward) tās notika un kā sauca ārstu (attendingPhysician).	Atrodiet visas reizes, kad kāds ārsts (norādot to vārdu un uzvārdu) poliklīnikā (OutpatientEpisode) uzstādīja diagnozi "Adenoīdu hipertrofija" un saņēma mazāk par 11 naudas vienībām, norādot samaksāto summu.	Divas klases, kas savienotas ar ceļu, kurā ir 2 saites (viena starpklase), kur nepieciešams zīmēt visus elementus.
5	Saskaitiet, cik daudz dažādas diagnozes izrakstīja ārsts, kad bija nosūtošais ārsts (referringPhysician), norādot ārsta vārdu, uzvārdu un diagnožu skaitu.	Saskaitiet pacientus, kuri ārstējas nodaļā (ward) 1-5.	1 klase pa vidu, kurai nav nosacījumu, bet ir nepieciešama agregācija.

2020. gadā novērtēšanas tests tika atkārtots ar 15 studentiem, lietotāju rezultāti ir sasnieguši vidēji 8.5 korektu vaicājumu lielumu (K.Čerāna veikts testa rezultātu apkopojums), bet šajā laikā bez iepriekš minētajām izmaiņām ir arī veikti rīka uzlabojumi, kas nav šī darba

autores izstrādāti, bet uz vieglāku vaicājumu definēšanu vērsti, tāpēc nevar viennozīmīgi noteikt, cik lielā mērā rezultātus ietekmē tieši šī darba ietvaros veiktas rīka izmaiņas.

Kā var secināt no veiktajiem testiem, tie ir paredzēti ViziQuer rīka testēšanai kā vienas veselas vienības un neļauj viennozīmīgi noteikt kādas funkcionalitātes iespaidu uz lietotāja spējām atrisināt uzdevumus un izveidot korektus vaicājumus. Lai novērtēt tieši izstrādāto funkcionalitāšu ietekmi uz lietotāja darbībām tika sagatavots vaicājumu komplekts, kas tika paredzēts maģistra studiju līmeņa studentiem ar mērķi salīdzināt viņu kā lietotāju panākumus ar un bez izstrādātām funkcionalitātēm. Katrs no vaicājumiem atbilst noteiktai situācijai, kad pēc šī darba autores domām ir nepieciešams izmantot kādu no izstrādātām funkcionalitātēm (2.1.tab.). Bez rezultātu apliecinošas informācijas (rezultātu skaits un pirmie 5 rezultāti, ja vaicājums tādus paredz) no studentiem ir prasīts saglabāt laiku, kas tika patērēts katra uzdevuma risināšanai atsevišķi, kas ļauj novērtēt ieguvumu ne tikai statistiski salīdzinot korektu vaicājumu skaitu, bet arī risināšanas ātrumu katrai no funkcionalitātēm.

2.2. tabula

Add Link funkcionalitātes novērtēšanas testēšanai sagatavota vaicājuma darbību analīze

Npk.	*Bez funkcionalitātēm*		*Add Link funkcionalitāte*	
	Apraksts	*Skaits*	*Apraksts*	*Skaits*
Atrodiet 5 dārgākas slimnīcas epizodes un kā sauca pacientu.				
1	Ievietot slimnīcas epizodes klasi			
2	Uzstādīt kārtošanu			
3	Uzstādīt izmaksas rādīšanu			
4	Uzstādīt ierobežojumu uz ierakstu skaitu			
5	Ievietot pacientu klasi	3	Izsaukt funkcionalitāti	2
6	Savienot klases	3	Izvēlēties pacienta klasi no saraksta	2
7	Uzstādīt (uz)vārda rādīšanu			
Kopā		6	Kopā	4

Testa uzdevumiem ir pievienota neliela anketa, kur studentiem tika dota iespēja pašiem novērtēt darbu ar ViziQuer rīku, pievēršot uzmanību testējamām funkcionalitātēm un saviem iespaidiem no ar tām veiktajām darbībām. Lai veiktu korekto testēšanu ViziQuer rīkam tika sagatavota speciālā konfigurācija, kas no aktuālas tiešsaistes versijas atšķiras ar to, ka ir

atslēgtas tieši Add Link un Connect Classes funkcionalitātes. Konfigurācija un vaicājumi ir notestēti, pārbaudot to darbību un korekta ne-tukša risinājuma esamību izveidotajiem vaicājumiem.

2.3. tabula

Add Link un Aggregate Wizard funkcionalitāšu novērtēšanai sagatavota vaicājuma darbību analīze

Npk.	*Bez funkcionalitātēm*		*Add Link funkcionalitāte*		*Aggregate Wizard funkcionalitāte*	
	Apraksts	*Skaits*	*Apraksts*	*Skaits*	*Apraksts*	*Skaits*
Atrodiet tādus pacientu uzvārdus un personas kodus, kas tika hospitalizēti vairāk par 3 reizēm, un atbilstošu slimnīcas epizožu skaitu.						
1	Ievietot pacienta klasi					
2	Uzstādīt pacienta klases atribūtu rādīšanu					
3	Ievietot otro klasi	3	Izsaukt funkcionalitāti	2	Add Link darbības	
4	Savienot klases	3	Izvēlēties pacienta klasi no saraksta	2		
5	Uzstādīt saites tipu	3	Uzstādīt saites tipu	1		
6	Ierakstīt apkopojumu (simbolu ievadīšana kā viena darbība)			5	Izsaukt funkcionalitāti	1
7	Izveidot nosacījumu			5	Atzīmēt rādīšanu	1
8	Uzstādīt apkopojuma rādīšanu			3	Izveidot nosacījumu	1
9	-				Apstiprināt darbības	1
Kopā		21	Kopā	18	Kopā	9

Neskatoties uz sagatavotu lietotāju testu tas netika piedāvāts studentiem sakarā ar valstī esošo situāciju un atbilstoši neiespējamo lietotāju apmācību un ViziQuer rīka novērtēšanas testēšanu klātienē. Izstrādāto funkcionalitāšu testēšana ir viena no tuvākā laika prioritātēm, un, balstoties uz rezultātiem un lietotāju atsauksmēm, tiks pieņemts lēmums par tālāko izstrādāto funkcionalitāšu pilnveidošanas nepieciešamību.

Tomēr tika veikta vaicājumu detalizēta analīze uz darbības skaitu, kas ir jāveic lietotājam, lai sasniegt korektu rezultātu, tālāk tiek apskatīti trīs no tiem (no 1., 3., un 5. kategorijām, kas

atbilst Npk vērtībai 2.1.tabulā). Vaicājuma izvēle ir noteikta no tā mērķa un paredzamo funkcionalitāšu izmantošanas skaita, lai pārstāvēt visas izstrādātas funkcionalitātes. Veicot aprēķinus tika veikti daži pieņēmumi, kas ir aprakstīti tālāk.

2.4. tabula

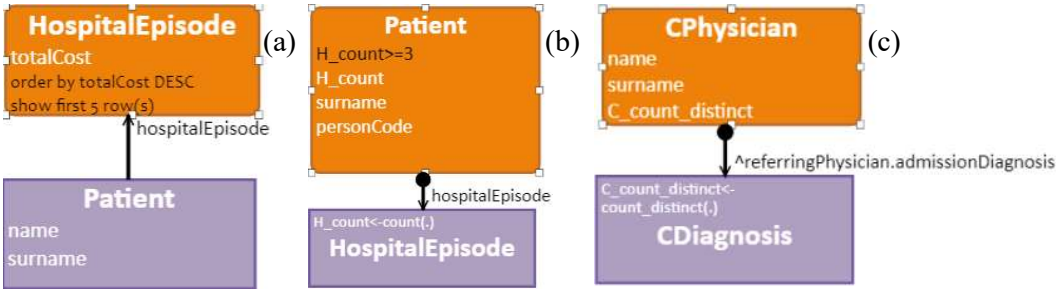
Connect Classes un Aggregate Wizard funkcionalitāšu novērtēšanai sagatavota vaicājuma darbību analīze

Npk.	*Bez funkcionalitātēm*		*Connect Classes*		*Aggregate Wizard*	
	Apraksts	*Skaits*	*Apraksts*	*Skaits*	*Apraksts*	*Skaits*
Saskaitiet, cik daudz dažādas diagnozes izrakstīja ārsts, kad bija nosūtošais ārsts (referringPsysician), norādot ārsta vārdu, uzvārdu un diagnožu skaitu.						
1	Ievietot ārsta klasi					
2	Uzstādīt ārsta klases atribūtu rādīšanu					
3	Ievietot otro klasi	3	Ievietot otro klasi	3	Izsaukt Add Link funkcionalitāti	2
4	Savienot klases	3	Izvēlēties abas klases	1	Uzstādīt saites tipu	1
5	Uzstādīt saites vārdu	5	Izsaukt funkcionalitāti	2	Izsaukt Connect Classes funkcionalitāti	1
6	Uzstādīt saites tipu	3	Izvēlēties saiti	2	Izvēlēties saiti	2
7	Uzstādīt apkopojumu			5	Izsaukt funkcionalitāti	1
8	Uzstādīt apkopojuma rādīšanu			5	Atzīmēt rādīšanu	1
9	-				Apstiprināt darbības	1
Kopā		24	Kopā	18	Kopā	9

Vienādi soļi, kas sakrīt visiem vaicājumiem, netiek analizēts; teksta ierakstīšana tika skaitīta kā 1 darbība, kaut gan faktiski lietotājs ievada vairākus simbolus, bet pat tāds tuvinājums ļauj novērtēt ātrdarbību. Visās analīzēs tiek pieņemts, ka lietotājs pārzin ontoloģiju un ViziQuer rīku, t.i. nav veiktas papildus darbības, meklējot pareizu risināšanas cēlu; nav ņemts vērā, ka lietotājs kustina pēli vai pārvieto objektus sev ērtākā pozīcijā. Vēl viens pieņēmums ir tas, ka saites nosaukums tiek piedāvāts automātiski – šī funkcionalitāte ir ievesta

nesen, savienojot tieši saistītas klases ar datu shēmā definēto saiti, bet nav šī darba ietvaros izstrādāta.

Korektie vaicājumi vizuālajā formā visos trīs detalizēti apskatītajos gadījumos sastāv no 2 klasēm un vienas saites, kur klasēm ir pievienoti vaicājumā minētie atribūti un nosacījumi (2.16.att.). Vieglākais vaicājums, kura mērķis ir noteikt Add Link funkcionalitātes ietekmi uz lietotājiem, soļu skaits ir vienāds, bet pašu darbību skaits ir 6 bez funkcionalitātes un 4 ar funkcionalitāti (2.2.tab.). Tā nav lielā starpība absolūtajās vienībās, tomēr ja skaitīt relatīvajās vienībās, tad ieguvums ir viena trešdaļa (par 100% pieņemot bez funkcionalitātēm iegūto vērtību). Ja vaicājums ir paredzēts Add Link un Aggregate Wizards funkcionalitāšu analīzei, tad rezultāti situācija bez Aggregate Wizard ir līdzīga – 21 darbība bez funkcionalitātēm un 18 ar Add Link funkcionalitāti, kas ir gandrīz 15% ieguvums. Aggregate Wizard funkcionalitātes pievienošana būtiski samazina darbību skaitu, sasniedzot tikai 9 darbību skaitu – aptuveni 57% ieguvums (2.3.tab.). Kā trešais piemērs tika apskatīts vaicājums, kurā ir vairāku saišu attālumā atrodamas klases, šajā gadījumā ierobežojoties ar divu saišu attālumu, un apkopojums; arī šeit pašas funkcionalitāte klašu zīmēšanai (Connect Classes funkcionalitāte, kas šajā gadījumā palaista no klašu izlases) neienes lielu darbību skaita ieguvumu, pazeminot to no 24 līdz 18 (25% uzlabojums), bet Aggregate Wizard ļāvis sasniegt 9 darbību skaitu – aptuveni par 63% mazāks darbību skaits (2.4.tab.).



2.16.att. **Korekti definēto vaicājumu izskats tabulās analizētajiem vaicājumiem:**
(a) 2.2.tab., (b) 2.3.tab., (c) 2.4.tab.

Veicot visu izstrādāto vaicājumu analīzi analoģiski jau aprakstītajiem piemēriem, tika iegūti skaitļi, kas ir apkopoti tabulā (2.5.tab.). Vaicājumiem, kas paredzami tikai Add Link funkcionalitātes novērtējumam ir vienādi dažādu darbību skaiti; Connect Classes funkcionalitātei vērtības ir vairāk izkliedētas. Ja vaicājums satur apkopojumu, tad Add link un Connect Classes rāda mazāku uzlabojumu, bet pieslēdzot Aggregate Wizard funkcionalitāti, darbību skaits būtiski samazinās, noteiktiem vaicājumiem sasniedzot pat 40% lielumu salīdzinot ar sākotnējo vērtību.

Balstoties uz šiem rezultātiem ir izdarāms sekojošs secinājums: izstrādātas funkcionalitātes uzlabo noteikta veida vaicājumu definēšanu, un lielākais uzlabojums novērojams tad, ja uzdevums atbilst apkopojumam un, atbilstoši, Aggregate Wizard izmantošanai.

2.5. tabula

Lietotāju novērtēšanai sagatavota vaicājuma darbību analīzes apkopojums
Apzīmējumi: Vaicājumu numuri atbilst 2.1. tab. pēc formas komplekts.Npk

Funkcionalitātes	1.1	2.1	1.2	2.2	1.3	2.3	1.4	2.4	1.5	2.5
Atslēgtas	6	6	6	6	21	19	21	25	19	24
Add Link	4	4	4	4	18	15	-	-	-	-
Connect Classes	-	-	-	-	-	-	14	18	18	18
Aggregate Wizard	-	-	-	-	9	12	-	-	15	9
Ietaupītas darbības, %										
Add Link	33	33	33	33	14	21	-	-	-	-
Connect Classes	-	-	-	-	-	-	33	28	5	25
Aggregate Wizard	-	-	-	-	57	37	-	-	21	63

Analizējot veiktus aprēķinus jāņem vērā vairākus iepriekš minētus pieņēmumus un dažus apsvērumus, kas nav atspoguļoti veiktajā vaicājumu analīzē:

- Add Link un Connect Classes funkcionalitātes kalpo ne tikai darbību samazināšanai, bet arī ontoloģijas netiešajai rādīšanai lietotājam, samazinot nepieciešamību meklēt klasi un saiti, kas atbilst datu shēmai un vaicājuma uzdevuma formulējumam;
- Visas testējamas funkcionalitātes ir izstrādātas izvirzot par vienu no mērķiem kļūdu un tekstuālo ievadu skaitu samazināšanu.

Atskatoties uz ar lietotājiem veiktajiem novērtēšanas testiem ir zināms, ka korekti definēto vaicājumu skaits ir būtiski palielinājies pēc Add Link un Aggregate Wizard funkcionalitāšu ievešanas, bet pirms jaunāko funkcionalitāšu (piemēram, priekšā teikšanas) izstrādes. Tāpēc būtu nepieciešams novērtēt šo funkcionalitāšu ietekmi uz lietotāju rezultātiem arī tagad, kad ir piedāvāts plašāks metožu klāsts, kas vērsti uz vienāda mērķa sasniegšanas, lai noteikt funkcionalitāšu tālākās attīstības nepieciešamību, un, ja tāda ir nepieciešama, arī virzienu.

Add Outer Query funkcionalitātei nav paredzēti novērtēšanas vaicājumi to lietotāja saskarnes vienkāršības dēļ. Tomēr lai pārliecināties par funkcionalitātes efektivitāti tika veikta darbību analīze vienam no apskatītājiem vaicājumiem (2.14.att.(b)), apskatot arī Aggregate

Wizard funkcionalitātes pievienošanas iespaidu uz rezultātiem. Bez papildus funkcionalitātēm lietotājam bez kopējiem soļiem ir nepieciešami 25 soļi, lai izveidot korektu vaicājumu, bet Add Outer Query pazemināja to skaitu līdz 12. Tik liels ieguvums (52%) ir saistīts ar to, ka pievienota klase nav datu shēmas daļa un tā parametru uzstādīšanai ir nepieciešami vairākas darbības. Pievienojot klāt vēl Aggregate Wizard funkcionalitāti, darbību skaits ir sarūcis līdz 10, un kopējais ieguvums ir 60%.

Veiktā funkcionalitāšu analīze apliecināja, ka visas apskatītas funkcionalitātes spēj samazināt nepieciešamo darbību skaitu, ļaujot lietotājam ātrāk definēt vaicājumu. Tas nozīmē, ka izstrādātas funkcionalitātes sasniedz vienu no izvirzītājiem tās izstrādes mērķiem, nodrošinot lietotāju ar instrumentu sava mērķa sasniegšanai īsākā laikā.

2.5. Tālākie darbi

ViziQuer rīks, atšķirība no daudziem apskatītajiem rīkiem, turpina attīstīties, pēdējo dažu gadu laikā tam ir pievienotas jaunās – ieskaitot arī šī darba ietvaros izstrādātas – funkcionalitātes, kas atrisina iepriekš pamanītas grūtības, kurus izsauca noteiktie vaicājumu tipi. Lietotāju pieredzes novērtējošie testi liecina, ka pievienotas funkcionalitātes sasniedz izvirzīto mērķi. Tomēr ir nepieciešams attīstīt ViziQuer rīka funkcionalitāti, lai arī definējot grūtākus vaicājumus lietotāji pārsvarā spētu izveidot korekto vaicājumu. Lai izprast esošo rezultātu cēloņus ir nepieciešams veikt lietotāju testus, kuru mērķis nav visa rīka testēšanu, bet kas ir vērsti uz noteiktas funkcionalitātes novērtēšanu, kas šī darba turpinājums.

Apskatot citu pētnieku izstrādātus rīkus QueryVOWL rīkam tika pamanīts interesants lietotāju saskarnes risinājums, kad daļa no izsaucamas funkcionalitātēm atrodas uz paša objekta kā aktīvs elements, ļaujot piekļūt biežāk lietotām funkcionalitātēm uzreiz, neizsaucot īsinājumizvēlni vai citus saskarnes elementus. Šis risinājums liekas pievilcīgs, bet dotajā brīdī notiek to novērtēšana no tehniskām ieviešanas iespējām, sarežģītības pakāpes un izmaksām. Iespējams, tik veikta arī rīkjoslas funkcionalitāšu palielināšana, pieliekot jaunas iespējas, kas nav saistītas ar objektu pievienošanu diagrammai, bet ļauj piekļūt dažām biežāk lietotām funkcijām ar vienu klikšķi.

Strādājot pie ViziQuer rīka attīstības par vienu no nākamajiem lieliem virzieniem ir uzskatīta tādas rīka versijas izstrāde, kas spētu labi darboties ar milzīgām datu shēmām kā WikiData. Tas paceļ vairākus jautājumus:

- Ātrdarbība, veidojot sakastus no vairākiem elementiem, kas var būtiski ietekmēt Add Link funkcionalitāti, un vēl vairāk – Connect Classes funkcionalitāti garo ķēžu gadījumos;
- Garo variantu sarakstu attēlošana, ja elementu skaits pārsniedz noteikto skaitli;
- Rezultātu prioritātes noteikšana, jo meklēšana ne vienmēr ļaus izveidot īso sarakstu pietiekami ātri, t.i. ievadot nelielu simbolu skaitu, vai pat īss saraksts datu shēmas izmēra un vārda īpatnības dēļ nebūs iespējams;
- Un citi.

Daļēji šīs problēmas ir adresētas jau esošajā Connect Classes funkcionalitātes realizācijā, piedāvājot lietotājam nerādīt rezultātus, kuros ir iekļautas inversas saites. Tomēr ar to nepietiek, un ir nepieciešams novērtēt citus risinājumus, lai nodrošināt lietotājam iespēju strādāt arī ar šāda izmēra datu shēmām, saglabājot visas esošas ViziQuer rīka iespējas pilnā mērā.

REZULTĀTI

Darba ietvaros veikts pētījums liecina, ka vizuālie vaicājumu rīki paliek pētījumu fokusā, tiek attīstīti vecie un izstrādāti jaunie rīki. Aktīvāka darbība notika 2016.-2018. gados, bet arī pēc tam dažas zinātnieku komandas piedāvā savus risinājumus, kas pēc izstrādātāju domām, spēj atvieglot lietotāju pieredzi, vaicājot pār RDF datiem. Par tādu rīku pieprasījumu liecina vairāku tiešsaistē pieejamajām datu bāzēm pievienoti SPARQL vaicājumvalodu izmantojošie meklēšanas rīki.

ViziQuer vizuālais vaicājumu rīks, atšķirībā no daudziem citiem līdzīgiem rīkiem, par savu mērķi ir uzstādījis nodrošināt lietotājus ar iespēju izmantot SPARQL vaicājumvalodas sarežģītus elementus, kas izvirza lietotājam paaugstinātas prasības tā izglītības līmenim. Tāda rīka izmantošana spēj atrisināt jauno IT nozares vai iepriekš ar citiem vaicājumvalodām strādājošo specialistu specifisko SPARQL vaicājumvalodas konstrukciju nezināšanas gadījumus, ļaujot sasniegt datu iegūšanas mērķi. Atšķirībā no vienkāršākiem, pat triviāliem pielietojumiem, tādiem lietotājiem ir nepieciešama arī bagātīgo vaicājumu definēšanas iespējas.

Izstrādājot "Lietotāja saskarnes risinājumi vizuālā vaicājumu rīkā" darbu tika sasniegti sekojoši rezultāti:

1. Balsoties uz veikta literatūras un pieejamo vizuālu vaicājumu rīku demonstrēšanas lapu izpētes tika noteiktas esošu rīku stiprās un vājās puses, darbības īpatnības un lietotāja saskarnes risinājumi;
2. Novērtēti LU MII izstrādātā ViziQuer vizuālā vaicājumu rīka esošais stāvoklis un saskarnes risinājumi, nosakot esošo funkcionalitāšu uzdevumus, priekšrocības un trūkumus, salīdzinot ar citiem rīkiem un ņemot vērā katra rīka definēto mērķi un potenciālajiem lietotājiem izvirzītas prasības;
3. Novērtēti citos rīkos implementētie lietotāja saskarnes risinājumi un to piemērotība ViziQuer rīkam;
4. Analizēti ViziQuer rīka lietotāju testi, nosakot nepieciešamos risinājumus, kas sekmē ātrāku un precīzāku vaicājumu definēšanu atbilstoši uzdevumam, kā arī nodrošina netiešu piekļuvi datu shēmai;
5. Balstoties uz veiktas analīzes, ir izstrādātas Add Link, Connect Classes un Aggregate Wizard funkcionalitātes līdz līmenim, kas ļauj tās izmantot publiski izplatāmā ViziQuer rīka versijā:

5.1. Add Link un Conncect Classes funkcionalitātēm ir ieviesta meklēšana sarakstā ar iespēju noteikt meklējamas simbolu virknes esamību saites un/vai klases vārdos;

5.2. Add Link funkcionalitātei ir pilnveidota saskarne, ļaujot definēt saites tipu un ļaujot iegūt papildus informāciju par saites tipu, kā arī ļaujot izsaukt Aggregate Wizard vedni;

5.3. Connect Classes funkcionalitātei ir uzlabots vizuālais izskats, paslēpjot funkcionalitātes parametrus, un uzlabota lasāmība, izmantojot krāsas klases vārda izcelšanai;

5.4. Connect Classes funkcionalitātei ir izstrādāts iestatījumu logs, palielinot lietotājam pieejamo iestatījumu klāstu;

5.5. Izstrādāta Aggregate Wizard funkcionalitāte, kas darbojas kā vednis, ļaujot izveidot esamības pārbaudi vai apkopojumu;

5.6. Izstrādātas funkcionalitātes ir pievienotas atbilstošajās vietas, kur tās spēj nodrošināt lietotājam ērtāku rīka izmantošanu;

6. Analizēti ViziQuer rīkam veiktie novērtēšanas testi, kas ir apliecinājuši Add Link un Aggregate Wizard izstrādāto funkcionalitāšu pozitīvo ietekmi uz lietotāju rezultātu korektumu;
7. Sagatavoti lietotāju testi un atbilstoša ViziQuer rīka infrastruktūra, kas ir vērsti uz izstrādāto funkcionalitāšu novērtēšanu ar mērķi noteikt izstrādāto funkcionalitāšu ietekmi uz lietotāju spējas definēt vaicājumus atbilstoši uzdevumam, ņemot vērā ViziQuer rīka pēdējā laikā veiktus uzlabojumus;
8. Veikta sagatavoto novērtēšanas testu analīze un secināti teorētiski iespējami uzlabojumi lietotāju rezultātos no nepieciešamu darbību skatu punkta;
9. Balstoties uz iegūtām zināšanām un rīka izstrādātāju komandas apspriedēm, apskatīti ViziQuer rīka tālākas attīstības iespējamie virzieni;
10. Darba rezultāti ir līdzautoru prezentēti divās konferencēs un ir publicēti atbilstošos konferenču rakstu krājumos [8, 9].

Izstrādājot darbu ir sasniegti visi izvirzītie mērķi un izpildīti definētie uzdevumi.

SECINĀJUMI

Balstoties uz apskatītas literatūras avotiem, izpētītiem vizuālajiem vaicājumu rīkiem un pildot šī darba ietvaros paredzētus uzdevumus, ir izdarīti sekojoši secinājumi:

1. Vaicājuma definēšana kļūst arvien aktuālāks jautājums gan IT nozares speciālistiem, gan lietotājiem bez iepriekšējas atbilstošas izglītības, un to nosaka pieejamās dažāda tipa strukturētas informācija apjoms un daudzveidība;
2. Strukturētiem datiem vaicājumvalodas konstrukcijas spēj nodrošināt informācijas atlasi atbilstoši dažādiem nosacījumiem, tajā skaitā arī komplicētiem, bet tās konstrukcijas nav ne-speciālistiem viegli saprotamas un pareizi pielietojamas;
3. Vizuālo vaicājumu rīku izstrāde ir aktuāla tēma, jo šādi rīki ļauj piekļūt strukturēto datu daļai ātri un ērti, samazinot nepieciešamību izprast tehnisko informāciju (datu shēmas, vaicājumvalodas konstrukcijas), tomēr mūsdienās nav izveidota vienota pieeja vizuālo vaicājumu rīku tipam, bet SPARQL vaicājumvalodai ir plašāk izmantoti uz dabiskās valodas un uz diagrammām balstītie rīki;
4. Uz diagrammām balstītie rīki ļauj vieglāk implementēt SPARQL vaicājumvalodas konstrukcijas, atbalstot arī sarežģītākās, un piekļūt datu shēmas datiem dažādos veidos: tiešā veidā, rādot ontoloģiju, un/vai netiešā veidā, nodrošinot vaicājuma definēšanas posmus ar datu shēmai atbilstošiem variantiem;
5. ViziQuer vizuālais vaicājumu rīks ir izveidots lietotājam ar iepriekšējām zināšanām IT jomā, bet kuram var nebūt labas zināšanas un pieredzes tieši SPARQL vaicājumvalodas izmantošanā, jo tas:
    5.1. Ļauj veikt strukturēt vaicājumus ar projektu un diagrammu līmeņu ieviešanu;
    5.2. Nodrošina lietotājam iespējas definēt bagātīgo vaicājumu, atbalstot plašāku SPARQL vaicājumvalodas konstrukciju atbalstu;
    5.3. Ar saskarnes risinājumiem nodrošina lietotājiem ērtu pieeju dažādām funkcionalitātēm, ļaujot lietotājam izvēlēties sev piemērotu atbilstoši uzdevumam;
6. Šī darba izstrādātas funkcionalitātes ir spējīgas uzlabot lietotāja rezultātus pildot uzdevumus; tas ir pierādīts ar novērtēšanas testiem, to izstrādes kvalitāte ļāva pievienot šī funkcionalitātes ViziQuer rīka publiski izplatāmā versijā;
7. ViziQuer vizuālam vaicājumu rīkam ir uzstādīti ambiciozi mērķi, kuru sasniegšanai ir nepieciešama tālāk rīka attīstība, kas atrisinātu izvirzītos uzdevumus.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. LZA TK ITTEA terminu datu bāze. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: termini.lza.lv
2. An Introduction to Multilingual Web Addresses. [tiešsaiste]. – [atsauce 10.05.2020.] Pieejams: Pieejams: https://www.w3.org/International/articles/idn-and-iri/
3. Microsoft Terminology 2015. Entry from the Microsoft Language Portal. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: termini.lza.lv
4. OWL Web Ontology Language. Reference W3C Recommendation 10 February 2004. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://www.w3.org/TR/owl-ref/
5. LVS ISO 5127:2005 Informācija un dokumentācija. Vārdnīca Informācijas zinātnes termini (bibliotēkas, arhīvi un muzeji). [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: termini.lza.lv
6. Datu pārraides un apstrādes sistēmas. Angļu-krievu-latviešu skaidrojošā vārdnīca — R., SWH, 1995. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: termini.lza.lv
7. Wikidata Query Service. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://query.wikidata.org/
8. Cerans K., Ovcinnikova J., Lace L., Hodakovska J., Romane A., Grasmanis M., Kalnina E., Sprogis A., Sostaks A., "Visual Query Environment over RDF Data", *Proc. the Posters and Demo Track the 15th Int. Conf. on Semantic Systems co-located with 15th Int. Conf. on Semantic Systems (SEMANTiCS 2019),* Alam M. et al. (Eds.). [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: http://ceur-ws.org/Vol-2451/paper-32.pdf
9. Cerans K., Lace L., Romane A., Ovcinnikova J., Kozlovics S., Grasmanis M., Hodakovska J., Sprogis A., Sostaks A., "Visual Queries over Scholarly Data and other Linked Data Endpoints", *ISWC 2019 Satellites. Proc. ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th Int. Semantic Web Conf. (ISWC 2019)*, Suárez-Figueroa et al. (Eds.). [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: http://ceur-ws.org/Vol-2456/paper77.pdf
10. Danko V. "Who owns the information, he owns the world / LatAm Business Outlooks 2016". [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://www.linkedin.com/pulse/who-owns-information-he-world-latam-banking-outlook-2016-vadym-danko
11. Albert Einstein Quotes. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://www.brainyquote.com/quotes/albert_einstein_163057

12. Mitchell Kapor>Quotes>Quotable Quote. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://www.goodreads.com/quotes/1432753-getting-information-off-the-internet-is-like-taking-a-drink
13. How much does it cost to develop search engine in 2019. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://azati.ai/how-much-does-it-cost-to-develop-search-engine/
14. Berners-Lee T., "Linked Data". [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://www.w3.org/DesignIssues/LinkedData.html
15. Wagner A., Ladwig G., Tran T., "Browsing-oriented Semantic Faceted Search", *Database and Expert Systems Applications. DEXA 2011. Lecture Notes in Computer Science,* vol. 6860, Hameurlain A., Liddle S.W., Schewe KD., Zhou X. (eds), Springer, Berlin, Heidelberg, 2011, 303.–319. lpp
16. Heim P., Ertl T., Ziegler J., "Facet graphs: Complex semantic querying made easy", *The Semantic Web: Research and Applications. ESWC 2010. Lecture Notes in Computer Science,* vol. 6088, Aroyo L. et al. (eds), Springer, Berlin, Heidelberg, 2010, 288.–302.lpp
17. Bauleo E., Carnevale S., Catarci T., Kimani S., Leva M., Mecella M., "Design, realization and user evaluation of the SmartVortex Visual Query System for accessing data streams in industrial engineering applications", *J. of Visual Languages and Computing,* vol. 25, Issue 5, 2014, 577.–601. lpp
18. 3 Creating Queries by Example. Oracle® Argus Insight User's Guide. Release 7.0.2. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://docs.oracle.com/cd/E39523_01/doc.702/e38588/qbe.htm
19. Borsje J., Embregts H., "Graphical Query Composition and Natural Language Processing in an RDF Visualization Interface", bachelor thesis, Erasmus School of Economics and Business Economics, Erasmus University Rotterdam, 2006. Pieejams: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.137.5037&rep=rep1&type=pdf
20. Arenas M., Grau B.C., Kharlamov E., Marciuška Š., Zheleznyakov D., "Faceted search over RDF-based knowledge graphs", *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 37–38, 2016, 55.–74. lpp. Pieejams: https://doi.org/10.1016/j.websem.2015.12.002
21. Fathalla S., Lange Ch., Auer S., "A Human-friendly Query Generation Frontend for a Scientific Events Knowledge Graph", 2019. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://doi.org/10.13140/RG.2.2.23596.92802
22. Namaki M.H., Song Q., Wu Y., "NAVIGATE: Explainable Visual Graph Exploration by Examples", *SIGMOD '19: Proc. of the 2019 Int. Conf. on Management of Data*, 2019,

1965.–1968. lpp. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://doi.org/10.1145/3299869.3320245

23. OpenLink iSPARQL, iSPARQL demonstrācijas lapa. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: http://dbpedia.org/isparql/
24. Semantic Crystal. Rīka mājas lapa. Dynamic & Distributed information Systems Group, University of Zurich. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://files.ifi.uzh.ch/ddis/oldweb/ddis/research/talking-to-the-semantic-web/semanticcrystal/index.html
25. Smart P. R., Russell A., Braines D., Kalfoglou Y., Bao J., Shadbolt N. R., "A Visual Approach to Semantic Query Design Using a Web-Based Graphical Query Designer", *Knowledge Engineering and Knowledge Management (EKAW)*, Springer, 2008, 275.–291. lpp.. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://www.researchgate.net/publication/39996370_A_Visual_Approach_to_Semantic_Query_Design_Using_a_Web-Based_Graphical_Query_Designer
26. Hogenboom F., Milea V., Frasincar F., Kaymak U., "RDF-GL: A SPARQL-Based Graphical Query Language for RDF", *Emergent Web Intelligence: Advanced Information Retrieval. Advanced Information and Knowledge Processing*, Chbeir R., Badr Y., Abraham A., Hassanien AE. (eds), Springer, London, 2010, 87.–116. lpp
27. Clemmer A., Davies S., "Smeagol: a "specific-to-general" semantic web query interface paradigm for novices", *Database and Expert Systems Applications (DEXA)*, Springer, 2011, 288.–302. lpp. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: http://stephendavies.org/writings/SmeagolASpecificToGeneral.pdf
28. Bettembourg C., Dameron O., Bretaudeau A., Legeai F., "AskOmics: Integration and Interrogation of Genomic and PostGenomic Regulation Networks". Prezentācija. ProdInra – the INRA (French National Institute for Agricultural Research) open archive. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: http://prodinra.inra.fr/ft?id=2C926ED6-9954-4470-84A7-3BE9C3F6A62A
29. Rebuild of AskOmics. Rīka lapa uz github platformas. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://github.com/askomics/flaskomics
30. SPARQLGraph - Visual Query Builder for Biological RDF Databases. Rīka lapa github platformā [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://github.com/tadKeys/sparqlgraph
31. Sturgeon A., Przybylo A., Milstead K. "Querying Large SysMLModels" Global Product Data Interoperability Summit 2018. Prezentācijas materiāli. [tiešsaiste]. – [atsauce

14.05.2020.] Pieejams: https://gpdisonline.com/wp-content/uploads/2018/09/Boeing-SturgeonPrzybylo-QueryingLargeSysMLModels-MBSE-Open-1.pdf

32. SparqlBlocks. Rīka mājas lapa. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: http://sparqlblocks.org/
33. Using Blockly library for building SPARQL queries with blocks. Rīka lapa uz github platformas. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://github.com/miguel76/SparqlBlocks
34. Visual SPARQL Builder - Model SPARQL-Select-Queries in a browser. Rīka lapa uz github platformas. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://github.com/leipert/vsb
35. Pabón M.C., Collazos C.A., "A Visual Query Language for Data Graphs: an approach from the User-Centered Design", *Interacción '15: Proc. of the XVI Int. Conf. on Human Computer Interaction*, 2015, Article No.: 49. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://doi.org/10.1145/2829875.2829918
36. GraphQL Testing Suite. Rīka lapa uz github platformas. [tiešsaiste]. – [atsauce 14.05.2020.] Pieejams: https://github.com/ManuelDeLeon/GraphtQL
37. Visual Data Web. VOWL: Visual Notation for OWL Ontologies. QueryVOWL mājas lapa. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: http://vowl.visualdataweb.org/queryvowl/
38. Web VOWL. Visualizing ontologies on the Web. Rīka lapa uz github platformas. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://github.com/VisualDataWeb/WebVOWL
39. Optique. Scalable End-user Access to Big Data (Seventh Framework Program (FP7) of the European Commission). Optique projekta mājas lapa. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: http://optique-project.eu/
40. Optique Project. Twitter sociālā tīkla lietotāja konts. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://twitter.com/OptiqueProject
41. OptiqueVQS. Rīka lapa uz github platformas. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://gitlab.com/ernesto.jimenez.ruiz/OptiqueVQS
42. A query interface for the knowledge graph based on a light version of OptiqueVQS. Rīka lapa uz github platformas. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://github.com/TBFY/kg-interface
43. Klungre V.N., Soylu A., Jimenez-Ruiz E., Kharlamov E., Giese M., "Query Extension Suggestions for Visual Query Systems Through Ontology Projection and Indexing", *New*

Gener. Comput., vol. 37, 2019, 361.–392. lpp. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://doi.org/10.1007/s00354-019-00071-1

44. Bartolomeo S. D., Pepe G., Savo D. F., Santarelli V., "Sparqling: Painlessly Drawing SPARQL Queries over Graphol Ontologies", *Int. Workshop on Visualization and Interaction for Ontologies and Linked Data (VOILA'2018)*, Ivanova V. et al. (Eds.), [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: http://ceur-ws.org/Vol-2187/paper6.pdf
45. Visual tool for SPARQL queries on graphol graphs. Rīka lapa uz github platformas. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://github.com/picorana/painless_sparql
46. SPARQL visual query builder and RDF explorer. Rīka lapa uz github platformas. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://github.com/hvarg/RDFExplorer
47. Vargas H., Buil-Aranda C., Hogan A., López C.,"RDF Explorer: A Visual SPARQL Query Builder", *The Semantic Web – ISWC 2019. ISWC 2019. Lecture Notes in Computer Science*, vol 11778, Ghidini C. et al. (eds), Springer, Cham, 2019, 647.-663. lpp. . [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://link.springer.com/chapter/10.1007/978-3-030-30793-6_37
48. Seifer P., Härtel J., Leinberger M., Lämmel R., Staab S., "Empirical study on the usage of graph query languages in open source Java projects", *SLE 2019: Proc. of the 12th ACM SIGPLAN Int. Conf. on Software Language Engineering*, 2019, 152.–166. lpp. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://doi.org/10.1145/3357766.3359541
49. Bhowmick S.S.. "We Don't Need No Education: From Building for Coders to Building for Users", *GRADES-NDA'19: Proc. of the 2nd Joint Int. Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, 2019. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://doi.org/10.1145/3327964.3328491
50. Eiropas Datu Portāls. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://www.europeandataportal.eu/lv
51. Europeana. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://www.europeana.eu/en
52. Personālie datori. Angļu-krievu-latviešu skaidrojošā vārdnīca — R., Dati, 1998. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: termini.lza.lv
53. Saxena S., Dubey S.K., "Impact of Software Design Aspects on Usability", *Int. J. of Computer Applications*, vol. 61, No. 22, 2013, 48.-53. lpp
54. Chen Y.-H., Germain C.A., Rorissa A., "Defining Usability How Library Practice Differs from Published Research", *Portal: libraries and the academy*, vol.11, Issue 2, 2011, 599.-

628. lpp. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: http://scholarsarchive.library.albany.edu/dis_fac_scholar/4

55. Kaufmann E., Bernstein A., "How Useful Are Natural Language Interfaces to the Semantic Web for Casual End-Users?", *ISWC/ASWC – 2007 – LNCS 4825*, Aberer K. et al. (Eds.), 281.–294. lpp
56. Elbedweihy K., Wrigley S.N., Ciravegna F., "Evaluating Semantic Search Query Approaches with Expert and Casual Users", *ISWC 2012 – LNCS 7650*, Part II, Cudrer-Mauroux P. et al. (Eds.), Springer-Verlag Berlin Heidelberg, 2012, 274.–286. lpp
57. VSB Prototyp. Visual Query Builder rīka lapa. Rīka lapa uz github platformas. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: http://leipert.github.io/vsb/dbpedia/
58. Hodakovska J., "Lietotāja servisi vizuālā vaicājumu rīkā", bakalaura darbs, Datorikas fakultāte, Latvijas Universitāte, Rīga, 2018
59. QueryVOWL beta 0.1.1. QueryVOWL rīka lapa. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: http://vowl.visualdataweb.org/queryvowl/queryvowl.html#
60. Soylu A., Giese M., Jimenez-ruiz E., Vega-Gorgojo G., Horrocks I., "Experiencing OptiqueVQS: A Multi-paradigm and Ontology-based Visual Query System for End Users", *Universal Access in the Information Society*, vol. 15, Issue 1, 2015, 129.–152. lpp.
61. Sparqling rīka lapa. Rīka lapa uz github platformas. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://picorana.github.io/sparqling/
62. RDF Explorer. Rīka tiešsaistes versija. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://www.rdfexplorer.org/
63. Wylot M., Hauswirth M., Cudré-Mauroux P., Sakr Sh., "RDF Data Storage and Query Processing Schemes: A Survey", *ACM Comput. Surv*. 51, 4, , 2018, Article 84. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://doi.org/10.1145/3177850
64. Europeana pro. Europeana SPARQL API. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://pro.europeana.eu/page/sparql#examples
65. Catarci T., Santucci G., "5. Diagrammatic Languages: A Experiment", V*isual Database Systems* 3, S. Spaccapietra et al. (eds.) Springer Science+Business Media Dordrecht, 1995
66. Vega-Gorgojo G., Slaughter L., Giese M., Heggestøyl S., Soylu A., Waaler A., "Visual query interfaces for semantic datasets: An evaluation study", *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 39, 2016, 81.–96.lpp. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: http://dx.doi.org/10.1016/j.websem.2016.01.002
67. Bhowmick S. S., Choi B., Li C., "Graph querying meets HCI: State of the art and future directions", *ACM Int. Conference on Management of Data*, ACM, 2017, 1731.–1736. lpp.

68. Haag F., Lohmann S., Siek S., Ertl T., “QueryVOWL: A Visual Query Notation for Linked Data”. *Extended Semantic Web Conference (ESWC)*, Springer, 2015, 387.–402. lpp.
69. Soylu A., Kharlamov E., Zheleznyakov D., Jime´nez-Ruiz E., Giese M., Skjæveland M. G., Hovland D., Schlatte R., Brandt S., Lie H., Horrocks I., “OptiqueVQS: A visual query system over ontologies for industry”. *Semantic Web*, vol. 9, Issue 5, 2018, 627.–660. lpp,
70. Cerans K., Sostaks A., Bojars U., Ovcinnikova J., Lace L., Grasmanis M., Romane A., Sprogis A., and Barzdins J., “ViziQuer: A Web-Based Tool for Visual Diagrammatic Queries Over RDF Data”, *European Semantic Web Conference (ESWC)*, 2018, 158.–163. lpp
71. Barzdins G., Rikacovs S., Zviedris M., “Graphical Query Language as SPARQL Frontend”, *Local Proc. of 13th East-European Conference*, 2009, 93.-107. lpp
72. Barzdins G., Liepins E., Veilande M., Zviedris M., “Ontology Enabled Graphical Database Query Tool for End-Users”, *Selected papers from DBIS'2008,* F*rontiers in Artificial Intelligence and Applicatons series*, Haav H.-M. (Eds.), IOS Press, 2009
73. Zviedris M., “Dati kā ontoloģija – glabāšana, vaicāšana, vizualizācija”, promocijas darbs, Datorikas fakultāte, Latvijas Universitāte, 2014
74. Hodakovska J., “ViziQuer rīka realizācija tīmekļa vidē”, kvalifikācijas darbs, Datorikas fakultāte, Latvijas Universitāte, 2016
75. Čerāns K., Bārzdiņš J., Šostaks A., Ovčiņņikova J., Lāce L., Grasmanis M., Sproģis A., “Extended UML Class Diagram Constructs for Visual SPARQL Queries in ViziQuer/web”, *VOILA 2017. Visualization and Interaction for Ontologies and Linked Data. Proc. of the Third Int. Workshop on Visualization and Interaction for Ontologies and Linked Data co-located with the 16th Int. Semantic Web Conference (ISWC 2017)*. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: http://ceur-ws.org/Vol-1947/ paper08.pdf
76. ViziQuer/web Tool for RDF Data Analysis Queries. ViziQuer rīka mājas lapa. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: http://viziquer.lumii.lv/
77. ViziQuer. Structured Semantic Data Search Tool. Galdvirsmas versijas mājas lapa. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: http://viziquer.lumii.lv/local/index.html
78. Structured Semantic Data Search Tool. Rīka lapa uz github platformas. [tiešsaiste] – [atsauce 14.05.2020.] Pieejams: https://github.com/LUMII-Syslab/viziquer/
79. Hodakovska J., “Lietotāja saskarnes risinājumi vizuālā vaicājumu rīkā: esošās situācijas analīze”, kursa darbs, Datorikas fakultāte, Latvijas Universitāte, 2020

80. Čerāns K., “ViziQuer/web notācija, rīks un metodoloģija”, VPP NexIT 2014-2017 projekta «Ontoloģiju, semantiskā tīmekļa un drošības tehnoloģijas» rezultātu prezentācija. 13.04.2018.
81. Cerans K., Sostaks A., Bojars U., Ovcinnikova J., Lace L., Grasmanis M., Romane A., Sprogis A., Barzdins J., “ViziQuer: a Visual Notation for RDF Data Analysis Queries”, *Metadata and Semantic Research. MTSR 2018. Communications in Computer and Information Science*, vol 846, Garoufallou E., Sartori F., Siatri R., Zervas M. (eds), Springer, Cham, 2019

Maģistra darbs: **Lietotāja saskarnes risinājumi vizuālā vaicājumu rīkā**

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

Autors: _____/ Jūlija Hodakovska/
(Autora paraksts)

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: _____/ Kārlis Čerāns /___
(Vadītāja paraksts)

Darbs iesniegts maģistrantūras sekretariātā __18.05.2020._____.
(Iesniegšanas datums)

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

Studiju metodiķe: _____.
(Metodiķes paraksts)

Recenzents: _____ profesors, Dr.filoz. Jurģis Šķilters_____
(Akad. amats, zin. grāds, vārds, uzvārds)

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____
(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____
(Sekretāra paraksts)