LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

VIZUĀLO VAICĀJUMU ĢENERĒŠANA NO TEKSTUĀLĀS FORMAS

MAĢISTRA DARBS

Autors: **Inga Pīre**
Stud. apl. Nr. ip11201
Darba vadītājs: profesors Dr. dat. Kārlis Čerāns

RĪGA 2020

ANOTĀCIJA

Pieaugot datu apjomam tīmeklī, ir nepieciešami piemēroti rīki informācijas atrašanai. Lai izpildītu vaicājumus pār RDF grafiem, tiem ir jāatbilst SPARQL sintaksei. Pastāv vairāki rīki, kas ļauj būvēt vaicājumus no grafiskiem elementiem un automātiski pārveidot tos par tekstuāliem SPARQL vaicājumiem, viens no šādiem rīkiem ir *ViziQuer*. Taču sarežģītāks uzdevums ir automātiski vizualizēt tekstuālus vaicājumus.

Darba mērķis ir veikt sistemātisku analīzi par SPARQL vaicājumu vizualizācijas iespējām *ViziQuer* notācijā un novērtēt, kurās situācijās izstrādes stadijā esošais vizualizācijas algoritms spēj vaicājumus vizualizēt veiksmīgi un kurās neveiksmīgi, kā arī iezīmēt iespējamos risinājumus neveiksmīgas vizualizācijas gadījumā.

Rezultātā ir izveidota vaicājumu kopa, kura ir praktiski izmantojama tālākā *ViziQuer* rīka pilnveidē. Atklātajām nepilnībām ir doti ieteikumi to novēršanai un sniegti ierosinājumi vizuālās notācijas papildināšanai.

Atslēgvārdi: *ViziQuer*, SPARQL, RDF, vaicājumu vizualizēšana

ABSTRACT

GENERATING VISUAL QUERIES FROM TEXTUAL FORM

As the amount of data on the web increases, appropriate tools for finding information are needed. To execute queries over RDF graphs, they must conform to the SPARQL syntax. Several tools exist that allow to build queries from graphical elements and automatically convert them to textual SPARQL queries, one of them is *ViziQuer*. However, more complex task is to automatically visualize textual queries.

The purpose of thesis is to perform a systematic analysis of SPARQL query visualization possibilities in *ViziQuer* notation and to assess in which cases the visualization algorithm is able to visualize queries successfully and in which cases unsuccessfully, as well as to mark possible solutions in case of unsuccessful visualization.

In result a set of queries have been created that can be used in further development of *ViziQuer*. For discovered shortcomings recommendations are given for their elimination and suggestions for supplementing the visual notation are also provided.

Keywords: *ViziQuer*, SPARQL, RDF, query visualization

AUTOREFERĀTS

Darba ietvaros tika veikta sistemātiska literatūras apskate, aplūkojot 10 dažādus vaicājumu vizualizācijas rīkus. Rezultātā tika noskaidrots, ka tikai viens no 10 aplūkotajiem rīkiem spēj ģenerēt vizuālus vaicājumus no tekstuāliem SPARQL vaicājumiem. Praktiskajā daļā veicot sistemātisku analīzi par SPARQL vaicājumu vizualizācijas iespējām *ViziQuer* notācijā, autore izveidoja vaicājumu kopu vizualizācijas spēju novērtēšanai. Izveidotā vaicājumu kopa ir sadalīta 6 apakškopās jeb vaicājumu klasēs. Vaicājumu kopa ir praktiski izmantojama tālākā *ViziQuer* rīka pilnveidē. Darba izstrādes laikā atklātajām vizualizācijas moduļa nepilnībām ir doti priekšlikumi to novēršanai. Praktiskā daļa veido aptuveni pusi no šī darba.

Darbā ir aplūkoti 37 dažādi literatūras avoti, no kuriem 9 ir Latvijas Universitātes Datorikas fakultātē izstrādāti noslēguma darbi saistībā ar rīka *ViziQuer* izstrādi un pilnveidošanu, 8 ir interneta avoti, 19 ir žurnālu publikācijas, konferences darbi un atskaites, 1 ir standarts, kas izmantots SPARQL vaicājumu kopas izveidošanā.

Lai lasītājs varētu veiksmīgi lasīt un saprast darbu, ir aprakstītas darbā izmantotās semantiskās tehnoloģijas. Literatūras apskate par dažādiem vaicājumu vizualizēšanas rīkiem un vaicājumu piemēru kopas veidošana ir veikta un darbā aprakstīta sistemātiski, lai izvairītos no elementu atlases neobjektivitātes. Izveidotajiem tekstuālajiem un vizuālajiem vaicājumiem ir doti attēli un detalizēti apraksti.

Darbs tika noformatēts balstoties uz formālajām un neformālajām prasībām, kuras dotas Latvijas Universitātes Datorikas fakultātē izstrādātajā dokumentā "Maģistra darba izstrādes un aizstāvēšanas metodiskie norādījumi". Darba noformējums atbilst metodisko norādījumu dokumentā 6. pielikumā dotajam darba noformējuma kontrolsarakstam. Nozares terminu tulkošanai no angļu valodas uz latviešu valodu tika izmantots LZA terminoloģijas portāls[1]. Pareizrakstības un gramatikas kļūdu pārbaudei tika izmantots programmā *Microsoft Word* piedāvātais pareizrakstības un gramatikas pārbaudes rīks.

Visi izmantotie literatūras avoti, no kuriem ir aizgūtas idejas, formulējumi un attēli, darbā ir atzīmēti ar literatūras atsaucēm.

[1] http://termini.lza.lv/term.php?

SATURA RĀDĪTĀJS

APZĪMĒJUMU SARAKSTS

Ajoo - platforma (ietvars) tīmekļa grafisko rīku veidošanai.

HTML (*Hypertext Markup Language*) - hiperteksta iezīmēšanas valoda tīmekļa lapu veidošanai.

JSON (*JavaScript Object Notation*) - datu apmaiņas formāts, kurš ir saprotams un apstrādājams gan cilvēkiem, gan mašīnām.

Lua - vairāku paradigmu programmēšanas valoda, kas veidota kā skriptu valoda ar paplašinātu semantiku.

LUMII - Latvijas Universitātes Matemātikas un Informātikas institūts**.**

Meteor - platforma tīmekļa un mobilo lietotņu izveidošanai izmantojot *JavaScript*.

OWL (*Web Ontology Language*) - semantiskā tīmekļa valoda, kas paredzēta bagātīgu un sarežģītu zināšanu atspoguļošanai par lietām, lietu grupām un attiecībām starp lietām.

RDF (*Resource Description Framework*) - resursu aprakstīšanas ietvars.

SKOS (*Simple Knowledge Organization System*) - vienkārša zināšanu organizācijas sistēma.

SPARQL (*SPARQL Protocol and RDF Query Language*) – vaicājumu valoda vaicājumu veidošanai pār RDF grafiem.

URI (*Uniform Resource Identifier*) - vienotais resursu identifikators, kuru veido īsa, noteiktai sintaksei atbilstoša simbolu virkne.

W3C (*World Wide Web Consortium*) - galvenā starptautiskā organizācija, kas veido standartus vispasaules tīmeklim.

XML (*Extensible Markup Language*) - paplašināmā iezīmēšanas valoda.

IEVADS

Datu apjoms tīmeklī pieaug un ir nepieciešamas atbilstošas tehnoloģijas un rīki pieejamās informācijas atrašanai un atbilžu rašanai uz specifiskiem jautājumiem. RDF nodrošina pamatu datu publicēšanai un sasaistei [1]. Lai vaicātu šos datus, tiek izmantota RDF specifiska vaicājuma valoda – SPARQL [2]. Taču, lai izpildītu vaicājumus pār RDF grafiem, tiem ir jāatbilst SPARQL sintaksei, kas var būt šķērslis informācijas iegūšanai cilvēkiem bez tehniskām zināšanām. Lai lietotāji varētu izpildīt SPARQL vaicājumus bez zināšanām par SPARQL sintaksi, ir izstrādāti vairāki rīki, kas ļauj būvēt vaicājumus no grafiskiem elementiem un pārveido tos par tekstuāliem vaicājumiem pirms to izpildes.

Viens no šādiem rīkiem ir *ViziQuer*, taču tāpat kā lielākā daļa konkurējošo rīku, tas nepiedāvā veikt inverso pārveidošanu – no tekstuālas SPARQL sintakses ģenerēt atbilstošus vizuālos vaicājumus. Rīks *ViziQuer* vēl ir izstrādes stadijā, un tas tiek nepārtraukti uzlabots. Arī vizualizācijas modulis, kurš nodrošina vizuālo vaicājumu ģenerēšanu no tekstuāliem SPARQL vaicājumiem, ir izstrādes stadijā.

Darba mērķis ir apskatīt dažādu rīku spēju automātiski vizualizēt tekstuālus vaicājumus un veikt sistemātisku analīzi par SPARQL vaicājumu vizualizācijas iespējām *ViziQuer* notācijā, tajā skaitā novērtēt izstrādes stadijā esošā vizualizācijas moduļa spējas, kurās situācijās tas vaicājumus spēj vizualizēt veiksmīgi un kurās situācijās vizualizācija ir neveiksmīga.

Lai sasniegtu izvirzīto mērķi, tika noteikti šādi uzdevumi:

- veikt literatūras un standartu apskati par tādām semantiskajām tehnoloģijām kā SPARQL, RDF un OWL;
- aplūkot Latvijas Universitātē izstrādātos noslēguma darbus saistībā ar rīku *ViziQuer*, lai noskaidrotu, kas ir paveikts un izpētīts pirms autores darba uzsākšanas;
- aplūkot rīkam *ViziQuer* konkurējošus rīkus, kas ļauj veidot grafiskus SPARQL vaicājumus un izpētīt, vai kāds no šiem rīkiem spēj automātiski veidot vizuālus vaicājumus no tekstuāliem vaicājumiem un kā šāda funkcionalitāte ir izstrādāta konkrētajā rīkā;
- iepazīties ar SPARQL vaicājumvalodas specifikāciju un definēt vaicājumu klases;
- balstoties uz definētajām vaicājumu klasēm, veikt vaicājumu izveidi un novērtēt vizualizācijas iespējas rīkā *ViziQuer*.

Darbs ir sadalīts četrās nodaļās. Pirmajā nodaļā ir aplūkota informācija par semantisko tīmekli un tā atbalstošajām tehnoloģijām kā RDF, SPARQL un OWL. Otrajā nodaļā tiek aplūkoti pētījumi, kas veikti pirms autores darba uzsākšanas pie rīka *ViziQuer*, lai noskaidrotu, kas šajā jomā jau ir sasniegts un izpētīts. Kā arī tiek apskatīta vaicājumu veidošana izmantojot rīku *ViziQuer* un ir dots vispārīgs apraksts par esošo risinājumu tekstuālu vaicājumu ģenerēšanai no vizuāliem vaicājumiem un arī pretēji - vizuālu vaicājumu ģenerēšanai no tekstuāliem vaicājumiem. Trešajā nodaļā ir veikta sistemātiska literatūras apskate, lai noteiktu kādi vizuālu SPARQL vaicājumu veidošanas rīki pastāv un izpētītu, vai kāds no šiem rīkiem spēj automātiski vizualizēt tekstuālus vaicājumus. Darba ceturtajā nodaļā ir izdalītas sešas vaicājumu klases, uz kurām balstoties tālāk šajā nodaļā ir analizēta *ViziQuer* spēja vizualizēt vaicājumus. Darbu noslēdz rezultāti un secinājumi.

1. SEMANTISKĀS TEHNOLOĢIJAS

Cilvēkiem ir viegli saprast ikdienas valodu un vārdu nozīmi, taču datoriem šādas spējas nepiemīt. Sākotnēji tīmeklis tika veidots galvenokārt cilvēkiem, tā mērķis bija nodrošināt pieejamību dokumentiem cilvēkiem lasāmā formātā [3]. Piemēram, ja ir nepieciešams atrast veikalu, kurā kāda prece tiek pārdota par visizdevīgāko cenu, var izmantot kādu no tīmekļa vietnēm, kurā ir iespējams salīdzināt preces cenu dažādos veikalos. Bez datiem mašīnlasāmā formā preču cenu salīdzināšanas vietnēm ir jāizmanto programmas, kas iegūst nepieciešamo informāciju no vairākām veikalu vietnēm, izmantojot ekrāna satura nolasīšanu. Pirms šādas programmas izveidošanas izstrādātājam ir jāveic HTML struktūras analīze katrai veikala vietnei, no kuras nepieciešams iegūt datus. Taču šāds risinājums nav ilgtspējīgs, ja kādai veikala tīmekļa vietnei tiks veiktas izmaiņas HTML dokumentā, būs jāveic izmaiņas arī ekrāna satura nolasīšanas programmā.

Mūsdienās arvien vairāk tīmeklī tiek izplatīti dati arī mašīnlasāmā formātā. Bieži vien tie tiek izplatīti atsevišķos failos un atšķirīgos formātos, turklāt dati cilvēkiem lasāmā formātā var atšķirties no datiem mašīnlasāmā formātā [3]. Rezultātā programmas redz tikai informāciju par datu prezentāciju, bet ne par datu nozīmi. W3C strādā pie tehnoloģiju kopas veidošanas [4], kas ļautu veidot tādu datu tīmekli, kura saturs būtu mašīnlasāms un dotu iespēju datoriem veikt vairāk noderīgu darbu, izmantojot pieejamos datus. Termins "semantiskais tīmeklis" attiecas uz W3C redzējumu par saistīto datu tīmekli, kas ir mašīnlasāms [4]. Semantiskā tīmekļa galvenie mērķi ir uzlabot zināšanu reprezentāciju un informācijas izguvi no tīmekļa. Semantisko tīmekli atbalsta tādas tehnoloģijas kā RDF, SPARQL, OWL un SKOS [4]. Šajā darbā tiks vairāk apskatīts par trīs no šīm tehnoloģijām – RDF, SPARQL un OWL.

Mūsdienās datu apjoms tīmeklī pieaug. Cilvēki, uzņēmumi un ierīces ir kļuvuši par datu "ražotājiem", kas katru dienu izveido lielu datu apjomu tīmeklī. Pēc *Internet World Stats*[2] statistikas kopš 2019. gada 30. jūnija pasaulē ir 4,5 miljardi interneta lietotāju, kas ir vairāk nekā puse pasaules populācijas. Pieaugot interneta lietotāju skaitam, pieaug arī saražotais datu apjoms tīmeklī. Ierīces, ko izmanto ikdienā, ir milzīgs datu avots. Šīs ierīces ir ne tikai mobilās ierīces, bet arī viedtelevizors, automašīnas, lidmašīnas un daudzas citas. Lietu internets rada arvien lielāku datu daudzumu, pēc *Cisco* prognozes līdz 2023. gadam IP tīkliem pieslēgto ierīču skaits būs vairāk nekā trīs reizes lielāks salīdzinājumā ar pasaules iedzīvotāju skaitu [5]. Ir būtiski, ka pastāv tehnoloģijas, kas ļauj saprast un iegūt zināšanas no lielā datu apjoma tīmeklī.

[2] https://www.internetworldstats.com/stats.htm

Domēna modeļiem ir svarīga loma visā programmatūras izstrādes dzīvesciklā, sākot ar prasību analīzi un projektēšanu līdz ieviešanai un arī pēc tās. Mūsdienās pastāv vairāki programmatūras izstrādes rīki, kas atbalsta koda ģenerēšanu no UML diagrammām, turklāt uz modeļa virzīta arhitektūra ļauj izstrādātājiem sinhronizēt un verificēt tehnisko realizāciju pret lietotāju prasībām [6]. Tomēr domēna modeļu atkārtota lietojamība bieži ir ierobežota, jo tie pēc definīcijas ir specifiski kādam domēnam un ņem vērā tikai abstrakcijas, kas vajadzīgas, lai rastu risinājumu individuālai problēmai, taču tīmeklis ir daudz plašāks par to [6]. Kaut arī liela daļa programmatūras arvien vairāk tiek iegulta tīmeklī, izstrādes procesos vēl pilnībā neizmanto potenciālu, ko var sniegt modeļu atkārtota izmantošana no tīmekļa, līdz ar to semantiskais tīmeklis var kalpot kā platforma, kurā var izveidot, koplietot un atkārtoti izmantot domēna modeļus [6].

1.1 Ontoloģijas un OWL

Ontoloģija ir terminu kopums, ko izmanto, lai aprakstītu kādu domēnu [7]. Ontoloģija ir specifiska konkrētam domēnam un raksturo kādu zināšanu jomu, tas ir formāls veids, kā definēt struktūru zināšanām kādā jomā. Domēns apzīmē kādu konkrētu priekšmetu reālā pasaulē, piemēram, izglītību, sportu un medicīnu. Ontoloģija satur jēdzienus un attiecības starp šiem jēdzieniem [7]. Zināšanas, kas ir attēlotas ontoloģijā, ir viegli saprotamas un apstrādājamas datoriem.

Pastāv vairākas valodas, kuras paredzētas ontoloģiju veidošanai, taču visbiežāk izmantotā ir OWL [7]. OWL ir tīmekļa ontoloģiju valoda, kas paredzēta bagātīgu un sarežģītu zināšanu atspoguļošanai par lietām, lietu grupām un attiecībām starp lietām [7]. Zināšanas, kas ir izteiktas izmantojot OWL, var izmantot datorprogrammās [7]. OWL dokumentus, kas pazīstami kā ontoloģijas, var publicēt tīmeklī, un tie var atsaukties uz citām OWL ontoloģijām. OWL ir daļa no W3C semantiskā tīmekļa tehnoloģiju kopas [4].

Pašreizējo OWL versiju [7] jeb OWL 2 ir izstrādājusi W3C OWL darba grupa, šī versija tika publicēta 2009. gadā, un otrais izdevums tika publicēts 2012. gadā. OWL 2 ir paplašinājums un pārskatījums 2004. gada OWL versijai.

Uz ontoloģiju bāzēta datu piekļuve (OBDA - *Ontology Based Data Access*) attiecas uz virkni paņēmienu, algoritmu un sistēmu, ko var izmantot, lai risinātu problēmas saistībā ar datu neviendabīgumu, kas ir izplatīta daudzās organizācijās un arī tīmeklī [8]. OBDA paradigmā ontoloģija definē augsta līmeņa shēmu datu avotam un nodrošina vārdnīcu lietotāju vaicājumiem.

Ontoloģijas tiek izmantotas, lai nodrošinātu globālu skatu pār vairākām lokālām datu kopām, lai aprakstītu attiecības starp šādām globālām un lokālām shēmām, parasti izmanto kartēšanu [8]. Kopš pirmsākumiem šī joma ir attīstījusies vairākos virzienos – viens no tiem ir uz OBDA balstītas vaicājumu tulkošanas metodes, kas balstās uz kartēšanu un ir īpaši noderīgas dinamiskiem datu avotiem [8].

1.2 RDF

RDF (resursu aprakstīšanas ietvars) nodrošina pamatu datu publicēšanai un sasaistei, tas ir standarta modelis datu apmaiņai tīmeklī [1]. Šajā datu modelī fakti tiek izteikti kā trīsdaļīgi apgalvojumi, kuri tiek saukti par "trijniekiem" [1]. Katrs trijnieks sastāv no subjekta, predikāta un objekta, vienkāršāk šīs daļas var nosaukt kā aprakstāmā resursa identifikators, īpašības nosaukums un īpašības vērtība.

1.1. tabula

RDF trijnieku piemēri

Subjekts	**Predikāts**	**Objekts**
Jānis	Dzimšanas vieta	Rīga
Līga	Dzimšanas gads	1990
Ieva	E-pasts	ieva@gmail.com

Piemēri RDF trijniekiem ir redzami 1.1. tabula. Tabulā redzamie trijnieki veido tādus apgalvojumus kā "Jāņa dzimšanas vieta ir Rīga", "Līgas dzimšanas gads ir 1990" un "Ievas e-pasts ir ieva@gmail.com". Vienai īpašībai var būt vairākas vērtības, piemēram, Ievai var būt divas e-pasta adreses.

RDF piedāvā funkcijas, kas atvieglo datu sapludināšanu, pat ja pamatā esošās shēmas atšķiras, kā arī RDF modelis ļauj kombinēt strukturētus un daļēji strukturētus datus [1]. Apvienojot dažādus datu avotus var rasties neskaidrības un apjukums starp līdzīgiem nosaukumiem. Lai no tā izvairītos, RDF trijnieka subjektam un predikātam ir jābūt piederīgiem kādai noteiktai nosaukumu telpai (*namespace*). Nosaukumu telpas reprezentēšanai tiek izmantots URI jeb vienotais resursu identifikators [1], kas viennozīmīgi identificē kādu resursu. Piemēram, URI Ievai no tabulā dotās datu kopas varētu būt šāds: "*http://dbpedia.org/resource/Ieva*".

Dažādas tehnoloģijas ļauj iegult datus dokumentos, piemēram, viena no šādām tehnoloģijām ir RDFa [3], kas nodrošina iespēju glabāt RDF trijniekus HTML dokumentos. Lai marķētu mašīnlasāmus datus HTML dokumentos, W3C ir izstrādājis rekomendāciju RDFa izmantošanai HTML5 dokumentos [3]. RDFa piedāvā HTML atribūtu kopumu, kas ļauj dokumentā attēlotos datus atzīmēt ar mašīnlasāmiem atribūtiem. Izmantojot RDFa, datus ir iespējams padarīt lasāmus gan cilvēkiem, gan mašīnām, nedublējot saturu.

1.3 SPARQL

Pēc iepriekšējās darba apakšnodaļas par RDF, ir skaidrs, ka pastāv datu modeļi un formāti, kuri ir mašīnlasāmi. Lai varētu meklēt un analizēt datus, kas ir aprakstīti izmantojot RDF, ir nepieciešama vaicājumu valoda, kurā var izveidot vaicājumu un iegūt vēlamos rezultātus. "Vaicājums" semantiskā tīmekļa kontekstā nozīmē tehnoloģijas un protokolus, kurus izmanto, lai iegūtu informāciju no datu tīkla [2]. Tāpat kā relāciju datu bāzēm ir vajadzīga īpaša vaicājumu valoda SQL, arī datu tīmeklim, ko parasti attēlo izmantojot RDF grafu, ir nepieciešama RDF specifiska vaicājuma valoda. Šī valoda ir SPARQL vaicājumu valoda un ar to saistītie protokoli [2].

Pieaugot datu apjomam tīmeklī, 2008. gada sākumā W3C izdeva trīs jaunas specifikācijas priekš SPARQL, kas atvieglotu datu kopīgošanu un atkārtotu izmantošanu starp dažādām lietotnēm, uzņēmumiem un kopienām, šīs specifikācijas ir: SPARQL vaicājumvaloda priekš RDF, SPARQL protokols priekš RDF un SPARQL vaicājumu rezultāti XML formātā [9]. 2013. gadā ir publicēts SPARQL 1.1 standarts [10], kas sastāv no 11 specifikācijām.

SPARQL vaicājumu valoda nodrošina dažādas iespējas vaicājumu veidošanai, kuras detalizēti ir aprakstītas SPARQL vaicājumu valodas specifikācijā [11]. SPARQL var pielīdzināt relāciju datu bāžu vaicājumvalodai SQL - abas šīs valodas dod lietotājam iespēju veidot, apvienot un iegūt strukturētus datus. SQL vaicājumi tiek izpildīti relāciju datu bāzēs, taču SPARQL vaicājumi tiek izpildīti saistīto datu tīklā. Pastāv četras SPARQL vaicājumu formas [11]:

- *SELECT* - tiek atgriezti visi vaicājuma šablona mainīgie vai to apakškopa;
- *CONSTRUCT* – tiek atgriezts RDF grafs, kas izveidots aizstājot mainīgos trijnieku šablonos;
- *ASK* - tiek atgriezta Būla vērtība, norādot, vai vaicājuma šablons atbilst vai neatbilst;
- *DESCRIBE* – tiek atgriezts RDF grafs, kurā aprakstīti visi atrastie resursi.

Tipiski SPARQL vaicājumi atbilst šādai struktūrai, kurā dažas no daļām ir neobligātas [11]:

- prefiksu saīšņu deklarēšana (neobligāta daļa): "*PREFIX :* <...>";
- vaicājuma rezultāta klauzula: "*SELECT* ...";
- datu kopas definēšana (neobligāta daļa): "*FROM* <...>";
- vaicājuma šablons: "*WHERE {...}*";
- vaicājuma modificētāji (neobligāta daļa): "*GROUP BY, HAVING, ORDER BY, LIMIT, OFFSET*".

Iepriekšējā sadaļā tika aprakstīts, ka RDF trijnieki sastāv no subjekta, predikāta un objekta. Līdzīgi arī SPARQL vaicājumu valodā nosacījumi tiek aprakstīti izmantojot trijniekus, kuros atšķirībā no RDF trijnieka, visi vai viens no subjekta, predikāta un objekta var būt mainīgie.

```
1  PREFIX : <http://lumii.lv/ontologies/UnivExample.owl#>
2  SELECT ?studentName WHERE
3  {
4    ?Student a :Student.
5    ?Student :studentName ?studentName.
6  }
```

1.1. att. **SPARQL vaicājuma piemērs**

Piemērs SPARQL vaicājumam ir dots 1.1. att., vaicājumā tiek atlasīti visi studentu vārdi. Vaicājuma 4. un 5. rindiņā ir redzami piemēri trijnieku šabloniem, kuri satur mainīgos. SPARQL vaicājumu valodā mainīgos var norādīt izmantojot gan jautājuma zīmes simbolu ("?"), gan dolāra zīmes simbolu ("$") [11], rīks *ViziQuer* atbalsta abus prefiksus, taču, ģenerējot tekstuālu SPARQL vaicājumu no vizuāla vaicājuma, mainīgie tiek norādīti izmantojot jautājuma zīmi.

2. RĪKS VIZIQUER

Lai izpildītu vaicājumus pār RDF grafiem, ir izstrādāti vairāki rīki, kas ļauj būvēt vaicājumus no vizuāliem elementiem un pārveido tos par tekstuāliem SPARQL vaicājumiem pirms to izpildes. Viens no šādiem rīkiem ir rīks *ViziQuer*, kurš ir izstrādāts LUMII un tiek attīstīts tālāk.

Šajā nodaļā tiek aplūkoti pētījumi, kas veikti pirms autores darba uzsākšanas pie rīka *ViziQuer*, lai noskaidrotu, kas šajā jomā jau ir sasniegts un izpētīts. Tiek apskatīta arī vaicājumu veidošana, izmantojot rīku *ViziQuer*.

2.1 Iepriekš veikti pētījumi

Rīka *ViziQuer* izstrāde un uzlabošana notiek vairākus gadus, kuru laikā ir tapuši vairāki pētījumi. Lai atrastu pirms darba uzsākšanas veiktos pētījumus, tika izmantota akadēmiskā meklētājprogramma *Google Scholar*[3]. Izmantojot meklēšanas terminu "*ViziQuer*", kopā tika atrasti 115 meklēšanas rezultāti. Viens no pirmajiem pētījumiem, kurš ir saistīts ar rīku *ViziQuer*, ir veikts 2009. gadā [12].

Latvijas Universitātes Datorikas fakultātē ir izstrādāti arī vairāki noslēguma darbi saistībā ar rīku *ViziQuer*. Pirmie darbi ir izstrādāti jau 2010. gadā. Viens no tiem ir kvalifikācijas darbs "*ViziQuer* grafiku kompilēšana par SPARQL" [13]. Šajā darbā tika izstrādāta programma, kas vizuālos SPARQL vaicājumus pārveido par tekstuāliem SPARQL vaicājumiem, savukārt tekstuālie vaicājumi tālāk tiek izpildīti uz servera *Virtuoso Universal Server* un pēc vaicājuma izpildes tiek izvadīts rezultāts tekstuālā formā [13]. Programma tika veidota izmantojot programmēšanas valodu *Lua* un tās bibliotēku *IQuery*. Šajā darbā tika izstrādāta iespēja veidot vizuālus SPARQL vaicājumus izmantojot tikai nelielu daļu no mūsdienās pieejamām SPARQL iespējām.

2010. gadā ir izstrādāts vēl viens kvalifikācijas darbs saistībā ar rīku *ViziQuer* - "*ViziQuer* pieprasījumu grafiskās ievadformas realizācija", līdzīgi kā iepriekš minētajā darbā, programmas daļa ir veidota izmantojot programmēšanas valodu *Lua* un tās bibliotēku *IQuery* [14].

Abi 2010. gadā izstrādātie kvalifikācijas darbi attiecas uz rīka darbvirsmas versiju. Pieaugot tīmekļa lietotņu popularitātei, arī rīkam *ViziQuer* ir izstrādāta tīmekļa versija, kuras izstrāde ir aprakstīta 2016. gadā izstrādātajā kvalifikācijas darbā "*ViziQuer* rīka realizācija tīmekļa vidē" [15].

[3] https://scholar.google.com/

Darbā ir izstrādāta *ViziQuer* rīka tīmekļa versija, kas nodrošina pieeju plašākam lietotāju lokam atšķirībā no rīka darbvirsmas versijas. Tīmekļa versijas rīks ir veidots izmantojot platformu *Ajoo* un platformu *Meteor*, kas ir paredzētas tīmekļa un mobilo lietotņu izstrādei [15]. Darbā izstrādātā rīka funkcionalitāte ir kalpojusi par pamatu tālākai rīka attīstībai - ir izveidota iespēja lietotājam tīmekļa versijā veidot grafiskus vaicājumus izmantojot grafiskus elementus, kā arī ģenerēt atbilstošus tekstuālus SPARQL vaicājumus.

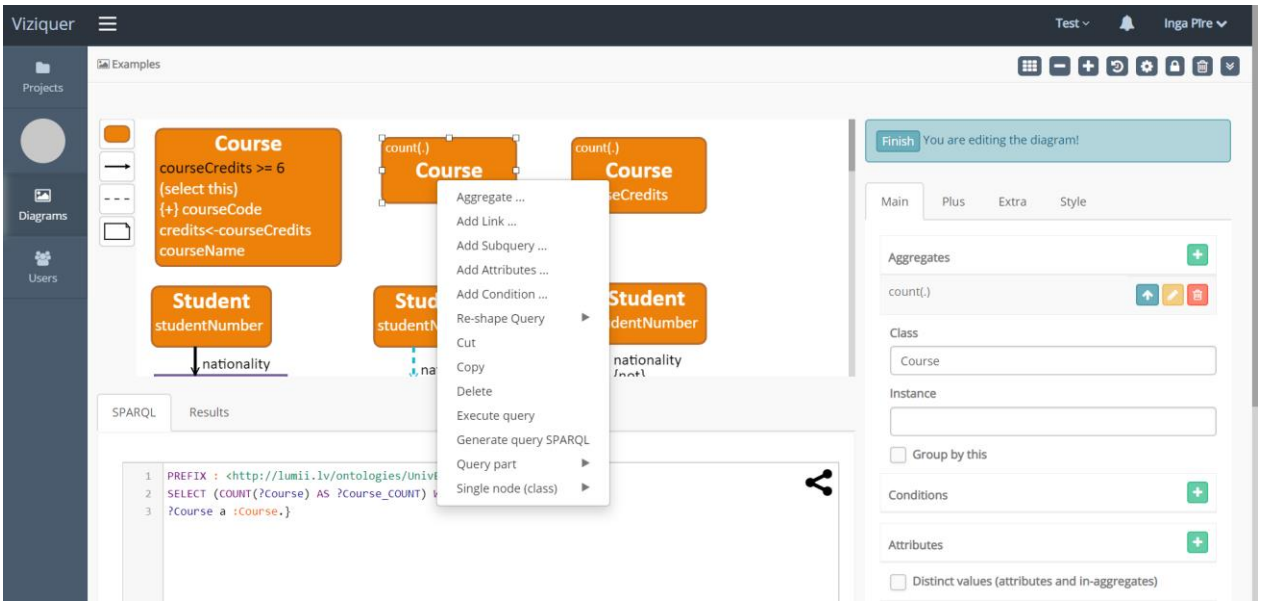
Savukārt 2018. gadā ir veikta tīmekļa rīka *ViziQuer* funkcionalitātes papildināšana, kas ir aprakstīta vairākos darbos [16] [17] [18] [19]. Vienā no darbiem ir izstrādāts izteiksmju sintakses priekšā teicējs, kura galvenā motivācija ir "atvieglot lietotāja darbu ar klašu nosacījumu izveidi" [16]. Divu citu darbu ietvaros ir veikta izstrādātā rīka lietojamības izpēte, kuras laikā tika pārbaudīta netehnisku studentu spēja veidot vizuālus SPARQL vaicājumus ar dažādu sarežģītības pakāpi [17] [18]. Abos darbos ir aplūkoti arī citi grafisko vaicājumu veidošanas rīki, taču visvairāk uzmanība ir pievērsta tādiem rīkiem kā *OptiqueVQS* un *QueryVOWL*. Pēc rīka *ViziQuer* un tam konkurējošo rīku apskates un analīzes, ir aprakstīti rīka *ViziQuer* trūkumi un potenciālie uzlabojumi. Turpinot papildināt rīku *ViziQuer*, kāda darba ietvaros papildus vizuālu SPARQL vaicājumu veidošanai ir izstrādāta iespēja veikt vizuālus *MySQL* vaicājumus vienas sistēmas ietvaros [19].

Eksistē vairāki SPARQL piekļuves punkti vaicājumu izpildei izmantojot SPARQL protokolu un HTTP pieprasījumus, tāpēc, lai noskaidrotu piekļuves punktu saderību ar rīku *ViziQuer*, vienā no darbiem ir veikta piekļuves punktu izpēte un tika secināts, ka saderību ietekmē piekļuves punkta klašu daudzums - liels klašu daudzums ietekmē rīka veiktspēju [20].

Lai uzlabotu lietotāju pieredzi izmantojot rīku *ViziQuer* lietotājiem, kuriem nav tehnisku zināšanu, liela nozīme ir arī rezultātu attēlošanas formātam. 2019. gadā ir izstrādāts maģistra darbs "Vizuālo vaicājumu rezultātu attēlošana gala lietotājam" [21]., kurā ir apskatīti dažādi datu attēlojumu un grafiku veidi, kā arī rīki un tehnoloģijas, kuru izmantošana nodrošinātu rezultātu attēlošanu grafiskā veidā. Darba rezultātā ir izveidots rīks, kas automātiski izvēlas piemērotāko grafika veidu atkarībā no vaicājuma rezultātiem, kā arī tas ir pieejams publiskā *Git* repozitorijā, tiesa gan darbā nav minēts par šī rīka integrācijas spējām ar rīku *ViziQuer* un var redzēt, ka pašreizējā rīka tīmekļa versijā šīs rīks nav integrēts un vaicājumu rezultāti tiek attēloti tikai tabulas veidā.

2.2 Lietotāja saskarne

ViziQuer rīks ir pieejams tīmeklī[4] un, lai varētu sākt ar to darboties, ir nepieciešams izveidot jaunu lietotāju. Pēc lietotāja izveidošanas var sākt veidot jaunus projektus rīkā, kurus ir iespējams inicializēt ar *Mini university* vai *Scholarly* datu shēmām. Pēc vizuālā vaicājuma izveidošanas ir iespējams ģenerēt no tā tekstuālu SPARQL vaicājumu, kas tiek izvadīts logā zem vizuālā vaicājuma. Var arī izvēlēties opciju izpildīt vaicājumu, kas ģenerē tekstuālu SPARQL vaicājumu un uzreiz to arī izpilda. Šobrīd notiek jaunas funkcionalitātes izstrāde – lietotājs var izmantot iepriekš sagatavotu tekstuālu SPARQL vaicājumu, to iekopēt vaicājuma logā un iegūt no tā vizuālu vaicājumu.



2.1. att. **Rīka *ViziQuer* lietotāju saskarne**

Attēlā (2.1. att.) var redzēt rīka *ViziQuer* lietotāju saskarni, kā arī augstāk aprakstītās opcijas vaicājuma izpildīšanai (*Execute query*) un tekstuāla SPARQL vaicājuma ģenerēšanai (*Generate query SPARQL*). Rīks ir papildināts ar iespēju veidot vizuālos vaicājumus no tekstuāliem vaicājumiem ar opciju "*visualizeSPARQL*", darba veidošanas laikā šī opcija vēl nebija iespējota publiski pieejamā rīka versijā, taču to var iespējot lokāli uzstādītai *ViziQuer* versijai.

[4] https://viziquer.lumii.lv/

2.3 Tekstuālu vaicājumu ģenerēšana

Apskatot rīka *ViziQuer* pirmkodu, tika secināts, ka tekstuāla vaicājuma ģenerēšana no vizuāla vaicājuma tiek veikta 3 galvenajos soļos:

1. lietotājs izveido vizuālu vaicājumu;
2. vizuālais vaicājums tiek reprezentēts rīkam specifiskā JSON formāta struktūrā;
3. lietotājam izvēloties opciju ģenerēt tekstuālu vaicājumu, no JSON tiek izveidots tekstuāls SPARQL vaicājums.

Tālāk šajā sadaļā tiek detalizētāk aprakstīti šie trīs galvenie soļi.

2.2. att. **Rīkā *ViziQuer* izveidota vizuāla vaicājuma piemērs**

Lietotājs izmantojot rīku *ViziQuer* var formulēt vaicājumu izmantojot vizuālus elementus. Piemērs šādam vizuālam vaicājumam ir redzams 2.2. att. Izpildot vizuālo vaicājumu, tiek atlasītas visas "*Patient*" klases instances, kurām īpašība "*gender*" satur vērtību "*female*", jeb - tiek atlasīti visi pacienti, kuri ir sievietes. Vizuālais vaicājums rīkā *ViziQuer* tiek reprezentēts izmantojot JSON objektu ar rīkam specifisku un abstraktu struktūru.

```
1  PREFIX : <http://lumii.lv/ontologies/2016/mini-bkus-en#>
2  SELECT ?surname ?personCode ?name ?gender ?birthDate WHERE{
3  ?Patient a :Patient.
4  OPTIONAL{?Patient :surname ?surname.}
5  OPTIONAL{?Patient :personCode ?personCode.}
6  OPTIONAL{?Patient :name ?name.}
7  OPTIONAL{?Patient :gender ?gender.}
8  OPTIONAL{?Patient :birthDate ?birthDate.}
9  FILTER(STR(?gender) = "female")}
```

2.3. att. **Tekstuāls SPARQL vaicājums**

Lietotājam izvēloties iespēju ģenerēt tekstuālu SPARQL vaicājumu, tas tiek izveidots apstaigājot JSON objektu, kas satur informāciju par vaicājuma vizuālo izskatu. 2.3. att. ir redzams 2.2. att. dotajam vizuālajam vaicājumam atbilstošs tekstuāls SPARQL vaicājums. Ja lietotājs neiegūst vēlamos rezultātus un vēlas pielāgot tekstuālo SPARQL vaicājumu, to var darīt veicot

izmaiņas uzģenerētajā vaicājumā. Pēc tam no pielāgotā tekstuālā vaicājuma var uzģenerēt jaunu vizuālo vaicājumu. Tekstuālu vaicājumu var arī vizualizēt iekopējot jaunu vaicājumu, nevis rediģējot esošu. Nākamajā darba apakšnodaļā tiek apskatīts, kā no tekstuāla vaicājuma tiek automātiski izveidots vizuāls vaicājums.

2.4 Vizuālu vaicājumu ģenerēšana

Rīkā *ViziQuer* vizuāla vaicājuma ģenerēšana no tekstuāla SPARQL vaicājuma tiek veikta 4 galvenajos soļos:

1. lietotājs izveido tekstuālu SPARQL vaicājumu un izvēlas opciju to vizualizēt;
2. izmantojot *JavaScript* programmēšanas valodā izstrādātu parsētāju *SPARQL.js*[5], tekstuālais SPARQL vaicājums tiek parsēts par JSON;
3. no parsētā JSON tiek izveidots JSON, kas ir rīkam *ViziQuer* specifiskā un abstraktā struktūrā;
4. iegūtais JSON tiek izmantots vaicājuma vizualizēšanā.

Tālāk šajā sadaļā tiek detalizētāk aprakstīti šie četri galvenie soļi.

```
{
  "queryType": "SELECT",
  "variables": [
    "?surname",
    "?personCode",
    "?name",
    "?gender",
    "?birthDate"
  ],
  "where": [
    {
      "type": "bgp",
      "triples": [
        {
          "subject": "?Patient",
          "predicate": "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
          "object": "http://lumii.lv/ontologies/2016/mini-bkus-en#Patient"
```

2.4. att. **JSON fragments no *SPARQL.js* bibliotēkas izvada**

Kad lietotājs ir izveidojis tekstuālu SPARQL vaicājumu, to ir iespējams rīkā *ViziQuer* vizualizēt. Izmantojot *JavaScript* bibliotēku *SPARQL.js*, tekstuālais SPARQL vaicājums tiek parsēts par JSON objektu. Fragments no bibliotēkas izveidotā JSON ir dots 2.4. att. Šis fragments atbilst augstāk 2.3. att. dotajam tekstuālajam SPARQL vaicājumam.

[5] https://github.com/RubenVerborgh/SPARQL.js/

Bibliotēkas *SPARQL.js* izveidotais JSON objekts neatbilst rīka *ViziQuer* specifikajai JSON struktūrai, kura tiek izmantota vizualizējot vaicājumus. Tāpēc tālāk apstaigājot JSON objektu, tas tiek kartēts par JSON objektu, kas ir rīkam *ViziQuer* atbilstošā struktūrā. Fragments no rīkam ViziQuer specifiska JSON objekta ir dots 2.5.att.

```
{
  "localName": "http://lumii.lv/ontologies/2016/mini-bkus-en#Patient",
  "identification": {
    "localName": "Patient",
    "URI": "http://lumii.lv/ontologies/2016/mini-bkus-en#Patient",
    "Namespace": "http://lumii.lv/ontologies/2016/mini-bkus-en#",
    "Prefix": "",
    "DefaultNamespace": "http://lumii.lv/ontologies/2016/mini-bkus-en#",
    "triplesMaps": [],
    "short_name": "Patient"
  },
  "instanceAlias": null,
  "isVariable": false,
  "isUnit": false,
  "isUnion": false,
  "conditions": [
    "str(gender) = \"female\""
  ],
  "fields": [
    {
      "alias": "",
```

2.5.att. **Fragments no rīkam *ViziQuer* specifiska JSON**

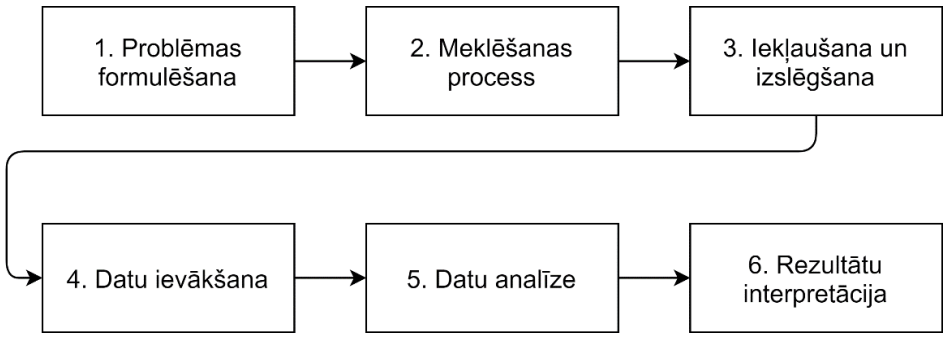
Attēlā 2.5.att. dotais *ViziQuer* JSON objekta fragments ir iegūts kartējot to no augstāk 2.4. att. dotā JSON fragmenta, kas iegūts izmantojot bibliotēku *SPARQL.js*. JSON formāta dati ar šādu struktūru tālāk tiek izmantoti veidojot vizuālo vaicājumu.

3. VIZUĀLO VAICĀJUMU VEIDOŠANAS RĪKI

Pastāv vairāki rīki, kas nodrošina līdzīgu funkcionalitāti kā iepriekšējā nodaļā apskatītais rīks *ViziQuer*. Darba apakšnodaļā 2.1 apskatot iepriekšējos pētījumus, kas veikti saistībā ar rīku *ViziQuer*, var redzēt, ka jau iepriekš ir veikta citu konkurējošu rīku salīdzināšana ar rīku *ViziQuer,* taču šie pētījumi ir veikti aptuveni pirms diviem gadiem, kas ir pietiekams laiks, lai tiktu papildināta esošo rīku funkcionalitāte, kā arī šajos darbos netiek aplūkots autorei interesējošais jautājums – vai kāds no rīkiem spēj veikt inverso pārveidošanu. Tajā pat laikā, ir iespējams, ka ir izstrādāti jauni rīki, tāpēc tika veikta sistemātiska literatūras apskate.

3.1 Literatūras apskates metode

Programmatūras inženierijas nozarē pieaugošā pētījumu skaita dēļ ir nepieciešama sistemātiska pieeja pētījumu rezultātu novērtēšanai un apkopošanai, lai nodrošinātu objektīvu izpētes kopsavilkumu par konkrētu tēmu [22]. Šajā darbā izmantotā pētījuma metode ir precīza meklēšanas rezultātu apskate. Tās mērķis ir nodrošināt caurredzamību un atkārtojamību, kā arī sniegt pierādījumus visiem izdarītajiem secinājumiem. Autore veica sistemātisku literatūras apskati, lai sniegtu objektīvu pārskatu par to, kas ir publicēts saistībā ar dažādiem SPARQL vaicājumu vizualizācijas rīkiem un atrastu atbildes uz izpētes jautājumiem.



3.1. att. **Sistemātiskas literatūras apskates soļi**

Sistemātiska literatūras apskate tika veikta izpildot vairākus secīgus soļus (skatīt 3.1. att.). Šie soļi tika iegūti no pētījuma par sistemātisku literatūras apskati programmatūras inženierijā, kur pats pētījums ir veikts izmantojot sistemātisku literatūras apskati [23]. Pēc pētījuma struktūras, tika noteikti šādi seši soļi, kas ir secīgi attēloti arī 3.1. att.:

1. problēmas formulēšana – tiek definēti izpētes jautājumi;

2. meklēšanas process – tiek identificēti raksti un pētījumi izmantojot meklēšanas terminus un meklētājprogrammas;
3. darbu iekļaušana un izslēgšana - tiek izvēlēti atbilstošie raksti un pētījumi;
4. datu ievākšana – tiek iegūti dati no atlasītajiem rakstiem un pētījumiem;
5. datu analīze – tiek aprakstīti iegūtie rezultāti;
6. rezultātu interpretācija.

Šie soļi tika izmantoti autores darbā, un tie ir aprakstīti nākamajās darba apakšnodaļās. Rezultātu interpretācija ir veikta darba rezultātu nodaļā.

3.2 Problēmas formulēšana

Balstoties uz nodaļas sākumā aprakstīto problēmu, tika definēti šādi izpētes jautājumi:

- Kādi vizuālu SPARQL vaicājumu veidošanas rīki pastāv?
- Kuri rīki nodrošina iespēju automātiski veidot vizuālus vaicājumus no tekstuāliem SPARQL vaicājumiem?
- Kā šī funkcionalitāte ir izstrādāta rīkos, kuri to nodrošina?

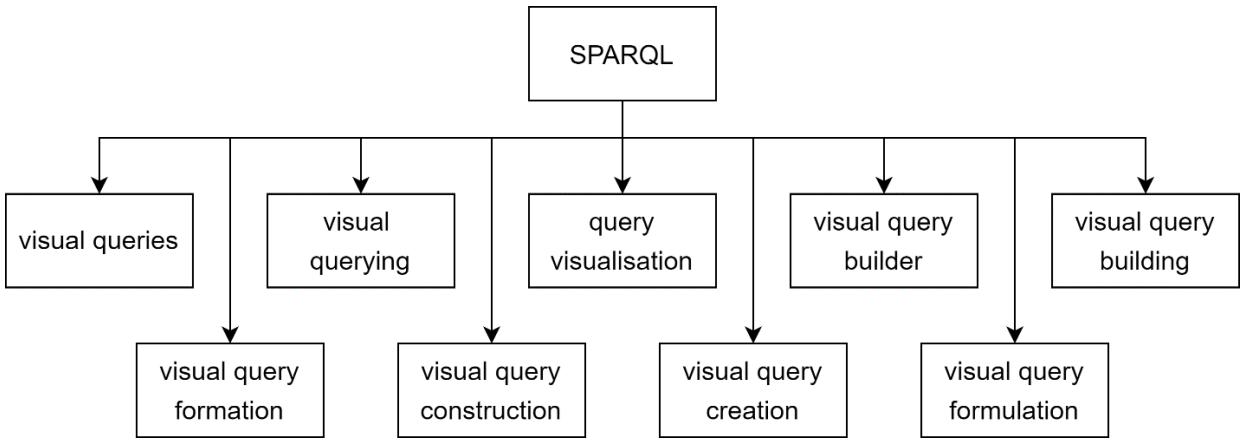
Lai atbildētu uz šiem jautājumiem, tika veikta literatūras apskate. Novērtēšana, vai kāds konkrēts rīks piedāvā funkcionalitāti, kas ļauj automātiski veidot vizuālus vaicājumus no tekstuāliem SPARQL vaicājumiem, tika veikta balstoties uz rakstos un pētījumos doto informāciju, bez praktiskas rīka funkcionalitātes apskates. Ja pēc konkrētā avotā dotās informācijas šis rīks nodrošina šo funkcionalitāti, tad papildus teorētiskam apskatam tika veikta arī praktiska rīka apskate, lai noskaidrotu, kā šī funkcionalitāte ir izstrādāta.

3.3 Meklēšanas process

Meklēšana tika veikta manuāli, izmantojot akadēmisko meklētājprogrammu *Google Scholar*[6]. Pirms meklēšanas procesa uzsākšanas tika noteikti būtiskie atslēgvārdi angļu valodā pēc kuriem veikt meklēšanu. Atslēgvārdi jeb meklēšanas termini, ir vārdi, kuri tiek ievadīti meklēšanas lodziņos. Tie raksturo galvenos pētījuma tēmas jēdzienus. Ja nav izvēlēti pareizie atslēgvārdi, var būt grūtības atrast vajadzīgos rakstus. Atlasot atslēgvārdus, tika veiktas šādas darbības: tēmas galveno atslēgvārdu identificēšana, sinonīmu un līdzīgu atslēgvārdu identificēšana.

[6] https://scholar.google.com/

Kā tēmas galvenie atslēgvārdi tika identificēti divi atslēgvārdi: "*SPARQL"* un "vizuāli vaicājumi" ("*visual queries*"). Atslēgvārdam "*SPARQL*" netika noteikti līdzīgi termini, taču meklēšanas terminam "vizuāli vaicājumi" tika noteikti šādi līdzīgi termini: vizuālā vaicāšana (*visual querying*), vaicājumu vizualizācija (*query visualisation*), vizuālu vaicājumu veidotājs (*visual query builder*), vizuālu vaicājumu veidošana (*visual query building, visual query formation*), vizuālu vaicājumu konstruēšana (*visual query construction*), vizuālu vaicājumu radīšana (*visual query creation*) un vizuālu vaicājumu formulēšana (*visual query formulation*).



3.2. att. **Meklēšanas terminu veidošana**

Rezultātā meklēšanas termini tika iegūti savienojot dotos atslēgvārdus, kā tas ir attēlots *3.2. att.* Meklētājprogrammā *Google Scholar* tika izvēlēta iespēja veikt izvērsto meklēšanu, un tika norādītas šādas meklēšanas opcijas rakstu meklēšanā:

- ar visiem vārdiem "*SPARQL*";
- un ar vismaz vienu no vārdiem "*visual queries*", "*visual querying*", "*query visualisation*" "*visual query builder*", "*visual query building*", "*visual query formation*", "*visual query construction*", "*visual query creation*", "*visual query formulation*";
- kā arī tika izvēlēta opcija, ka meklēšanas atslēgvārdi var būt kaut kur rakstā.

Sākotnējā meklēšanā netika ierobežots publicēšanas datums. Izmantojot šos meklēšanas iestatījumus, tika iegūti 479 meklēšanas rezultāti. Lai samazinātu meklēšanas rezultātu skaitu, tika veiktas izmaiņas meklēšanas opcijās - tika izvēlēts meklēt rakstus sākot no 2010. gada un izslēgt rakstus, kuri satur atslēgvārdu "*ViziQuer*". Izmantojot šos meklēšanas iestatījumus, tika iegūti 422 meklēšanas rezultāti, kuri tika sakārtoti pēc atbilstības un tālākai atlasei tika izvēlēti pirmie 30 meklēšanas rezultāti.

3.4 Iekļaušana un izslēgšana

Iekļaušanas un izslēgšanas procesa mērķis ir atlasīt atbilstošus rakstus no visiem meklēšanas rezultātos atrastajiem rakstiem. Atbilstošs raksts ir tāds, kas palīdz atbildēt uz vismaz vienu 3.2 apakšnodaļā definēto izpētes jautājumu. Lēmums, par to vai raksts ir atbilstošs vai nē tika pieņemts pēc anotācijas izlasīšanas, ja pēc anotācijas izlasīšanas netika iegūta pietiekama informācija, tika pārskatīts viss raksts. No meklēšanas rezultātiem nācās izslēgt ne tikai neatbilstošus rakstus, bet arī rakstus, kas neatbilda papildus šādiem kritērijiem:

- raksts ir angļu valodā;
- raksts ir pieejams bez maksas;

Pēc rakstu atlases tika iegūts saraksts ar 16 rakstiem turpmākai izpētei. No šiem rakstiem tika iegūti dati, kas nepieciešami, lai atbildētu uz pētījuma jautājumiem.

3.5 Datu ievākšana

Datu ievākšana ir strukturēta, pamatojoties uz nodaļas sākumā definētajiem izpētes jautājumiem. Tāpēc šajā sadaļā tiek uzskaitīti atrastie vizuālo *SPARQL* vaicājumu veidošanas rīki, kas atbild uz pirmo izpētes jautājumu. Pie katra no atrastā rīka tiek dota atbilde arī uz otro un trešo izpētes jautājumu – vai šis rīks nodrošina iespēju veikt inverso pārveidošanu, ja jā, tad kā šī funkcionalitāte ir izstrādāta.

Konduit VQB ir RDF vizuālo darbplūsmu veidošanas rīks priekš *Nepomuk-KDE*[7] datiem, kas nodrošina elastīgu piekļuvi lokāli esošajiem RDF datiem, kā arī tīmeklī bāzētiem datiem [24]. Vaicājumu veidošanas lietojumprogrammas saskarne ietver trīs galvenās daļas - centrālo daļu vizuālo vaicājumu veidošanai balstoties uz shēmu un divus priekšskatītājus ģenerētajam vaicājumam un tā rezultātiem. Abos apskatītajos pētījumos par rīku *Konduit VQB* [24] [25] ir minēts, ka to var izmantot SPARQL vaicājumu ģenerēšanai, taču netiek minēta spēja ģenerēt vizuālus vaicājumus no tekstuāliem vaicājumiem.

QueryVOWL ir vizuāla vaicājumu valoda, kas balstās uz ontoloģijas vizualizāciju VOWL un definē kartēšanu uz SPARQL, tās mērķis ir intuitīva un viegli lietojama valoda, kas vienlaikus saglabā elastīgumu un SPARQL ekspresivitāti [26]. Vaicājumus var izveidot pilnībā ar vizuāliem

[7] http://nepomuk.semanticdesktop.org/index.html

elementiem. Rīks ir brīvi pieejams tīmeklī[8]. Rīka attīstībai tiek apsvērts izstrādāt jaunu funkcionalitāti, kas vairāk atbalstītu lietotāju izpratni - dinamiski parādīti paskaidrojumi un vaicājuma daļu attēlošana dabiskajā valodā, taču netiek minēta spēja ģenerēt vizuālus vaicājumus no tekstuāliem vaicājumiem.

LODeX ir rīks, kas lietotājam nodrošina ērtu vaicājumu veidošanas veidu ātrā un interaktīvā manierē, atbalsta rezultātu pārlūkošanu un arī iespēju precizēt vaicājumu [27]. Arī šim rīkam netiek minēta spēja ģenerēt vizuālus vaicājumus, kā arī nav minēts par iespējamu šādas funkcionalitātes izstrādi nākotnē. Kā attīstības virzieni tiek minēts efektīvs shēmas apkopojums lielām datu kopām, jauni vaicājumu rezultātu formāti (diagramma, vienkāršs teksts), kā arī papildus SPARQL operatoru pievienošana (*GROUP BY* un *UNION*).

SparqlFilterFlow ļauj sastādīt SPARQL vaicājumus, izmantojot tikai grafiskus elementus un vienkāršas teksta virknes, vienlaikus izvairoties no jebkādas strukturētas teksta ievades [28]. Lai izmantotu rīku, lietotājam nav vajadzīgas zināšanas par semantiskā tīmekļa jēdzieniem, tikai pamata izpratne par RDF. Arī šim rīkam nākotnes plāni neietver vizuālu vaicājumu veidošanu no tekstuāliem vaicājumiem, bet gan uzsvars tiek likts uz jaunas funkcionalitātes izstrādi priekš vizuālo vaicājumu veidošanas, piemēram, nodrošināt atbalstu *DESCRIBE* un *CONSTRUCT* vaicājumu izveidošanai papildus esošajiem *SELECT* un *ASK* vaicājumiem. Tam būtu nepieciešams intuitīvs veids, kā attēlot grafa struktūru *CONSTRUCT* vaicājuma rezultātam.

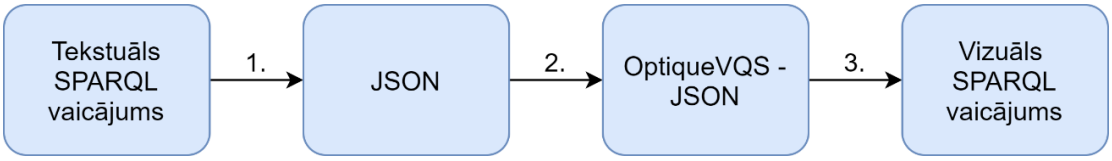
OptiqueVQS ir sistēma, kas nodrošina atbalstu lietotājam vizuālu vaicājumu formulēšanā un ļauj soli pa solim konstruēt vaicājumu ontoloģijā, katrā solī sniedzot lietotājam atbilstošu informāciju, lai turpinātu vaicājuma būvniecību [29]. *OptiqueVQS* izmanto uz logrīkiem balstītu arhitektūru, kā arī vairākas attēlošanas un mijiedarbības paradigmas. Maģistra darbā, kas izstrādāts Oslo Universitātē tiek aprakstīta rīka *OptiqueVQS* papildināšana ar jaunu funkcionalitāti – SPARQL tekstuālu vaicājumu pārveidošana par vizuāliem vaicājumiem [30].

Darbā kā galvenais mērķis šīs funkcionalitātes izstrādei rīkā *OptiqueVQS* ir minēts efektīvas sadarbības nodrošināšana starp domēna ekspertiem un tehniskajiem speciālistiem [30]. Domēna eksperti var viegli izprast jēdzienus un attiecības, kuras ir attēlotas ar vizuāliem elementiem, un izmantot vizuālos elementus, lai formulētu vaicājumus. Gadījumā, ja domēna eksperti ir izveidojuši vaicājumu atbilstoši viņu prasībām, taču netiek iegūti vēlamie rezultāti, tehniskie speciālisti var izpētīt tekstuālo vaicājumu un labot to. Šādam risinājumam ir vairākas priekšrocības - tehniskie speciālisti bieži vien spēj labāk orientēties tekstuālos SPARQL vaicājumos nekā

[8] http://www.visualdataweb.de/webvowl/#

vizuālos, tekstuālos vaicājumus ir viegli pielāgot, kā arī tos ir vieglāk kopīgot nekā vizuālos vaicājumus. Tādā veidā starp domēna ekspertiem un tehniskajiem speciālistiem tiek nodrošināta efektīva sadarbība vaicājuma formulēšanas procesā, lai tiktu sasniegts vēlamais rezultāts. Aplūkotā darba mērķis ir izstrādāt funkcionalitāti, kas nodrošinātu sinhronizāciju starp tekstuālo un vizuālo vaicājumu visā vaicājuma formulēšanas procesā [30].

Pirms izmaiņu uzsākšanas, rīkā bija iespējams ģenerēt SPARQL vaicājumus no vizuāliem vaicājumiem, taču vizuālo vaicājumu ģenerēšana no tekstuāliem SPARQL vaicājumiem vēl nebija iespējama. SPARQL vaicājums ir simbolu virkne, ko nav viegli apstrādāt, tāpēc pirms to var pārveidot atpakaļ par vizuālo vaicājumu, to ir nepieciešams parsēt ērtākā formātā. Tā kā lielākā daļa *OptiqueVQS* loģikas jau bija izstrādāta klienta pusē, arī parsētājs, kas spēj parsēt SPARQL vaicājumus par JSON, tika implementēts klienta pusē [30]. Tika izmantots *Ruben Verborgh* veidotais SPARQL parsētājs[9], kas izstrādāts programmēšanas valodā *JavaScript*. Šis parsētājs dotam SPARQL vaicājumam veido atbilstošu JSON objektu, kas satur visu informāciju par vaicājumu. Rīks *OptiqueVQS* vaicājumu attēlošanai izmanto JSON objektu, kura struktūra ir rīkam specifiska, līdz ar to, lai sasniegtu mērķi, šī darba galvenais uzdevums bija izpētīt iespējas pārveidot JSON objektu, kas iegūts no SPARQL parsētāja, par JSON objektu ar atbilstošu struktūru rīkam *OptiqueVQS* [30].



3.3. att. **Soļi vizuāla vaicājuma ģenerēšanai no tekstuāla vaicājuma rīkā *OptiqueVQS***

Līdz ar to algoritms, kas tika izstrādāts vizuālo vaicājumu ģenerēšanai no tekstuāliem SPARQL vaicājumiem, sastāv no trīs galvenajiem soļiem (skatīt 3.3. att.):

1. izmantojot parsētāju, tekstuālais SPARQL vaicājums tiek pārveidots par JSON;
2. JSON tiek pārveidots par *OptiqueVQS* JSON;
3. no *OptiqueVQS* JSON tiek izveidots vizuāls SPARQL vaicājums.

Rīks ***Gruff*** ir izstrādāts, lai atvieglotu vairāku datu kopu izpēti [31]. *Gruff* ir uzlabota grafiskā lietotāju saskarne darbam ar RDF datiem. Viena no rīka *Gruff* jaunākajām funkcijām ir grafiskais vaicājuma skats, kas ļauj veidot vaicājumus kā mezglu un saišu diagrammas, pēc vaicājuma

[9] https://github.com/RubenVerborgh/SPARQL.js

diagrammas izveidošanas, *Gruff* uzģenerē SPARQL vai *Prolog* kodu vaicājumam [31]. *Gruff* ir pieejams bez maksas[10].

RDF Explorer ir sistēma, kas ļauj lietotājiem, vienlaikus orientēties un meklēt informāciju grafā, pakāpeniski izpētot datu kopu [32]. Rīks ir pieejams tīmeklī[11] un pašlaik tas ir konfigurēts piekļuvei *Wikidata* piekļuves punktam. Tīmeklī pieejamā rīkā var redzēt, ka tas šobrīd neatbalsta uzģenerētā SPARQL vaicājuma attēlošanu.

Rīka ***PepeSearch*** galvenais mērķis ir atvieglot piekļuvi saistītajiem atvērtajiem datiem plašākai sabiedrībai [33]. Šobrīd rīks atbalsta vaicājumu formulēšanu Norvēģijas entītiju reģistra datu kopai, taču tiek plānots nodrošināt rīka spēju darboties ar dažādām saistīto atvērto datu krātuvēm [33]. Pēc publikācijā dotajiem ekrānuzņēmumiem no rīka, var redzēt, ka tas neizvada uzģenerēto vaicājumu.

FedViz ir uz pārlūku balstīta lietojumprogramma, kas nodrošina lietotājiem elastīgu vizuālo saskarni, lai formulētu un izpildītu SPARQL vaicājumus [34]. FedViz atbalsta pilnu SPARQL1.1 konstrukciju klāstu, kuras var atlasīt izmantojot saskarni vai arī pievienot vaicājumam SPARQL redaktorā [34].

Rīks ***EXPLO-RDF*** atbalsta divus uzdevumus - automātisku shēmas uzbūvi apkopojot izmantoto RDF datu kopu paplašinātas UML klases diagrammas formā, un nodrošina palīdzību lietotājam SPARQL vaicājumu sastādīšanā [35].

3.6 Datu analīze

Tika aplūkoti 10 dažādi vaicājumu vizualizācijas rīki. Rezultātā tika noskaidrots, ka tikai viens no 10 aplūkotajiem rīkiem ir spēris soli pretim vizuālo vaicājumu ģenerēšanai no tekstuāliem SPARQL vaicājumiem. Pārējiem deviņiem apskatītajiem rīkiem šādas funkcionalitātes izstrāde nav minēta arī nākotnes plānos. Kā attīstības virzieni tika minēta jaunas funkcionalitātes izstrāde, kas vairāk atbalstītu lietotāju izpratni, piemēram, rīks *QueryVOWL* apsver vaicājuma daļu attēlošana dabiskajā valodā, atbalsta nodrošināšana jaunām SPARQL vaicājumvalodas iespējām, piemēram, rīks *SparqlFilterFlow* plāno nodrošināt atbalstu *DESCRIBE* un *CONSTRUCT* vaicājumu izveidošanai papildus esošajiem *SELECT* un *ASK* vaicājumiem.

[10] https://allegrograph.com/products/gruff/
[11] https://www.rdfexplorer.org/

4. VIZUĀLO VAICĀJUMU ĢENERĒŠANA RĪKĀ VIZIQUER

Pastāv vairāki rīki, kas ļauj ģenerēt programmatūras kodu dažādās programmēšanas valodās no modeļiem. Piemēram, šāds rīks ir *Acceleo*[12], kas pirmkodu ģenerē balstoties uz veidnēm. Šādi rīki tiek izmantoti modeļa virzītā izstrādē, kur rīka lietotājs (programmētājs) izveido vispirms modeli kādā modelēšanas valodā, kā piemēram UML, un pēc tam izmantojot šos rīkus ģenerē atbilstošu kodu, kuru tālāk var pielāgot atbilstoši prasībām. Kaut arī pastāv vairāki koda ģenerēšanas rīki, eksistē tikai daži rīki, ar kuriem iespējams izveidot modeli no programmatūras koda. Viens no šādiem rīkiem ir *Visual Paradigm*[13]. Izmantojot šo rīku, lietotājs var ģenerēt *Java* pirmkodu no UML klašu diagrammas un veicot izmaiņas kodā, redzēt veiktās izmaiņas arī UML modelī. Šāda pieeja nodrošina, ka *Java* pirmkods un programmatūras projektējums tiek sinhronizēti.

Ir pieejami arī vairāki rīki SQL vizuālo vaicājumu formulēšanai. Kā piemērs šādam rīkam ir *DbVisualizer*[14], kas nodrošina arī iespēju rekonstruēt *SELECT* vaicājumus. Pastāv arī dažādi rīki vizuālu SPARQL vaicājumu veidošanai, kas detalizētāk ir apskatīti darba iepriekšējā nodaļā. Veicot dažādu rīku aplūkošanu, tika secināts, ka tikai viens rīks no visiem aplūkotajiem spēj ģenerēt vizuālus vaicājumus no tekstuāliem SPARQL vaicājumiem – rīks *OptiqueVQS*.

Vairāki rīki nodrošina iespēju ģenerēt kodu un vaicājumus no vizuāliem elementiem, taču daudz mazāk ir tādu rīku, kas spēj to paveikt arī pretējā virzienā. Rīks *ViziQuer* vēl ir izstrādes stadijā, un tas tiek nepārtraukti uzlabots. Arī vizualizācijas modulis, kurš nodrošina vizuālo vaicājumu ģenerēšanu no tekstuāliem SPARQL vaicājumiem, ir izstrādes stadijā. Darba ietvaros tika veikta sistemātiska analīze par SPARQL vaicājumu vizualizācijas iespējām *ViziQuer* notācijā, tajā skaitā novērtējot, kādus vaicājumus ir iespējams veiksmīgi vizualizēt un kādus vēl nav iespējams veiksmīgi vizualizēt. Šajā darba nodaļā tiek izdalītas vairākas vaicājumu klases un, balstoties uz vaicājumu dalījumu klasēs, tiek aplūkoti vairāki konkrētai klasei atbilstoši vaicājumi un tiek noskaidrots, vai to vizualizācija ir veiksmīga.

[12] https://www.eclipse.org/acceleo/
[13] https://www.visual-paradigm.com/
[14] https://www.dbvis.com/

4.1 SPARQL vaicājumu dalījums klasēs

Veicot sistemātisku analīzi par SPARQL vaicājumu vizualizācijas iespējām *ViziQuer* notācijā, tika izdalītas vairākas vaicājumu klases. Kā par pamatu vaicājumu dalījumam klasēs tika izmantots *OptiqueVQS* rakstā [36] dotais sadalījums, kas tika papildināts balstoties uz SPARQL 1.1 vaicājumvalodas specifikāciju [11]. Kopā tika izdalītas sešas vaicājumu klases – konjunktīvi vaicājumi, disjunktīvi vaicājumi, vaicājumi ar apkopošanas funkcijām, vaicājumi ar noliegumu, vaicājumi ar risinājumu virknes modifikatoriem un vaicājumi ar apakšvaicājumiem. Šo klašu detalizētāks apraksts ir dots 4.1. tabula.

4.1. tabula

SPARQL vaicājumu veidi

Vaicājuma veids	**Apraksts**	**SPARQL sintakse**
Konjunktīvi vaicājumi	Dotā vaicājuma atomi ir savienoti ar UN operāciju, risinājums ir vairāku risinājumu kopu šķēlums.	SPARQL vaicājumi ar pamata grafu šabloniem, grupas grafu šabloniem un klauzulu *FILTER*.
Disjunktīvi vaicājumi	Daži dotā vaicājuma atomi ir savienoti ar VAI operāciju, risinājums ir vairāku risinājumu kopu apvienojums.	SPARQL vaicājumi ar vairākiem iespējamiem grafu šabloniem izmantojot *OPTIONAL* un alternatīviem grafu šabloniem izmantojot *UNION*.
Vaicājumi ar apkopošanas funkcijām	Vairākas vērtības tiek sagrupētas, lai veidotu vienu vērtību.	SPARQL vaicājumi, kas ietver tādas apkopošanas funkcijas kā *MIN*, *MAX*, *AVG*, *SUM*, kā arī *GROUP BY* un *HAVING* klauzulas, jo tās bieži vien tiek izmantotas kopā ar apkopošanas funkcijām.
Vaicājumi ar noliegumu	Vaicājumi, kuros tiek veikta pārbaude, vai konkrēts trijnieks eksistē vai neeksistē datu grafā.	SPARQL vaicājumi, kas ietver tādus operatorus kā *NOT EXISTS* un *MINUS*.

Vaicājuma veids	Apraksts	SPARQL sintakse
Vaicājumi ar risinājumu virknes modifikatoriem	Vaicājumi, kuros tiek modificēta risinājumu virkne.	SPARQL vaicājumi, kas ietver *ORDER BY*, *DISTINCT*, *REDUCED*, *OFFSET* un *LIMIT*.
Vaicājumi ar apakšvaicājumiem	SPARQL vaicājumi, kas satur citus vaicājumus.	*SELECT* vaicājums, kurš ietver vienu vai vairākus citus *SELECT* vaicājumus.

Šo vaicājumu klašu vizualizācijas iespējas *ViziQuer* notācijā detalizētāk tiek apskatītas nākamajās darba apakšnodaļās. Apakšnodaļās tiek doti vairāki vaicājumu piemēri, šie piemēri ir veidoti tā, lai daļa no tiem būtu sākotnēji veidoti kā tekstuāli SPARQL vaicājumi un daļa būtu sākotnēji veidoti kā vizuāli vaicājumi rīkā *ViziQuer*, no kuriem tālāk ir uzģenerēti atbilstoši tekstuāli SPARQL vaicājumi. Vaicājumu piemēri ir veidoti šādi, lai varētu secināt, vai spēju vizualizēt izveidotos vaicājumus ietekmē tas, ka vaicājums sākotnēji ir veidots kā vizuāls vaicājums izmantojot *ViziQuer* notāciju un pēc tam pārveidots par tekstuālu SPARQL vaicājumu. Visi vaicājumu piemēri ir veidoti izmantojot *ViziQuer* tīmekļa vietnē[15] doto *Mini-university* shēmu. Tika apskatīta tikai vaicājuma forma *SELECT* un netika apskatītas citas vaicājuma formas kā *CONSTRUCT*, *ASK*, *DESCRIBE*, jo tā ir vienīgā vaicājumu forma, ko atbalsta rīks *ViziQuer*.

4.2 Konjunktīvi vaicājumi

SPARQL pamatā ir atbilstības noteikšana grafu šabloniem (*graph pattern*). Sarežģītākus šablonus var iegūt dažādos veidos kombinējot mazākus un vienkāršākus šablonus. SPARQL 1.1 vaicājumvalodas specifikācijā [11] ir minēti vairāki šablonu kombinēšanas veidi. Viens no šablonu kombinēšanas veidiem ir pamata grafu šabloni (*basic graph patterns*) [11], kas ir trijnieku šablonu virkne kopā ar filtriem, kur katrs trijnieku šablons ir atdalīts ar punktu. Vēl viens no šablonu kombinēšanas veidiem ir grupas grafu šabloni (*group graph patterns*) [11], kur vienā grupā var būt viens vai vairāki pamata grafu šabloni. Grupas grafu šabloni tiek atdalīti ar figūriekavām.

Tā kā pamata grafa šablons ir trijnieku šablonu virkne kopā ar filtriem, šajā apakšnodaļā tiek apskatīta arī klauzula *FILTER*. Izmantojot *FILTER* klauzulu ir iespējams ierobežot risinājumus, atlasot tikai tos, kuriem norādītā izteiksme izpildās kā patiesa [11].

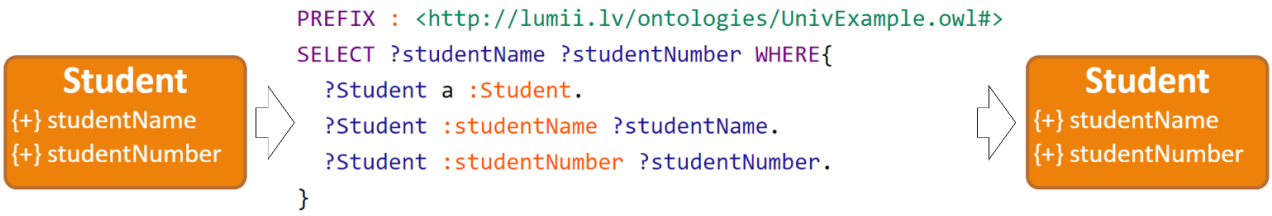
[15] https://viziquer.lumii.lv/

SPARQL vaicājumvalodā nav tāds atslēgvārds, kas norādītu konjunkciju (“un” operāciju) starp grafu šabloniem. Konjunktīvi grafa šabloni ir izvietoti blakus viens otram un ietverti figūriekavās, tādējādi veidojot grupas grafa šablonu. Grupas grafa šabloni var saturēt arī citas konstrukcijas [11], taču šajā apakšnodaļā tiek aplūkoti grupas grafa šabloni, kas sastāv tikai no pamata grafa šabloniem un tiek interpretēti konjunktīvi.

Šajā apakšnodaļā tiek aplūkoti šādi izveidotie vaicājumi ar vienu grupas grafa šablonu, kā arī rīka *ViziQuer* spēja vizualizēt šos vaicājumus:

- tekstuāls vaicājums, kurš automātiski izveidots no vizuāla vaicājuma un dati tiek atlasīti no vienas klases;
- tekstuāls vaicājums ar datu atlasi no vienas klases;
- tekstuāls vaicājums, kurš automātiski izveidots no vizuāla vaicājuma un atlasa datus no vairākām klasēm;
- tekstuāls vaicājums, kurš atlasa datus no vairākām klasēm.

Vispirms tika apskatīta rīka spēja vizualizēt tekstuālu SPARQL vaicājumu ar grupas grafa šablonu, kas sākotnēji ir veidots kā vizuāls vaicājums rīkā *ViziQuer*, pēc tam no tā ir ģenerēts tekstuāls SPARQL vaicājums, no kura atkal ir izveidots vizuāls vaicājums.



4.1. att. **Vaicājums ar grupas grafa šablonu, kas sākotnēji veidots kā vizuāls vaicājums**

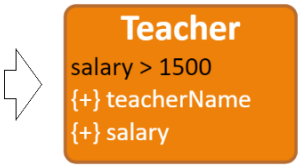
Vaicājums ar grupas grafu šablonu ir redzams 4.1. att. Attēla kreisajā pusē ir dots manuāli izveidotais vizuālais vaicājums, kuram tālāk seko rīkā *ViziQuer* automātiski veidots tekstuāls vaicājums, savukārt tālāk pēc tekstuālā vaicājuma seko no tā automātiski izveidotais vizuālais vaicājums. Tekstuālajā vaicājumā var redzēt, ka ir dota grupa ar vienu pamata grafu šablonu, kurš satur trīs trijnieku šablonus. Visi trijnieku šabloni ir doti viens otram blakus bez citām konstrukcijām, līdz ar to vaicājums ir konjunktīvs. Izpildot vaicājumu tiek atlasīta tāda informācija par studentu kā studenta vārds un numurs.

Manuāli veidotajā vizuālajā vaicājumā, kas dots 4.1. att. kreisajā pusē, ar oranžu taisnstūri ir attēlots mezgls ar klases nosaukumu *Student*, kas nozīmē, ka, izpildot vaicājumu, tiks atlasīti dati no šīs klases instancēm. Zem klases nosaukuma ir doti izvēlētie atribūti, kas nosaka, kāda

informācija tiks izvadīta izpildot vaicājumu. Atbilstoši šim vaicājumam – studenta vārds un numurs. Labajā pusē 4.1. att. dotais automātiski veidotais vizuālais vaicājums ir iegūts izejot pilnu ceļu (vizuāls vaicājums - tekstuāls SPARQL vaicājums - vizuāls vaicājums) un var redzēt, ka tas ir tāds pats kā sākotnējais vaicājums – attēlā pa kreisi esošais vaicājums. Līdz ar to var secināt, ka rīks *ViziQuer* spēj šādus triviālus vaicājumus vizualizēt pareizi.

Nākamais vaicājums, kas tika apskatīts, sākotnēji netika vizualizēts rīkā *ViziQuer*, bet gan izveidots kā tekstuāls SPARQL vaicājums. Vaicājums ir redzams 4.2. att. Attēla kreisajā pusē ir dots tekstuālais SPARQL vaicājums, savukārt labajā pusē ir dots uzģenerētais vizuālais vaicājums. Šajā vaicājumā ir dota grupa ar vienu pamata grafu šablonu, taču atšķirībā no iepriekšējā vaicājuma, šajā vaicājumā pamata grafa šablons ir dots kopā ar klauzulu *FILTER*. Vaicājumā tiek atlasīta tāda informācija kā pasniedzēja vārds un alga tikai par tiem pasniedzējiem, kuru alga ir lielāka par 1500.

```
PREFIX : <http://lumii.lv/ontologies/UnivExample.owl#>
SELECT ?teacherName ?salary WHERE{
  ?Teacher a :Teacher.
  ?Teacher :teacherName ?teacherName.
  ?Teacher :salary ?salary.
  FILTER(?salary > 1500)
}
```

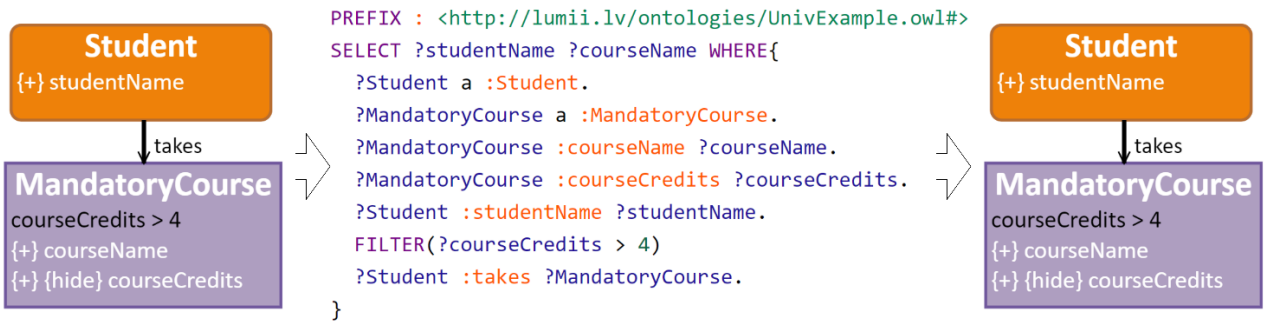


4.2. att. **Vaicājums ar grupas grafa šablonu, kas sākotnēji veidots kā tekstuāls vaicājums**

Vizuālais vaicājumus, kas redzams 4.2. att. labajā pusē, ir iegūts ģenerējot to no tekstuālā SPARQL vaicājuma, un kā redzams, tas ir izveidots pareizi – vizuālajā vaicājumā ir attēlota klase *Teacher*, zem klases nosaukuma ar melnu fontu ir dots nosacījums, kas tekstuālajā vaicājumā ir dots pie *FILTER* klauzulas. Zem nosacījuma tālāk seko atribūtu saraksts, kurā visi atribūti ir norādīti kā obligāti ar "+" simbolu.

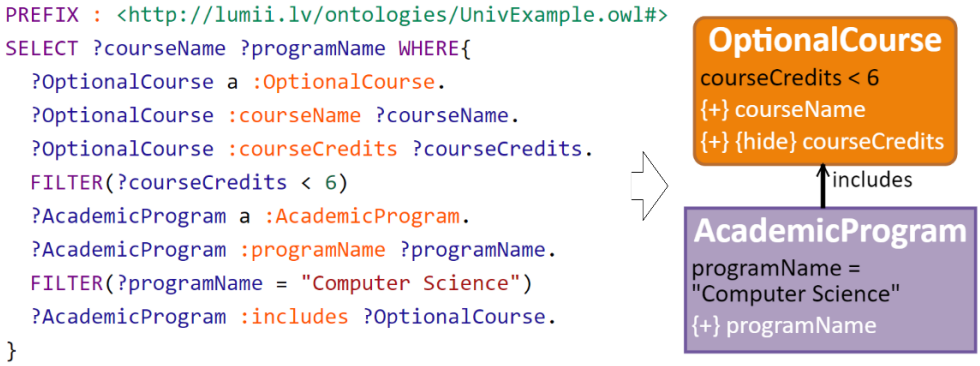
Rīkā *ViziQuer* vizuālo vaicājumu definēšana ir balstīta uz modeli, kas satur entītiju vārdnīcu un informāciju par shēmu, kā klašu īpašību piemērojamību, secību un kardinalitāti [37]. Vizuālie vaicājumi ir UML klašu diagrammas stila grafi ar mezgliem, kas apzīmē datu instances, un šķautnēm, kas apraksta attiecības starp mezgliem [37]. Rīks *ViziQuer* piedāvā notāciju vairāku datu instanču saistīšanai, tālākajos piemēros tiks apskatīta saites izmantošana starp vairākām klasēm.

Vaicājums ar saiti starp divām klasēm ir redzams 4.3. att. Attēla kreisajā pusē ir dots sākotnējais vizuālais vaicājums, no tā ģenerētais tekstuālais SPARQL vaicājums ir redzams pa vidu, taču kreisajā pusē ir redzams no tekstuālā vaicājuma ģenerētais vizuālais vaicājums. Šis vaicājums ir konjunktīvs vaicājums ar filtrēšanu. Tekstuālajā vaicājumā attiecība starp divām klasēm ir norādīta pēdējā trijnieku šablonā.



4.3. att. **Vaicājums ar savienošanas saiti, kas sākotnēji veidots kā vizuāls vaicājums**

Izpildot 4.3. att. doto vaicājumu tiek atlasīti studentu vārdi kopā ar obligāto kursu nosaukumiem, kur obligātajiem kursiem kredītpunktu skaits ir lielāks par 4. Kā redzams, arī šajā gadījumā vaicājums tiek vizualizēts pareizi un sakrīt ar sākotnējo vaicājumu. Saite vizuālajā vaicājumā ir attēlota kā bulta starp divām klasēm – *Student* un *MandatoryCourse*, kā arī pie bultas ir norādīts saites nosaukums – *takes*. Pie klases *MandatoryCourse* zem tās nosaukuma ar melnu fontu ir norādīts arī nosacījums, ka kursa kredītpunktu skaitam ir jābūt lielākam par 4.



4.4. att. **Vaicājums ar savienošanas saiti, kas sākotnēji veidots kā tekstuāls vaicājums**

Nākamais vaicājums, kas tika apskatīts ar saiti starp vairākām klasēm, sākotnēji tika izveidots kā tekstuāls SPARQL vaicājums. Izveidotais tekstuālais vaicājums ir dots 4.4. att. Vaicājumā tiek atlasīti neobligāto kursu nosaukumi kopā ar akadēmisko programmu nosaukumiem tiem neobligātajiem kursiem, kuriem kredītpunktu skaits ir mazāks par 6 un kuri ir ietverti datorzinātņu

akadēmiskajā programmā. Attēlā pa kreisi redzamais vizuālais vaicājums ir ģenerēts no tekstuāla SPARQL vaicājuma un tas ir vizualizēts pareizi – tajā ir attēlota saite ar nosaukumu *includes* starp divām klasēm – *OptionalCourse* un *AcadmicProgram*. Kā arī pie abām klasēm ir pareizi norādīti abi filtrēšanas nosacījumi.

Šajā apakšnodaļā tika apskatīti četri izveidotie konjunktīvie vaicājumi, gan bez filtrēšanas, gan ar, kā arī rīka *ViziQuer* spēja tos vizualizēt. Tika aplūkota arī instanču atlase no vairākām klasēm. Visi izveidotie vaicājumi tika vizualizēti veiksmīgi, neatkarīgi no tā, vai tie sākotnēji bija veidoti kā tekstuāli vai vizuāli vaicājumi.

4.3 Disjunktīvi vaicājumi

Iepriekšējā apakšnodaļā apskatītie grafu šabloni ļauj veikt vaicājumus, kur informācijai RDF grafā ir pilnībā jāatbilst vaicājumā dotajiem grafu šabloniem, lai tā tiktu uzskatīta par vaicājuma risinājumu. Taču RDF grafiem nav noteikta shēma un tie bieži vien satur neregulāras un nepilnīgas datu struktūras [11]. Šādās situācijās ir noderīgi, ja var izveidot tādus vaicājumus, kuros informācija tiek pievienota risinājuma kopai arī tad, ja risinājums neatbilst kādam vaicājuma šablonam. Atslēgvārds *OPTIONAL* nodrošina, ka risinājums netiek izslēgts no risinājumu kopas, ja tas neatbilst kādai neobligātai vaicājuma daļai [11]. Šīs konstrukcijas galvenais mērķis ir nodrošināt iespēju atgriezt datus, ja tādi eksistē, un ignorēt, ja tādi neeksistē.

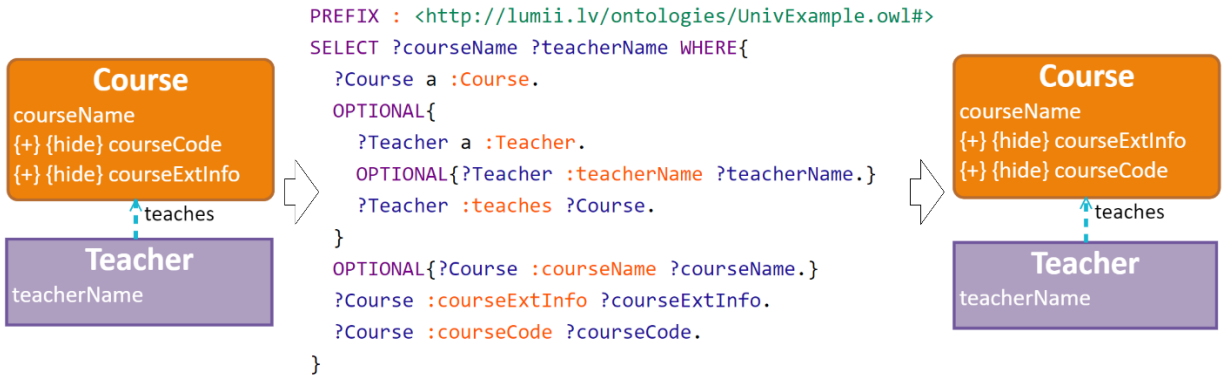
Izmantojot SPARQL vaicājumvalodu ir iespējams ne tikai norādīt neobligātus grafa šablonus, bet var arī norādīt šablonu alternatīvas. SPARQL vaicājumu valoda nodrošina iespēju apvienot rezultātus, kas atbilst vairākiem dažādiem grafu šabloniem. Rezultātu kopā tiek iekļauti visi risinājumi, kas atbilst vismaz vienam grafa šablonam. Šablonu alternatīvas sintaktiski tiek norādītas ar atslēgvārdu *UNION* [11].

Šajā apakšnodaļā tiek apskatīti 5 izveidotie disjunktīvie vaicājumi. No tiem trīs izveidotie vaicājumi ar *OPTIONAL*, no kuriem viens sākotnēji ir veidots kā vizuāls vaicājums un divi kā tekstuāli vaicājumi. Kā arī tiek apskatīti divi vaicājumi ar *UNION*, no kuriem sākotnēji viens ir veidots kā vizuāls un viens kā tekstuāls vaicājums. Vizuālajos vaicājumos starp klasēm esošās saites var būt neobligātas, tāpēc šajā apakšnodaļā apskatītajos vaicājumos tiks iekļauta arī datu atlase no vairākām klasēm ar neobligātām saitēm starp tām.

Analizējot rīka spēju vizualizēt disjunktīvus vaicājumus, vispirms tika apskatīts vaicājums ar *OPTIONAL*, kas sākotnēji ir veidots kā vizuāls vaicājums rīkā *ViziQuer*. Rīkā *ViziQuer* neobligātas

vaicājuma šablona daļas ir iespējams norādīt izvēloties atribūtus pie klases. Izvēloties atlasāmos atribūtus, pēc noklusējuma tiek pieņemts, ka tie nav obligāti. Ja kāds no atribūtiem ir obligāts, tad tas ir jāatzīmē, un vizuālajā vaicājumā tas tiks attēlots ar "+" simbolu. Arī veidojot saiti starp vaicājuma klasēm, to ir iespējams norādīt kā neobligātu.

Vaicājums ar neobligātiem grafu šabloniem ir dots 4.5. att. Vaicājumā neobligātie grafu šabloni ar atslēgvārdu *OPTIONAL* ir norādīti vairākās vietās. Izpildot vaicājumu tiek atlasīts kursa nosaukums kopā ar pasniedzēja vārdu visiem kursiem, kuriem ir norādīts kursa kods un papildus informācija par kursu. Viena no neobligātajām vaicājuma daļām ir, ka kursam jābūt norādītam nosaukumam, jeb risinājumu kopā tiks iekļauti arī tādi kursi, kuriem nosaukums nav norādīts. Vizuālajā vaicājumā obligātos un neobligātos atribūtus var atšķirt pēc pazīmes – pie obligātajiem atribūtiem figūriekavās ir norādīts "+" simbols, taču pie neobligātajiem atribūtiem šāda simbola nav.

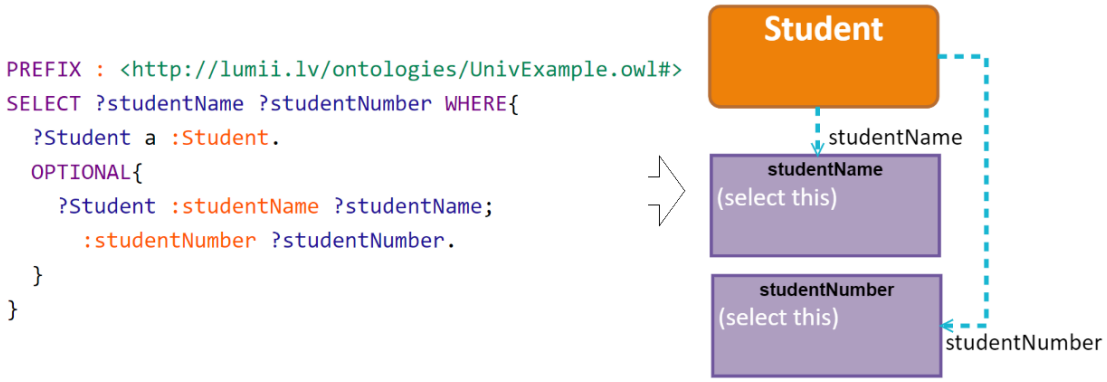


4.5. att. **Vaicājums ar *OPTIONAL*, kas sākotnēji veidots kā vizuāls vaicājums**

Iepriekšējā apakšnodaļā 4.2 tika aplūkoti vaicājumi, kuros tika atlasīti dati no dažādām klasēm, kā arī viens no saišu veidiem starp klasēm. Arī šajā vizuālajā vaicājumā, kurš redzams 4.5. att., ir dota savienošanas saite, taču tā ir neobligāta. Neobligātā saite ir dota starp *Course* un *Teacher* klasēm ar raustītu, gaiši zilu līniju ar bultu galā. Tas nozīmē, ka kurss tiks iekļauts risinājumu kopā arī tad, ja tam nebūs dots pasniedzējs, kurš pasniedz šo kursu. Labajā pusē 4.5. att. dotais vaicājums ir ģenerēts no tekstuāla SPARQL vaicājuma, kas sākotnēji ir veidots no vizuāla vaicājuma. Uzģenerētais vizuālais vaicājums ir pareizs, un tas sakrīt ar sākotnējo vizuālo vaicājumu, neskatoties uz vienu nebūtisku atšķirību – atribūtu kārtību klasē *Course*.

Nākamais vaicājums, kas tika apskatīts ar neobligātām šablona daļām, sākotnēji tika izveidots kā tekstuāls SPARQL vaicājums. Izveidotais tekstuālais SPARQL vaicājums ir redzams 4.6. att. Izpildot vaicājumu tiek atlasīti studentu vārdi un numuri. Attēla labajā pusē dotais vizuālais

vaicājums ir ģenerēts no tekstuālā vaicājuma un tas ir vizualizēts pareizi, taču tas nav intuitīvs. To izpildot tiks iegūts sagaidāmais rezultāts, taču tas neatbilst vaicājumam, kādu autore sagaida šajā gadījumā. Vizuālajā vaicājumā pareizi ir attēlota klase *Student*, taču neintuitīvi kā nosacījuma klases ir attēloti atribūti *studentName* un *studentNumber*. Šiem atribūtiem vajadzētu būt norādītiem pie klases *Student*, nevis kā atsevišķām nosacījuma klasēm. Vizuālais vaicājums ir šādi izveidots dēļ tekstuālajā vaicājumā norādītajiem trijnieku šabloniem pie *OPTIONAL*.



4.6. att. **Vaicājums ar *OPTIONAL*, kas sākotnēji veidots kā tekstuāls vaicājums**

Izveidotajā tekstuālajā vaicājumā trijnieku šablonu pieraksts atšķiras no tā, kas parasti ir izmantots rīka *ViziQuer* ģenerētajos tekstuālajos vaicājumos. *ViziQuer* vaicājumos trijnieku šabloni pie *OPTIONAL* parasti tiek doti figūriekavās pa vienam. Šajā vaicājumā figūriekavās ir norādīti vairāki trijnieku šabloni, nevis tikai viens.

4.7. att. **Manuāli vizualizēts vaicājums ar *OPTIONAL***

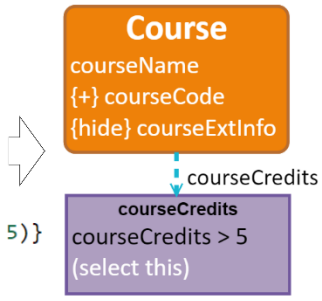
Tekstuālajam vaicājumam, kas dots 4.6. att., sagaidāmais vizuālais vaicājums ir vienkāršāks nekā uzģenerētais vizuālais vaicājums. Sagaidāmais vizuālais vaicājums, kurš tika vizualizēts manuāli, ir attēlots 4.7. att. Tajā ir dota viena klase *Student*, kurā ir norādīti izvēlētie neobligātie atribūti *studentName* un *studentNumber*. Šāds vaicājums ir intuitīvāks, jo neobligātie atribūti netiek attēloti kā nosacījuma klases.

Arī nākamais vaicājums, kas tika apskatīts, sākotnēji tika izveidots kā tekstuāls SPARQL vaicājums.

Izveidotais tekstuālais SPARQL vaicājums ir redzams 4.8. att. Izpildot vaicājumu tiek atlasīta tāda informācija par kursiem kā kursa nosaukums, kods un kredītpunktu skaits, kur kredītpunktu skaits tiek atlasīts tikai tad, ja to skaits ir lielāks par 5. Lai kurss būtu uzskatāms par vaicājuma risinājumu, tam ir jābūt dotai informācijai par kursa kodu, taču kursa nosaukums, kredītpunktu skaits un kursa papildus informācija var arī nebūt norādīta.

Uzģenerētajā vizuālajā vaicājumā, kas redzams 4.8. att. labajā pusē, tiek pareizi attēlotas vaicājuma šablona daļas, kurām risinājumam ir obligāti jāatbilst - ka risinājumam ir jābūt kursam (tiek attēlots ar oranžu taisnstūri ar klases nosaukumu *Course*) un, ka kursam ir jābūt norādītam kodam (tiek attēlots atribūtu sarakstā *courseCode* ar simbolu "+"). Pareizi tiek attēlotas arī vaicājuma neobligātās daļas – trijnieku šabloni ar kursa nosaukumu un kursa papildus informāciju. Kursa nosaukums un kursa papildus informācija atribūtu sarakstā pie klases tiek attēloti kā neobligāti atribūti bez "+" simbola, turklāt pie kursa papildus informācijas ir pareizi norādīts ar *"hide"* atslēgvārdu, ka šis atribūts netiek attēlots vaicājuma rezultātā jeb tiek paslēpts.

```
PREFIX : <http://lumii.lv/ontologies/UnivExample.owl#>
SELECT ?courseName ?courseCode ?courseCredits WHERE{
  ?Course a :Course.
  OPTIONAL{?Course :courseName ?courseName.}
  OPTIONAL{?Course :courseExtInfo ?courseExtInfo.}
  ?Course :courseCode ?courseCode.
  OPTIONAL{?Course :courseCredits ?courseCredits. FILTER (?courseCredits > 5)}
}
```



4.8. att. **Vaicājums ar *OPTIONAL*, kas sākotnēji veidots kā tekstuāls vaicājums**

Pareizi, taču neintuitīvi tiek vizualizēta viena no vaicājuma neobligātajām daļām - trijnieku šablons ar kredītpunktu skaitu. Šajā gadījumā atribūts *courseCredits* tiek attēlots kā nosacījuma klase. Kredītpunktu skaits tiek atlasīts tikai tad, ja to skaits ir lielāks par 5. Šāda atlase tiek panākta pie *OPTIONAL* norādot ierobežojumu izmantojot *FILTER*. Norādot nosacījumu vaicājuma neobligātajā daļā, netiks ietekmēts risinājumu skaits, bet gan atkarībā no tā, vai izpildās nosacījums, tiks ietekmēta kredītpunktu skaita atlase. Šāds vaicājums būtu intuitīvāks, ja *ViziQuer* vizuālā notācija atbalstītu iespēju izvēloties neobligātus atribūtus norādīt ierobežojumus ar *FILTER*. Šobrīd norādīt nosacījumu jeb *FILTER* ir iespējams tikai pie klases.

Manuāli izveidots vizuālā vaicājums piemērs ir dots 4.9. att., šis vaicājums atbilst 4.8. att. dotajam tekstuālajam vaicājumam, kurš automātiski tika vizualizēts neintuitīvs. Šādu vaicājumu izveidot izmantojot ViziQuer notāciju nav iespējams.

Attēlā 4.9. att. dotais vaicājums ir piemērs *ViziQuer* notācijas paplašināšanai - ir pievienots klāt nosacījums pie neobligātā atribūta *courseCredits*, kas nosaka, ka kursa kredītpunktu skaits tiks atlasīts tikai tad, ja to skaits ir lielāks par 5. Šādu iespēju pievienot nosacījumu vajadzētu arī pie neobligātām saitēm, kas vizuāli varētu izskatīties līdzīgi, kā nosacījums pie neobligātā atribūta dotajā attēlā.

4.9. att. **Vizuālās notācijas piemērs nosacījumu norādīšanai pie neobligātiem atribūtiem**

Šīs apakšnodaļas sākumā tika aprakstīts, ka izmantojot SPARQL vaicājumvalodu ir iespējams norādīt šablonu alternatīvas izmantojot atslēgvārdu *UNION*. Vispirms tika apskatīts vaicājums ar *UNION*, kurš tika veidots kā vizuāls vaicājums rīkā *ViziQuer*. Izveidot vizuālu vaicājumu ar *UNION* var mezglā kā klases nosaukumu norādot "[+]", līdz ar to šis mezgls neapzīmē kādu datu instanci, bet gan ievieš apakškoku disjunkciju [37].

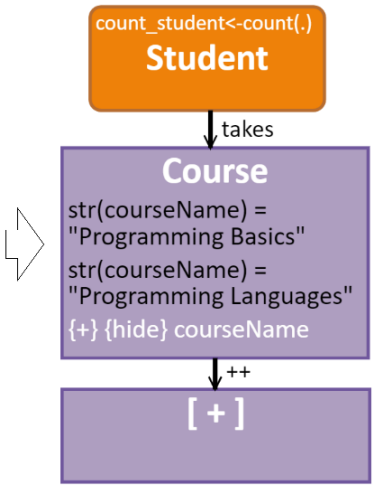
4.10. att. **Vaicājums ar *UNION*, kas sākotnēji veidots kā vizuāls vaicājums**

Manuāli izveidotais vizuālais vaicājums ir dots kreisajā pusē 4.10. att. Izpildot vaicājumu tiek atlasīti un apvienoti neobligāto un obligāto kursu nosaukumi, veidojot vienu sarakstu ar kursu nosaukumiem. Šādi vaicājumi ir īpaši noderīgi, ja ir nepieciešams apvienot datus no vairākiem avotiem. Kā redzams, vizuālajā vaicājumā ir dots apvienojuma mezgls ([+]), no kura iziet divas saites uz divām klasēm – vienā no tām tiek atlasīti neobligāto kursu nosaukumi, taču otrā obligāto kursu nosaukumi. Kā saites nosaukums ir norādīts "++", šādi tiek apzīmēta vaicājuma šķautne, kas neatbilst saitei starp datu instancēm [37]. Šajā gadījumā abās klasēs atribūtam *courseName* nācās

arī izmantot aizstājvārdu (*alias*), citādāk netiek uzģenerēts vēlamais SPARQL vaicājums un kursu nosaukumi netiek apvienoti pareizi. Labajā pusē 4.10. att. ir dots ģenerētais vizuālais vaicājums, un, kā redzams, tas nav vizualizēts pareizi. Tiek attēlota tikai viena no abām klasēm, kā arī tajā netiek attēloti izvēlētie atribūti. Pareizi uzģenerēts vizuālais vaicājums būtu tāds pats, kāds ir dots sākotnējais manuāli veidotais vizuālais vaicājums.

Tālāk tika apskatīta rīka spēja ģenerēt vizuālu vaicājumu ar *UNION*, no vaicājuma, kas sākotnēji ir veidots kā tekstuāls SPARQL vaicājums. Izveidotais tekstuālais SPARQL vaicājums ir dots 4.11. att., tajā var redzēt, ka atslēgvārds *UNION* it dots starp divām šablonu apakškopām, kur katra apakškopa ir ietverta figūriekavās. Risinājumam ir jāatbilst vismaz vienai no šīm šablonu apakškopām.

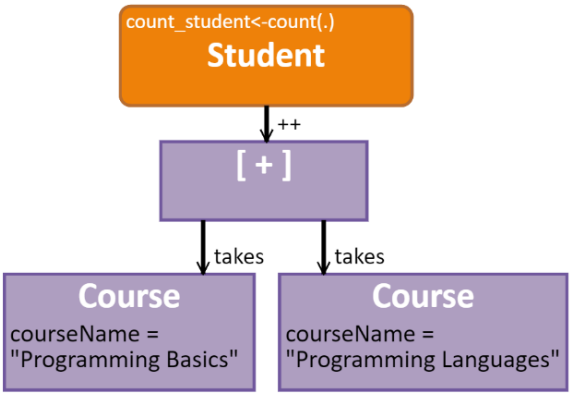
```
PREFIX : <http://lumii.lv/ontologies/UnivExample.owl#>
SELECT (COUNT(?Student) AS ?count_student) WHERE{
  ?Student a :Student.
  ?Course a :Course.
  ?Student :takes ?Course.
  {
    ?Course :courseName ?courseName.
    FILTER(STR(?courseName) = "Programming Basics")
  }
  UNION
  {
    ?Course :courseName ?courseName.
    FILTER(STR(?courseName) = "Programming Languages")
  }
}
```



4.11. att. **Vaicājums ar *UNION*, kas sākotnēji veidots kā tekstuāls vaicājums**

Izpildot tekstuālo vaicājumu, kurš dots 4.11. att., tiek iegūts studentu skaits, kuri studē kursu ar nosaukumu "*Programming Basics*" vai "*Programming Languages*". Mēģinājums uzģenerēt korektu vizuālu vaicājumu no dotā tekstuālā vaicājuma neizdevās. Kā redzams, attēla labajā pusē dotais vizuālais vaicājums nav atbilstošs tekstuālajam SPARQL vaicājumam. Uzģenerētajā vaicājumā pie *Course* klases norādītie nosacījumi nosaka, ka kursa nosaukumam jābūt "*Programming Basics*" un "*Programming Langugaes*", taču tekstuālajā vaicājumā ir dots, kursa nosaukumam jābūt "*Programming Basics*" vai "*Programming Langugaes*". Kā arī apvienošanas mezgls ir attēlots nepareizi, tam vajadzētu būt starp klasēm *Student* un *Course*.

Tā kā *ViziQuer* notācija atbalsta *UNION* klauzulu, atbilstošu vizuālo vaicājumu ir iespējams izveidot manuāli un tas ir dots 4.12. att. Vizuālajā vaicājumā kā galvenais mezgls ir dota klase *Student*, no kuras iziet saite uz apvienošanas mezglu. No apvienošanas mezgla iziet divas saites – viena uz klasi *Course*, kurā ir nosacījums, ka kursa nosaukumam jābūt "*Programming Basics*" un otra arī uz klasi *Course*, taču šai klasei ir dots nosacījums, ka kursa nosaukumam ir jābūt "*Programming Languages*".



4.12. att. **Manuāli vizualizēts vaicājums ar *UNION***

Šajā apakšnodaļā tika apskatīti pieci darba ietvaros izveidotie disjunktīvie vaicājumi, kā arī rīka *ViziQuer* spēja tos vizualizēt. No apskatītajiem 3 vaicājumiem ar *OPTIONAL*, pareizi tika vizualizēti divi vaicājumi, taču neintuitīvi. Tiesa gan tos bija iespējams izpildīt un iegūt pareizu rezultātu, taču tie nebija atbilstoši autores manuāli veidotajiem sagaidāmajiem vizuālajiem vaicājumiem. Šajos vaicājumos kā nosacījuma klases tika attēloti klašu atribūti. No apskatītajiem 2 vaicājumiem ar *UNION*, abi tika vizualizēti neveiksmīgi, taču autore darbā ir sniegusi manuāli izveidotus vaicājumus, kādiem tiem vajadzētu būt veiksmīgas vizualizācijas gadījumā.

4.4 Vaicājumi ar apkopošanas funkcijām

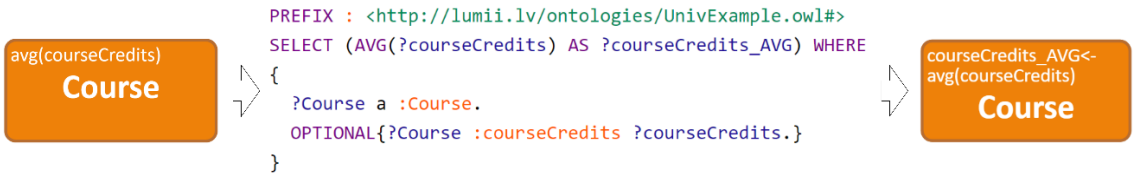
Apkopošanas funkcijas veic aprēķinus, izmantojot vienu vai vairākas vērtības, un rezultātā atgriež vienu vērtību. SPARQL 1.1 vaicājumvalodas specifikācijā definētās apkopošanas funkcijas ir saskaitīšana (*COUNT*), summēšana (*SUM*), mazākās vērtības atrašana (*MIN*), lielākās vērtības atrašana (*MAX*), vidējās vērtības aprēķināšana (*AVG*), grupas savirknēšana (*GROUP_CONCAT*) un parauga iegūšana (*SAMPLE*) [11]. Šajā apakšnodaļā tiek apskatītas arī *GROUP BY* un *HAVING* klauzulas, jo tās bieži vien tiek izmantotas kopā ar apkopošanas funkcijām. Rīks *ViziQuer* nodrošina atbalstu visām šīm apkopošanas funkcijām. Rīkā *ViziQuer* veidojot vizuālu vaicājumu

ir iespējams izvēlēties arī tādu apkopošanas funkciju kā *COUNT_DISTINCT*, kas nav iekļauta SPARQL 1.1 vaicājumvalodas specifikācija, taču rīkā *ViziQuer* tā ir realizēta izmantojot apkopošanas funkciju *COUNT* un modifikatoru *DISTINCT*.

Šajā apakšnodaļā tiek apskatīti šādi četri izveidotie vaicājumi ar apkopošanas funkcijām:

- vaicājums ar apkopošanas funkciju pār vienu risinājumu grupu;
- divi vaicājumi ar grupēšanu un apkopošanas funkciju pār vairākām risinājumu grupām;
- vaicājums ar grupēšanu, apkopošanas funkcijām un grupu filtrēšanu.

Vaicājuma piemēru apkopošanas funkciju izmantošanai vizuālos vaicājumos var redzēt 4.13. att. Vaicājumā tiek iegūts vidējais kredītpunktu skaits no visiem kursiem. Rīkā *ViziQuer* veidotajā vizuālajā vaicājumā apkopošanas funkcijas tiek attēlotas klases kreisajā augšējā stūrī ar baltu fonta krāsu.



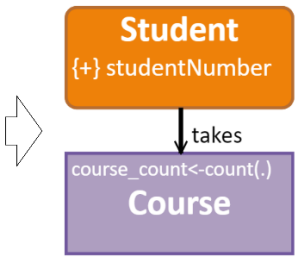
4.13. att. **Vaicājums ar apkopošanas funkciju pār vienu grupu, kas sākotnēji veidots kā vizuāls vaicājums**

Labajā pusē 4.13. att. ir dots vizuāls vaicājums, kas ģenerēts no tekstuāla SPARQL vaicājuma. Šis vizuālais vaicājums ir iegūts izejot pilnu ceļu (vizuāls vaicājums - tekstuāls SPARQL vaicājums - vizuāls vaicājums), taču starp sākotnējo vaicājumu un beigu vaicājumu ir neliela atšķirība – sākotnējā vizuālajā vaicājumā apkopošanas funkcijas rezultātam nav norādīts aizstājvārds (*alias*), taču uzģenerētajā vizuālajā vaicājumā tas ir norādīts. Pēc SPARQL 1.1 vaicājumvalodas specifikācijas [11], apkopošanas funkcijām ir jāizmanto aizstājvārdi. Veidojot vizuālu vaicājumu rīkā *ViziQuer*, apkopošanas funkcijām šis aizstājvārds nav obligāti jānorāda, taču tas tiek automātiski uzģenerēts un pievienots tekstuālam SPARQL vaicājumam. Attēlā ir dots piemērs tikai ar vidējās vērtības (*AVG)* apkopošanas funkciju, taču tādā pašā veidā tiek apstrādātas arī citas apkopošanas funkcijas.

Apkopošanas funkcijas izmanto kādai risinājumu grupai. Pēc noklusējuma risinājuma kopa sastāv no vienas grupas, kas satur visus risinājumus. Grupēšanu var veikt izmantojot *GROUP BY* [11]. Lai aprēķinātu risinājuma agregātvērtības, risinājumu kopa vispirms tiek sadalīta vienā vai vairākās grupās, un katrai grupai aprēķina agregātvērtību.

Tālāk tika apskatīta rīka spēja vizualizēt vaicājumu ar grupēšanu un apkopošanas funkciju pār vairākām risinājumu grupām, kas sākotnēji tika izveidots kā tekstuāls SPARQL vaicājums. Tekstuāls SPARQL vaicājums ar grupēšanu ir dots 4.14. att. Izpildot vaicājumu tiek iegūts kursu skaits katram studentam. Rezultāts tiek iegūts saskaitot kursus un sagrupējot tos pēc studenta numura.

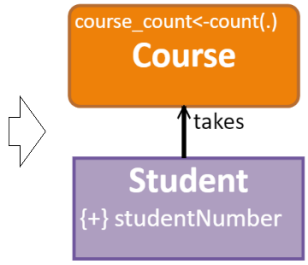
```
PREFIX : <http://lumii.lv/ontologies/UnivExample.owl#>
SELECT ?studentNumber (COUNT(?Course) AS ?course_count) WHERE{
  ?Student a :Student.
  ?Course a :Course.
  ?Student :studentNumber ?studentNumber.
  ?Student :takes ?Course.
}
GROUP BY ?studentNumber
```



4.14. att. **Vaicājums ar grupēšanu, kas sākotnēji veidots kā tekstuāls vaicājums**

Kaut arī sākotnēji var šķist, ka attēla labajā pusē dotais vizuālais vaicājums ir uzģenerēts pareizi, taču tas nav pareizs, jo vizuālajā vaicājumā ir nepareizi attēlots galvenais vaicājuma mezgls – *Student* un *Course* klasei vajadzētu būt mainītās vietās. Apkopošanas funkcijas var būt norādītas tikai pie vaicājuma galvenā mezgla, tāpēc klasei *Course* ir jābūt kā galvenajam mezglam.

```
PREFIX : <http://lumii.lv/ontologies/UnivExample.owl#>
SELECT ?studentNumber (COUNT(?Course) AS ?course_count) WHERE{
  ?Course a :Course.
  ?Student a :Student.
  ?Student :studentNumber ?studentNumber.
  ?Student :takes ?Course.
}
GROUP BY ?studentNumber
```



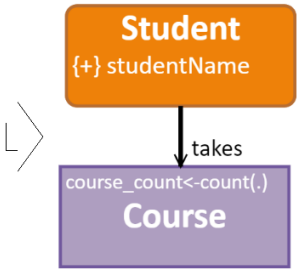
4.15. att. **Vaicājums ar grupēšanu, kas sākotnēji veidots kā tekstuāls vaicājums**

Kreisajā pusē 4.15. att. ir dots tekstuāls SPARQL vaicājums, kas ir ekvivalents augstāk 4.14. att. dotajam vaicājumam. Starp abiem vaicājumiem ir neliela atšķirība – trijnieku šablonu secība. Taču šī secība būtiski ietekmē rīka spēju uzģenerēt pareizu, atbilstošu vizuālo vaicājumu. Šajā gadījumā uzģenerētajā vizuālajā vaicājumā ir pareizi attēlots galvenais vaicājuma mezgls – tā ir klase *Course*, kā arī apkopošanas funkcija *COUNT* ir pareizi dota pie galvenā mezgla. Var secināt, ka SPARQL vaicājumu, kas satur apkopošanas funkcijas, veiksmīga vizualizēšana ir atkarīga no vaicājumā dotās trijnieku šablonu kārtības.

Apakšnodaļā 4.2 tika apskatīta individuālu risinājumu filtrēšana, taču SPARQL vaicājumvaloda nodrošina iespēju filtrēt arī risinājumu grupas. Ja ir nepieciešams filtrēt risinājuma

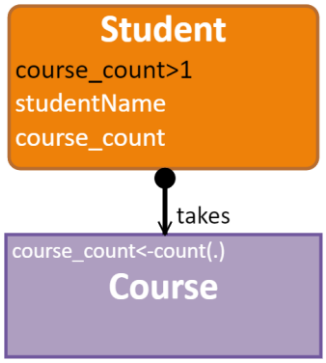
grupas līdzīgi kā individuālus risinājums, var izmantot *HAVING* klauzulu [11]. Vaicājums ar *GROUP BY* un *HAVING* ir dots 4.16. att., tajā tiek atlasīti studentu vārdi kopā ar studentu kursu skaitu, kur studenta kursu skaits ir lielāks par 1. Izmantojot *GROUP BY* klauzulu, risinājumi tiek sagrupēti pēc studenta vārda, izmantojot *HAVING* klauzulu, šīs grupas tiek filtrētas un risinājumu kopā tiek iekļautas tikai tās grupas, kur kursu skaits ir lielāks par 1.

```
PREFIX : <http://lumii.lv/ontologies/UnivExample.owl#>
SELECT ?studentName (COUNT(?Course) AS ?course_count) WHERE{
  ?Student a :Student.
  ?Course a :Course.
  ?Student :studentName ?studentName.
  ?Student :takes ?Course.
}
GROUP BY ?studentName
HAVING (COUNT(?Course)>1)
```



4.16. att. **Vaicājums ar *HAVING*, kas sākotnēji veidots kā tekstuāls vaicājums**

Kā redzams 4.16. att. labajā pusē, vizuālais vaicājums nav uzģenerēts korekti – ir novērojama tā pati kļūda, kas bija apskatīta iepriekšējos vaicājumos ar apkopošanas funkcijām, ka apkopošanas funkcijas var būt norādītas tikai vaicājuma galvenajā mezglā. Vizuālajā vaicājumā nav arī attēlota grupu filtrēšana izmantojot *HAVING* klauzulu. Veidojot vizuālu vaicājumu rīkā *ViziQuer*, ir iespējams pievienot grupēšanu pēc kāda atribūta, taču nav iespējams pievienot grupu filtrēšanu izmantojot *HAVING*. Var secināt, ka *ViziQuer* vizuālā notācija neatbalsta *HAVING* klauzulu.



4.17. att. **Manuāli vizualizēts vaicājums ar grupu filtrēšanu**

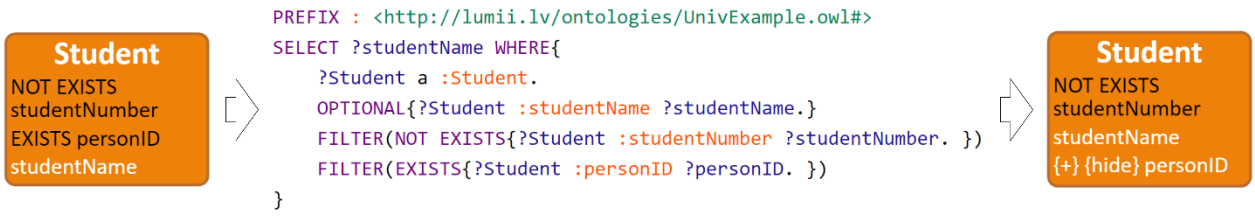
Kaut arī izmantojot *ViziQuer* vizuālo notāciju nav iespējams izveidot vaicājumus ar *HAVING*, lai iegūtu tādu pašu rezultātu, vaicājumu ir iespējams pārrakstīt citā formā – kā vaicājumu ar apakšvaicājumu. Manuāli izveidots vaicājums ar apakšvaicājumu ir dots 4.17. att. un tas atgriezīs tādu pašu rezultātu kā 4.16. att. dotais tekstuālais vaicājums ar *HAVING* klauzulu.

Šajā apakšnodaļā tika apskatīti četri darba ietvaros izveidotie vaicājumi ar apkopošanas funkcijām, kā arī rīka *ViziQuer* spēja tos vizualizēt. No apskatītajiem vaicājumiem divi tika vizualizēti neveiksmīgi un divi veiksmīgi. Tika secināts, ka no trijnieku šablonu kartības ir atkarīgs, vai vaicājums tiks vizualizēts veiksmīgi un būs pareizi norādīts galvenais mezgls ar apkopošanas funkciju. *ViziQuer* notācija neatbalsta *HAVING* klauzulu, tāpēc autore neveiksmīgi vizualizētajam vaicājumam piedāvā citu vaicājuma formu, lai iegūtu to pašu rezultātu - vaicājumu ar apakšvaicājumu.

4.5 Vaicājumi ar noliegumu

Pēc SPARQL 1.1 vaicājumvalodas specifikācijas [11], vaicājumvalodā ir iekļauti divi nolieguma veidi - viens no tiem ir balstīts uz rezultātu filtrēšanu atkarībā no tā, vai grafa šablons atbilst vai neatbilst datu kopai, savukārt otrs veids ir balstīts uz risinājumu izslēgšanu, kas atbilst citam grafu šablonam. Vaicājuma risinājumu filtrēšana tiek veikta *FILTER* izteiksmē, izmantojot *NOT EXISTS* un *EXISTS*, taču izslēgt noteiktus vaicājuma risinājumus var izmantojot otru nolieguma veidu – *MINUS*. Šajā apakšnodaļā tiek apskatīti šādi četri izveidotie vaicājumi ar noliegumu funkcijām:

- vaicājums ar filtrēšanu un *NOT EXISTS* izteiksmi, kur dati tiek atlasīti no vienas klases;
- vaicājums ar filtrēšanu un *NOT EXISTS* izteiksmi, kur dati tiek atlasīti no divām klasēm, starp kurām saite ir ar noliegumu;
- vaicājums ar noteiktu risinājumu izslēgšanu (*MINUS*).

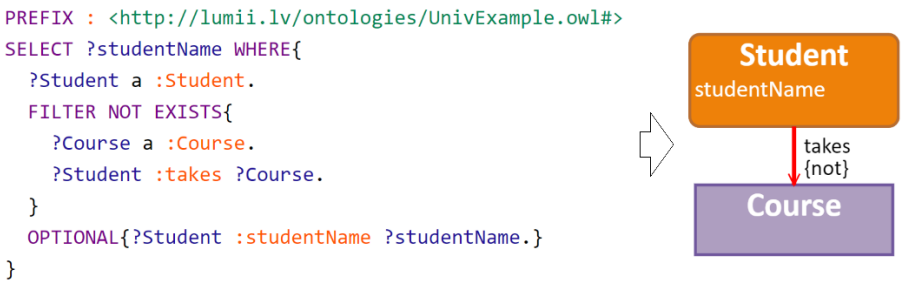


4.18. att. **Vaicājums ar *NOT EXISTS* un *EXISTS*, kas sākotnēji veidots kā vizuāls vaicājums**

Vaicājuma piemērs ar *FILTER, NOT EXISTS* un *EXISTS* ir dots 4.18. att. Šis vaicājums atlasa visus studentus, kuriem nav piešķirts studenta numurs, taču ir dots unikāls personas identifikators. Rīkā *ViziQuer FILTER* izteiksmi ir iespējams pievienot sadaļā "*Conditions*" norādot nosacījumu. Vizuālajā vaicājumā izveidotie nosacījumi tiek attēloti zem klases nosaukuma un atšķirībā no atribūtiem, nosacījumi tiek attēloti ar melnu fonta krāsu. Attēlā var redzēt, ka, izejot

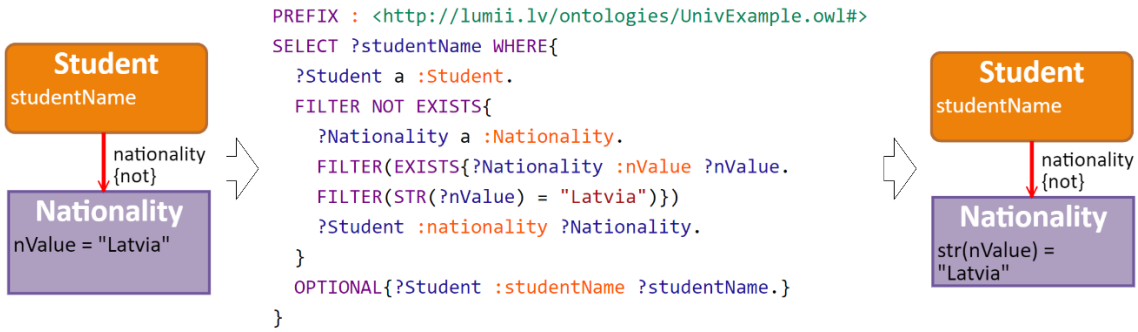
pilnu ceļu no sākotnējā vizuālā vaicājuma, beigās netiek iegūts tāds pats vizuālais vaicājums kā sākumā. Kaut arī vizuālie vaicājumi izskatās atšķirīgi, tie ir ekvivalenti un abi vaicājumi atgriezīs vienādu rezultātu.

Nākamais vaicājums, kas tika apskatīts, sākotnēji netika vizualizēts rīkā *ViziQuer*, bet gan izveidots kā tekstuāls SPARQL vaicājums. Izveidotais tekstuālais SPARQL vaicājums ir dots 4.19. att. Izpildot vaicājumu tiek iegūta informācija par to, kuriem studentiem nav neviena kursa, un šajā vaicājumā atšķirībā no iepriekšējā vaicājuma tiek atlasīti dati no divām klasēm – *Student* un *Course*.



4.19. att. **Vaicājums ar nolieguma saiti, kas sākotnēji veidots kā tekstuāls vaicājums**

Labajā pusē 4.19. att. dotais vizuālais vaicājumus ir ģenerēts no kreisajā pusē dotā tekstuālā SPARQL vaicājuma un kā redzams, tas ir izveidots pareizi – vizuālajā vaicājumā ir attēlotas divas klases – *Student* un *Course*, kā arī pareizi ir norādīta galvenā klase. Starp abām šīm klasēm ir attēlota attiecība, taču šajā gadījumā tā ir saite ar noliegumu. Saite ar noliegumu tiek attēlota ar sarkanu bultu starp klasēm un pie saites nosaukuma ir dots apzīmējums *{not}*.



4.20. att. **Vaicājums ar nolieguma saiti, kas sākotnēji veidots kā vizuāls vaicājums**

Nākamais vaicājums, kas tika apskatīts ar nolieguma saiti, tika veidots kā vizuāls vaicājums rīkā *ViziQuer*. Izveidotais vizuālais vaicājums ir dots 4.20. att., to izpildot tiek atlasīti studentu vārdi tiem studentiem, kuri ir ārzemju studenti jeb, kuriem studentiem nacionalitāte nav latviešu. Vaicājumā tiek atlasīti dati no divām klasēm – *Student* un *Nationality*.

Attēla 4.20. att. labajā pusē dotais vizuālais vaicājumus ir ģenerēts no tekstuāla SPARQL vaicājuma, kas ir ģenerēts no rīkā *ViziQuer* veidota vizuālā vaicājuma. Kā redzams, uzģenerētais vizuālais vaicājums ir pareizs un tas ir identisks sākotnējam vizuālajam vaicājumam, neskatoties uz nebūtiskām atšķirībām. Vizuālajā vaicājumā ir attēlotas divas klases – *Student* un *Nationality* – un pareizi ir norādīta galvenā klase *Student*. Pie *Nationality* klases ir attēlots nosacījums, ka nacionalitātei ir jābūt latviešu, taču starp *Nationality* un *Student* klasi ir attēlota nolieguma saite.

Otrs SPARQL vaicājumvalodas nolieguma veids ir *MINUS*, kuru izmantojot ir iespējams izslēgt noteiktus vaicājuma risinājumus no citas vaicājuma risinājumu kopas. Rīks *ViziQuer* nepiedāvā vizuālo notāciju vaicājumu veidošanai ar *MINUS*, tāpēc netiks apskatīts gadījums, kad vispirms ir veidots vizuālais vaicājums.

```
PREFIX : <http://lumii.lv/ontologies/UnivExample.owl#>
SELECT ?studentName WHERE{
  ?Student a :Student.
  ?Student :studentName ?studentName.
  MINUS {
    ?Student :studentName ?studentName
    FILTER(STR(?studentName) = "Eve")
  }
}
```

4.21. att. **Vaicājums ar *MINUS*, kas sākotnēji veidots kā tekstuāls vaicājums**

Izveidotais tekstuālais SPARQL vaicājums ar *MINUS* ir dots 4.21. att. Vaicājumā tiek atlasīti visi studentu vārdi, izņemot vārdu "*Eve*". Ģenerējot no tekstuālā SPARQL vaicājuma vizuālu vaicājumu, tas netika uzģenerēts korekti. Uzģenerētajā vizuālajā vaicājumā zem klases nosaukuma *Student* ir dots nosacījums, kas nozīmē, ka studenta vārdam ir jābūt vienādam ar simbolu virkni "*Eve*", līdz ar to uzģenerētais vizuālais vaicājums atgriež pretēju rezultātu sākotnējam tekstuālajam SPARQL vaicājumam – tiek atgriezts tikai studenta vārds "*Eve*", nevis kā sagaidāms, ka tieši šis vārds tiks izslēgts no risinājumu kopas. Rīkā *ViziQuer* vaicājumus, kas satur *MINUS*, nav iespējams izveidot un arī nav iespējams uzģenerēt no tekstuāliem vaicājumiem vizuālus vaicājumus.

Šajā apakšnodaļā tika apskatīti četri darba ietvaros izveidotie vaicājumi ar noliegumu, kā arī rīka *ViziQuer* spēja tos vizualizēt. No apskatītajiem vaicājumiem tikai viens tika vizualizēts neveiksmīgi – vaicājums ar *MINUS*, jo *ViziQuer* notācija neatbalsta šo nolieguma veidu. Taču pēc rīka autoru dotās informācijas, šo konstrukciju ir iespējams aizstāt ar globāliem apakšvaicājumiem [37].

4.6 Vaicājumi ar risinājumu virknes modifikatoriem

No vaicājumu šabloniem sākotnēji tiek iegūta nesakārtota kolekcija ar risinājumiem. Lai izveidotu citu risinājumu virkni, kas ir sakārtota noteiktā secībā vai kurā ir iekļauts ierobežots skaits risinājumu, var izmantot risinājumu virknes modifikatorus. Modifikatori kā ieejas datus saņem risinājumu virkni, pārveido to un atgriež jaunu risinājuma virkni. SPARQL 1.1 vaicājumvalodas specifikācijā [11] ir uzskaitīti šādi vaicājuma risinājumu virknes modifikatori:

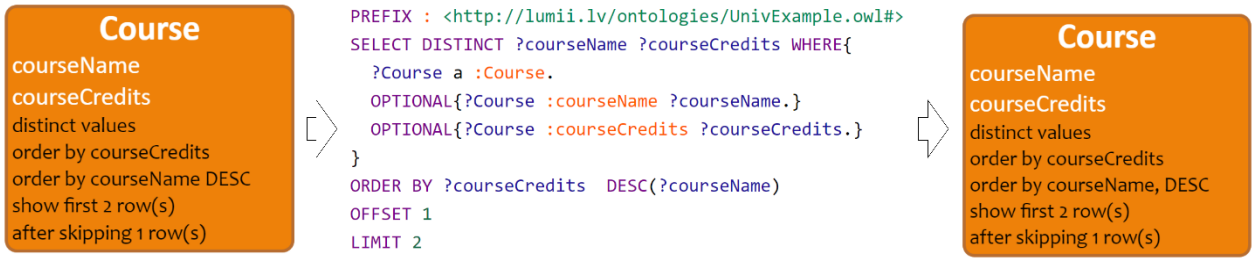
- kārtības modifikators *ORDER BY* – sakārto risinājumus noteiktā secībā;
- projicēšanas modifikators – atlasa noteiktus mainīgos, *SELECT* klauzula identificē mainīgos, kuriem jāparādās vaicājuma rezultātā;
- atšķirīgu vērtību modifikators *DISTINCT* – atlasa tikai unikālus risinājumus no visas risinājumu kopas;
- reducēšanas modifikators *REDUCED* – atļauj dažu vērtību izslēgšanu, kas nav atšķirīgas;
- nobīdes modifikators *OFFSET* - ierobežo, no kurienes sākas risinājumi dotajā risinājumu virknē;
- limita modifikators *LIMIT* – nosaka maksimālo risinājumu skaitu, kas tiek atgriezts no visiem vaicājuma risinājumiem.

Modifikatori tiek piemēroti tādā secībā, kāda tie ir norādīti augstāk dotajā sarakstā. Tālāk šajā sadaļā tiks apskatīti visi sarakstā dotie modifikatori, izņemot vienu. Netiks apskatīts projicēšanas modifikators, jo tas ir iekļauts ikvienā šajā darbā apskatītajā SPARQL vaicājumā, *SELECT* klauzulā identificējot mainīgos, kuriem jāparādās vaicājuma rezultātā. Visus risinājumu virknes modifikatorus ir iespējams izmantot vienlaicīgi vienā vaicājumā izņemot *DISTINCT* un *REDUCED*, tāpēc šie divi modifikatori tika apskatīti atsevišķos vaicājumos. Šajā apakšnodaļā tiek apskatīti šādi trīs izveidotie vaicājumi ar risinājumu virknes modifikatoriem:

- vaicājums ar *DISTINCT* un citiem risinājumu virknes modifikatoriem, kas sākotnēji veidots kā vizuāls vaicājums;
- vaicājums ar *DISTINCT*, citiem risinājumu virknes modifikatoriem un vairākām nosacījumu klasēm, kas sākotnēji veidots kā tekstuāls vaicājums;
- vaicājums ar *REDUCED* un citiem risinājumu virknes modifikatoriem.

Pirmais apskatītais vaicājums ar risinājumu virknes modifikatoriem tika veidots kā vizuāls vaicājums rīkā *ViziQuer*. Vaicājums ar *DISTINCT, ORDER BY, OFFSET* un *LIMIT* ir dots 4.22. att. Vaicājumā tiek atlasīta tāda informācija par kursu kā kursa nosaukums un kredītpunktu skaits,

rezultātu kopā ir iekļauti tikai atšķirīgi kursi jeb risinājumi, tie tiek kārtoti augošā secībā pēc kredītpunktu skaita un gadījumā, ja kredītpunktu skaits sakrīt, tad tālāk tiek kārtoti pēc kursa nosaukuma dilstošā secībā. No vaicājuma risinājumu kopas tiek atlasīta noteikta apakškopa – tiek izlaists pirmais risinājums un ir ierobežots risinājumu skaits – 2.

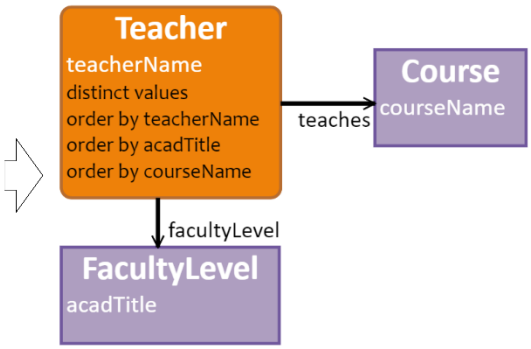


4.22. att. **Vaicājums ar risinājumu virknes modifikatoriem, kas sākotnēji veidots kā vizuāls vaicājums**

Dotie modifikatori 4.22. att. vizuālajā vaicājumā tiek attēloti zem izvēlētajiem klases atribūtiem ar melnu fonta krāsu. Ģenerējot no tekstuālā SPARQL vaicājuma vizuālu vaicājumu, tiek iegūts korekts vizuālais vaicājums un tas ir tāds pats kā sākotnējais vaicājums.

Apskatītais vaicājums satur tikai vienu klasi bez saitēm, tāpēc tika apskatīts arī vaicājums ar vairākām klasēm un saitēm starp tām. Tā kā iepriekšējais vaicājums sākotnēji tika veidots kā vizuāls vaicājums rīkā *ViziQuer*, tad vaicājums ar vairākām klasēm un risinājumu virknes modifikatoriem tika veidots kā tekstuāls SPARQL vaicājums.

```
PREFIX : <http://lumii.lv/ontologies/UnivExample.owl#>
SELECT DISTINCT ?teacherName ?acadTitle ?courseName WHERE{
  ?Teacher a :Teacher.
  ?Course a :Course.
  ?FacultyLevel a :FacultyLevel.
  OPTIONAL{?Teacher :teacherName ?teacherName.}
  OPTIONAL{?Course :courseName ?courseName.}
  OPTIONAL{?FacultyLevel :acadTitle ?acadTitle.}
  ?Teacher :teaches ?Course.
  ?Teacher :facultyLevel ?FacultyLevel.
}
ORDER BY ?teacherName ?acadTitle ?courseName
```



4.23. att. **Vaicājums ar risinājumu virknes modifikatoriem, kas sākotnēji veidots kā tekstuāls vaicājums**

Izveidotais tekstuālais SPARQL vaicājums ar risinājumu virknes modifikatoriem un vairākām datu klasēm ir redzams 4.23. att. Vaicājumā ir izmantoti divi risinājumu virknes modifikatori – *DISTINCT* un *ORDER BY*. Izpildot vaicājumu tiek atlasīts pasniedzēja vārds kopā ar pasniedzēja akadēmisko nosaukumu un kursu nosaukumu, ko pasniedzējs pasniedz. Vaicājuma

rezultāti tiek kārtoti augošā secībā pēc pasniedzēja vārda, akadēmiskā nosaukuma un kursa nosaukuma. Vizuālais vaicājums, kas ir ģenerēts no tekstuālā SPARQL vaicājuma, ir redzams 4.23. att. labajā pusē un tas ir pareizs. Vizuālajā vaicājumā ir attēlotas trīs klases – *Teacher*, *Course* un *FacultyLevel*, pie katras klases ir pareizi norādīti izvēlētie atribūti. Risinājumu virknes modifikatori ir uzskaitīti zem galvenās klases *Teacher* izvēlētajiem atribūtiem.

Vaicājumā izmantojot *DISTINCT* modifikatoru tiek atlasīti tikai unikāli risinājumi, izslēdzot visus dublikātus, savukārt *REDUCED* atšķirībā no *DISTINC* modifikatora neizslēdz dublikātus, taču atļauj tos izslēgt [11]. Izmantojot *REDUCED*, no vaicājuma risinājuma kopas var tikt izslēgti daži, visi vai neviens dublikāts. Izmantojot *ViziQuer* vizuālo notāciju šobrīd nav iespējams izveidot vizuālus vaicājumus ar *REDUCED*, tāpēc tika apskatīts tekstuāls SPARQL vaicājums.

```
PREFIX : <http://lumii.lv/ontologies/UnivExample.owl#>
SELECT REDUCED ?nValue WHERE{
  ?Nationality a :Nationality.
  OPTIONAL{?Nationality :nValue ?nValue.}
}
ORDER BY ?nValue
```

4.24. att. **Vaicājums ar *REDUCED*, kas sākotnēji veidots kā tekstuāls vaicājums**

Izveidotais tekstuālais SPARQL vaicājums ir dots 4.24. att. Vaicājumā tiek atlasītas visas nacionalitātes, no kurām dublikāti var tikt izslēgti. Vizuālais vaicājums ir daļēji pareizs – tajā ir pareizi attēlota klase, izvēlētie klases atribūti un arī secības modifikators *ORDER BY*, taču trūkst modifikators *REDUCED*. *ViziQuer* vizuālā notācija neatbalsta *REDUCED* modifikatoru. Vizuāli tas varētu tikt attēlots tāpat kā modifikators *DISTINCT*, taču pēc rīka autoru dotās informācijas, pastāv neskaidrības saistībā ar citām iespējamām implementācijām, piemēram, vaicājumu tulkošanu uz SQL [37].

Šajā apakšnodaļā tika apskatīti trīs darba ietvaros izveidotie vaicājumi ar risinājumu virknes modifikatoriem, kā arī rīka *ViziQuer* spēja tos vizualizēt. No apskatītajiem vaicājumiem viens tika vizualizēts neveiksmīgi – vaicājums ar *REDUCED*, jo *ViziQuer* notācija neatbalsta šo modifikatoru. Šajā gadījumā nav alternatīvas vaicājumu formas, ar kurām varētu iegūt tādu pašu rezultātu, kā izmantojot modifikatoru *REDUCED*.

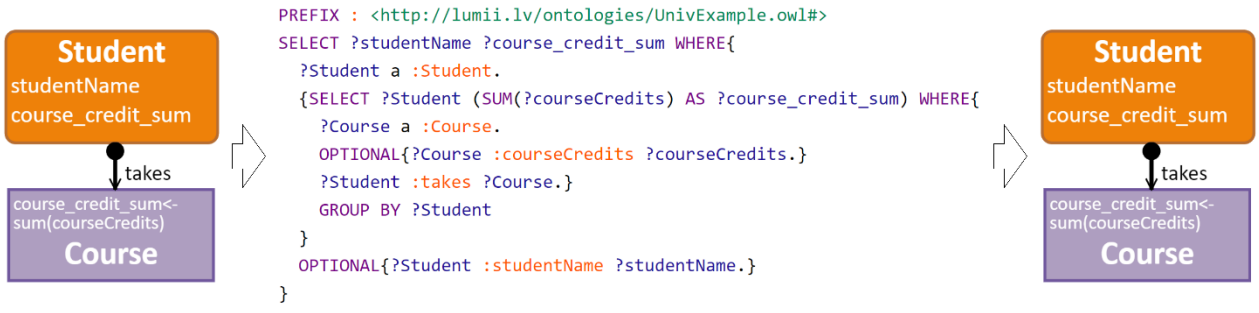
4.7 Vaicājumi ar apakšvaicājumiem

Aapakšvaicājumi ir veids, kā var izpildīt SPARQL vaicājumus citos vaicājumos, lai iegūtu rezultātu, kuru iespējams citādi nevar sasniegt. SPARQL vaicājumi tiek izpildīti no apakšas uz augšu, tas nozīmē, ka apakšvaicājumi tiek izpildīti pirmie un iegūtie rezultāti tiek nodoti ārējam vaicājumam [11].

Rīkā *ViziQuer* izveidot vizuālus vaicājumus ar apakšvaicājumiem var tāpat kā vaicājumus ar saitēm starp vairākām klasēm. Vizuālajā vaicājumā apakšvaicājumu no citām saitēm var atšķirt pēc bultas attēlojuma – tā sākas no melna punkta. Jāņem vērā arī, ja vēlas vaicājumā atlasīt apakšvaicājumā izvēlētos atribūtus, tad šie atribūti ir jāatlasa arī pie ārējā vaicājuma. Veidojot vizuālu vaicājumu ar apakšvaicājumiem, galvenais mezgls var būt gan kāda klase, gan arī kā mezgls bez klases, kuru var izmantot kā ārēju slāni apakšvaicājumu rezultātu apkopošanai, apvienošanai, projicēšanai, filtrēšanai, kā arī dažādu agregātfunkciju piemērošanai. Šādu mezglu klases nosaukumā vietā apzīmē ar kvadrātiekavām ([]) [37].

Tā kā galvenais mezgls var reprezentēt gan klasi, gan papildus slāni apakšavaicājumiem, un starp vaicājumiem un apakšvaicājumiem var būt dažādas saites, tālāk šajā sadaļā tiks aplūkota rīka *ViziQuer* spēja vizualizēt vaicājumus ar dažādiem galvenajiem mezgliem un dažādām saitēm.

Vaicājums ar apakšvaicājumu, kurā galvenais mezgls ir kāda klase, ir dots 4.25. att. Attēla kreisajā pusē ir dots sākotnējais vizuālais vaicājums, vidū ir tekstuālais vaicājums un kreisajā pusē ir dots rezultāts – no tekstuālā vaicājuma uzģenerētais vizuālais vaicājums.



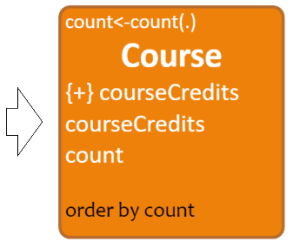
4.25. att. **Vaicājums ar apakšvaicājumu, kas sākotnēji veidots kā vizuāls vaicājums**

Var redzēt, ka vizuālajā vaicājumā 4.25. att. ir dotas divas klases *Student* un *Course*, un starp tām ir novilkta saite, kas apzīmē, ka galvenais mezgls (klase *Student*) ir ārējais vaicājums, taču klase *Course* ir apakšvaicājums. Izpildot vaicājumu tiek iegūts saraksts ar studentu vārdiem, kur katram studentam klāt ir norādīts viņa kopējais kredītpunktu skaits pa visiem kursiem. Attēla

kreisajā pusē dotais uzģenerētais vizuālais vaicājums ir tāds pats, kāds ir sākotnējais vizuālais vaicājums, līdz ar to var secināt, ka šīs vizuālais vaicājums ir uzģenerēts pareizs.

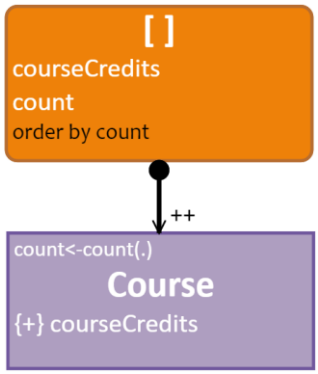
Tālāk tika apskatīts vaicājums ar apakšvaicājumu, kurā galvenais mezgls ir kā slānis apakšvaicājumu rezultātu modificēšanai un atšķirībā no iepriekšējā vaicājuma, sākotnēji tas ir veidots kā tekstuāls SPARQL vaicājums. Izveidotais tekstuālais SPARQL vaicājums un no tā uzģenerētais vizuālais vaicājums ir dots 4.26. att.

```
PREFIX : <http://lumii.lv/ontologies/UnivExample.owl#>
SELECT ?courseCredits ?count WHERE{
  {SELECT ?courseCredits (COUNT(?Course) AS ?count) WHERE{
        ?Course a :Course; :courseCredits ?courseCredits.
    }
    GROUP BY ?courseCredits
  }
}
ORDER BY ?count
```



4.26. att. **Vaicājums ar apakšvaicājumu, kas sākotnēji veidots kā tekstuāls vaicājums**

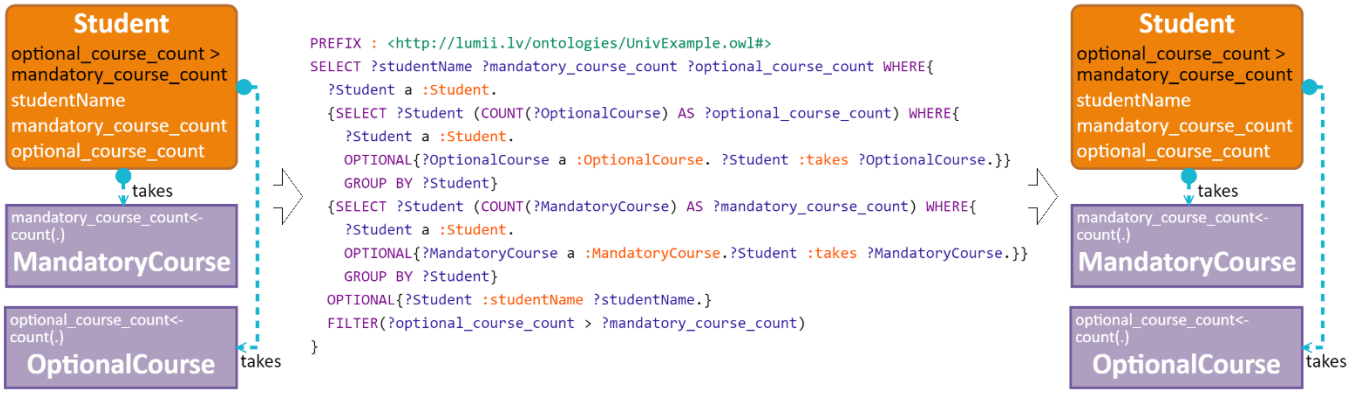
Vaicājumā tiek atlasīti kursu kredītpunkti un to biežums, kas ir kārtoti pēc biežuma. Kursi tiek grupēti pēc kredītpunktu skaita un kredītpunktu skaita biežums tiek iegūts saskaitot kursus katrā grupā. Uzģenerētais vizuālais vaicājums nav pareizs, tajā galvenajā mezglā ir sapludināts kopā ārējais vaicājums un apakšvaicājums. Tekstuālajam vaicājumam atbilstošs vizuālais vaicājums ir dots 4.27. att. Šis vaicājums tika vizualizēts manuāli.



4.27. att. **Manuāli vizualizēts vaicājums ar apakšvaicājumu**

Vizuālajā vaicājumā, kas dots 4.27. att., galvenais mezgls ir attēlots kā slānis pāri apakšvaicājumam un klases nosaukuma vietā ir attēlotas kvadrātiekavas. Galvenajā mezglā ir norādīti arī atribūti, kas jāiekļauj vaicājuma rezultātā. No galvenā mezgla iziet saite uz apakšvaicājumu, kurā tiek atlasīti un saskaitīti kursi, kas ir grupēti pēc kredītpunktu skaita.

Kā nākošais tika apskatīts vaicājums, kas sākotnēji ir veidots kā vizuāls vaicājums rīkā *ViziQuer*. Izveidotais vizuālais vaicājums ir dots 4.28. att. kreisajā pusē, pa vidu ir dots no tā uzģenerētais tekstuālais vaicājums. No tekstuālā vaicājuma automātiski vizualizētais vaicājums ir dots attēla labajā pusē.



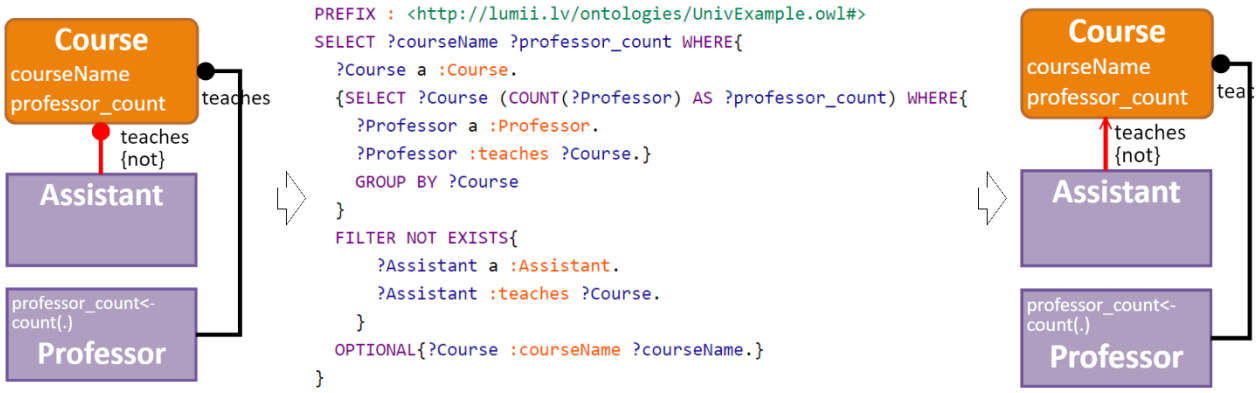
4.28. att. **Vaicājums ar apakšvaicājumiem un neobligātu saiti, kas sākotnēji veidots kā vizuāls vaicājums**

Attēlotajā vaicājumā tiek atlasīti visi studentu vārdi, kuriem neobligāto kursu skaits ir lielāks nekā obligāto kursu skaits. Šajā vaicājumā atšķirībā no iepriekš apskatītajiem vaicājumiem ir doti divi apakšvaicājumi un saites starp ārējo vaicājumu un apakšvaicājumiem ir neobligātas. Norādot saiti kā neobligātu, šajā gadījumā tiks atlasīti arī studenti kuriem nav neviena obligātā vai neobligātā kursa, tādā gadījumā attiecīgais kursu skaits būs 0. Ja saites būtu obligātas, tad obligāto un neobligāto kursu skaits nekad nebūtu 0, jo tiktu atlasīti tikai tie studenti, kuriem ir vismaz viens obligātais un neobligātais kurss.

Vizuālajā vaicājumā, kas dots 4.28. att., galvenais mezgls ir klase *Student*. Galvenajā mezglā ir norādīti tādi atribūti kā studenta vārds (*studentName*), obligāto kursu skaits (*mandatory_course_count*) un neobligāto kursu skaits (*optional_course_count*), kas jāiekļauj vaicājuma rezultātā. Pie galvenā mezgla ir norādīts arī nosacījums, ka neobligāto kursu skaitam ir jābūt lielākam nekā obligāto kursu skaitam (*optional_course_count* > *mandatory_course_count*). No galvenā mezgla iziet divas neobligātas saites uz apakšvaicājumiem, kas ir attēlotas ar zilu, raustītu bultu. Apakšvaicājumos tiek attiecīgi atlasīti studenti kopā ar obligāto kursu skaitu un studenti kopā ar neobligāto kursu skaitu.

Salīdzinot 4.28. att. kreisajā pusē un labajā pusē dotos vizuālos vaicājumus, var redzēt, ka tie ir vienādi un šāds vaicājums tiek vizualizēts veiksmīgi.

Arī nākamais vaicājums sākotnēji tika veidots kā vizuāls vaicājums, tas ir redzams 4.29. att. kreisajā pusē, savukārt labajā pusē ir dots uzģenerēts vizuālais vaicājums. Šī vaicājuma mērķis ir aplūkot rīka spēju vizualizēt vaicājumu, kurā starp ārējo vaicājumu un apakšvaicājumu ir saite ar noliegumu.



4.29. att. **Vaicājums ar apakšvaicājumiem un nolieguma saiti, kas sākotnēji veidots kā vizuāls vaicājums**

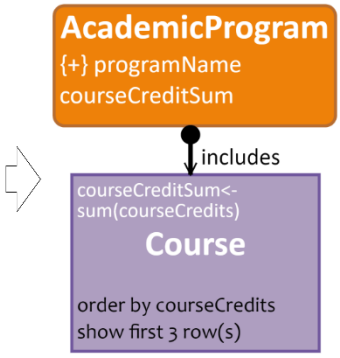
Vaicājumā tiek atlasīti kursu nosaukumi kopā ar profesoru skaitu tiem kursiem, kurus nepasniedz neviens asistents. Vizuālajā vaicājumā, kas dots 4.29. att. kreisajā pusē, galvenais mezgls ir klase *Course*. Galvenajā mezglā ir norādīti tādi atribūti kā kursa nosaukums (*courseName*) un pasniedzēju skaits (*professor_count*). No galvenā mezgla iziet divas saites uz apakšvaicājumiem. Uz vienu no apakšvaicājumiem iziet nolieguma saite, kas nosaka, ka kursu nepasniedz asistents. Savukārt otrā apakšvaicājumā tiek atlasīts kurss kopā ar pasniedzēju skaitu.

Salīdzinot 4.29. att. kreisajā pusē un labajā pusē dotos vizuālos vaicājumus, var redzēt, ka tie mazliet atšķiras. Labajā pusē esošajā vizuālajā vaicājumā saite ar noliegumu apzīmē attiecību starp vaicājumu un apakšvaicājumu, taču kreisajā pusē vizuālajā vaicājumā saite apzīmē noliegumu starp klasi un nosacījumu. Pēc rakstā [37] dotās informācijas, abas šīs saites ir līdzvērtīgas. Automātiski vizualizētais vaicājums ir pareizs, kaut arī tas mazliet atšķiras no sākotnējā vaicājuma.

Rakstā par *ViziQuer* [37] ir minēti arī globāli apakšvaicājumi, kurus ir nepieciešams izmantot gadījumos, kad kārtošanas un limitēšanas modifikatori ir būtiski apakšvaicājumā. Tāpēc ar šo vaicājumu tiek aplūkota rīka spēja vizualizēt globālus apakšvaicājumus. Izveidotais tekstuālais vaicājums ar kārtošanas un limita modifikatoriem apakšvaicājumā ir dots 4.30. att. kreisajā pusē. Izpildot vaicājumu tiek atlasīti akadēmisko programmu nosaukumi kopā ar kredītpunktu skaitu no trīs kursiem ar vismazāko kredītpunktu skaitu. Labajā pusē 4.30. att. ir dots automātiski izveidotais vizuālais vaicājums. Tajā kā galvenais mezgls ir norādīta klase *AcademicProgram*, pie kuras tiek

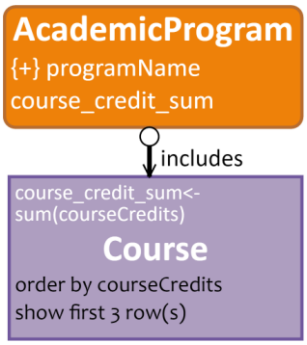
atlasīti tādi atribūti kā programmas nosaukums (*programName*) un atlasīto kursu kredītpunktu summa (*courseCreditSum*). No galvenā mezgla iziet saite uz apakšvaicājumu. Aapakšvaicājumā tiek atlasītas akadēmiskās programmas un kredītpunktu summa no trīs kursiem ar vismazāko kredītpunktu skaitu. Šis izveidotais vizuālais vaicājums nav pareizs, jo attēlotā saite apzīmē apakšvaicājumu, nevis globālu apakšvaicājumu. Izpildot uzģenerēto vizuālo vaicājumu, netiks ņemti vērā kārtības un limitēšanas modifikatori, jo apakšvaicājums nav dots kā globāls apakšvaicājums.

```
PREFIX : <http://lumii.lv/ontologies/UnivExample.owl#>
SELECT ?programName ?courseCreditSum WHERE{
  ?AcademicProgram a :AcademicProgram; :programName ?programName.
  {SELECT ?AcademicProgram (SUM(?courseCredits) AS ?courseCreditSum) WHERE{
    ?Course a :Course; :courseCredits ?courseCredits.
    ?AcademicProgram :includes ?Course.}
    ORDER BY ?courseCredits
    LIMIT 3
  }
}
```



4.30. att. **Vaicājums ar globālu apakšvaicājumu, kas sākotnēji veidots kā tekstuāls vaicājums**

Attēlā 4.31. att. ir redzams manuāli izveidots vizuālais vaicājums, kurš atbilst 4.30. att. dotajam tekstuālajam vaicājumam. Starp uzģenerēto vizuālo vaicājumu un manuāli izveidoto sagaidāmo vaicājumu ir neliela vizuāla atšķirība – saite starp ārējo vaicājumu un apakšvaicājumu atšķiras. Globāli apakšvaicājumi *ViziQuer* notācijā tiek attēloti ar bultu, kas sākas no nepildīta apļa.



4.31. att. **Manuāli vizualizēts vaicājums ar globālu apakšvaicājumu**

Šajā apakšnodaļā tika apskatīti pieci darba ietvaros izveidotie vaicājumi ar apakšvaicājumiem, kā arī rīka *ViziQuer* spēja tos vizualizēt. No apskatītajiem vaicājumiem trīs tika vizualizēti veiksmīgi, taču divi tika vizualizēti neveiksmīgi. Šiem vaicājumiem autore ir sniegusi arī manuāli izveidotus vaicājumus, kādiem tiem vajadzētu būt veiksmīgi vizualizējot.

REZULTĀTI

Darba ietvaros tika veikts pētījums ar mērķi izpētīt vaicājumu vizualizēšanas iespējas. Darba teorētiskajā daļā tika veikta literatūras un standartu apskate par semantisko tīmekli un tā atbalstošajām tehnoloģijām kā SPARQL, RDF un OWL. Tika aplūkoti vairāki Latvijas Universitātē izstrādāti noslēguma darbi saistībā ar rīku *ViziQuer*, lai noskaidrotu, kas ir paveikts un izpētīts pirms autores darba uzsākšanas.

Lai apzinātu citu rīku spējas automātiski vizualizēt tekstuālus vaicājumus, tika veikta sistemātiska literatūras apskate. Tika aplūkoti 10 dažādi vaicājumu vizualizācijas rīki. Rezultātā tika noskaidrots, ka tikai vienam no 10 aplūkotajiem rīkiem ir izstrādāta funkcionalitāte vizuālo vaicājumu ģenerēšanai no tekstuāliem SPARQL vaicājumiem – rīkam *OptiqueVQS*.

Praktiskajā daļā balstoties uz SPARQL vaicājumvalodas specifikāciju, tika izdalītas sešas vaicājumu klases. Atbilstoši vaicājumu klasēm tika izveidoti vairāki vaicājumi, lai novērtētu *ViziQuer* spēju vizualizēt dažādu klašu vaicājumus. Tika analizēta rīka spēja vizualizēt dažādus vaicājumus, gan tādus, kas sākotnēji ir veidoti kā tekstuāli SPARQL vaicājumi un pēc tam vizualizēti, gan arī tādi, kas ir sākotnēji veidoti rīkā *ViziQuer* kā vizuāli vaicājumi, no kuriem automātiski iegūts tekstuāls vaicājums, kurš tālāk ir automātiski vizualizēts.

Kopā tika aplūkoti 25 vaicājumu piemēri, 1. tabulā ir dots kopējais apskatīto vaicājumu skaits katrai vaicājuma klasei kā arī rezultāti veiksmīgi un neveiksmīgi vizualizētajiem vaicājumiem. Veiksmīgi un neveiksmīgi vizualizēti vaicājumi ir izdalīti pēc tā, vai vaicājums ir sākotnēji veidots kā tekstuāls vaicājums vai vizuāls vaicājums rīkā *ViziQuer*. Tiesa gan pēc iegūtajiem rezultātiem nevar spriest par to, vai tekstuāli vaicājumi, kas ir iegūti no sākotnēji vizualizētiem vaicājumi, tiek vizualizēti veiksmīgāk nekā vaicājumi, kas sākotnēji nav veidoti rīkā *ViziQuer*, jo katrā vaicājumā tika pārbaudīta rīka spējas vizualizēt kādu konkrētu vaicājumvalodas iespēju. Kopā no 25 apskatītajiem vaicājumiem veiksmīgi tika vizualizēti 18, taču neveiksmīgi 7 vaicājumi.

1. tabula

Veiksmīgi un neveiksmīgi vizualizēto vaicājumu skaits katrā klasē

Vaicājumu klase	**Apskatīto vaicājumu skaits**	**Veiksmīgi vizualizēti vaicājumi**		**Neveiksmīgi vizualizēti vaicājumi**	
		Tekstuāli	**Vizuāli**	**Tekstuāli**	**Vizuāli**
Konjunktīvi vaicājumi	4	2	2	0	0

Vaicājumu klase	Apskatīto vaicājumu skaits	Veiksmīgi vizualizēti vaicājumi		Neveiksmīgi vizualizēti vaicājumi	
		Tekstuāli	Vizuāli	Tekstuāli	Vizuāli
Disjunktīvi vaicājumi	5	2	1	1	1
Vaicājumi ar apkopošanas funkcijām	4	2	1	1	0
Vaicājumi ar noliegumu	4	1	2	1	0
Vaicājumi ar risinājumu virknes modifikatoriem	3	1	1	1	0
Vaicājumi ar apakšvaicājumiem	5	0	3	2	0

Atbilstoši atklātajām nepilnībām darbā ir doti ieteikumi to novēršanai, piemēram, vizuālās notācijas paplašināšanai un alternatīvas vaicājuma formas piedāvāšana gadījumos, kad vaicājumu nav iespējams vizualizēt dēļ vizuālās notācijas neesamības. Kā arī darbā ir doti manuāli veidoti sagaidāmie vizuālie vaicājumi gadījumos, kad vizuālā notācija ir pieejama, taču vaicājums tiek neveiksmīgi vizualizēts. Kopsavilkums par atklātajām nepilnībām un ieteikumiem to novēršanai ir dots 1. pielikumā.

Izveidotie vaicājumi ir apkopoti *ViziQuer* projektā, kurā ir izveidotas sešas diagrammas atbilstoši vaicājumu dalījumam klasēs. Ekrānuzņēmums no izveidotā projekta ir dots 2. pielikumā. Katra diagramma satur klasei atbilstošus vaicājumus un tā ir organizēta trīs kolonnās: sākotnējais vizuālais vai tekstuālais vaicājums, uzģenerētais vizuālais vaicājums un manuāli izveidots atbilstošs vizuālais vaicājums gadījumā, ja vaicājums tika vizualizēts neveiksmīgi. Manuāli izveidots vaicājums *ViziQuer* projektā ir dots tikai tādā gadījumā, ja to ir iespējams vizualizēt izmantojot *ViziQuer* notāciju. Ekrānuzņēmums no diagrammas ar vaicājumiem ar apkopošanas funkcijām ir dots 3. pielikumā. Izveidotais *ViziQuer* projekts ir pieejams publiskā repozitorijā[16].

[16] https://github.com/Ingaaa/MiniUniQueryExamples

SECINĀJUMI

Pēc vizuālo vaicājumu veidošanas rīku apskates, var secināt, ka jaunas funkcionalitātes izstrāde rīkam *ViziQuer* vizuālo vaicājumu ģenerēšanai no tekstuāliem SPARQL vaicājumiem, būs unikāla, jo šobrīd šādas funkcionalitātes izstrāde ir veikta tikai vienam no desmit apskatītajiem rīkiem – *OptiqueVQS*. Arī citu rīku nākotnes plānos nav minēta šādas funkcionalitātes izstrāde. Apskatītie rīki galvenokārt nākotnē plāno izstrādāt jaunu funkcionalitāti, kas vairāk atbalstītu lietotāju izpratni, piemēram rīks *QueryVOWL* apsver vaicājuma daļu attēlošana dabiskajā valodā. Nākotnes plānos parādās arī atbalsta nodrošināšana jaunām SPARQL vaicājumvalodas iespējām, piemēram, rīks *SparqlFilterFlow* plāno nodrošināt atbalstu *DESCRIBE* un *CONSTRUCT* vaicājumu izveidošanai papildus esošajiem *SELECT* un *ASK* vaicājumiem.

Kaut arī vizualizācijas modulis ir izstrādes stadijā, tas spēj veiksmīgi vizualizēt dažādas SPARQL vaicājumvalodas pamatkonstrukcijas. Jāņem vērā, ka autore ar nodomu veidoja tādus vaicājumus, kurus rīks *ViziQuer* varētu vizualizēt neveiksmīgi, lai pēc iespējas pilnīgāk atklātu ar vizualizāciju saistītās problēmas.

Darba veidošanas laikā, atklātās vizualizēšanas nepilnības var iedalīt divos galvenajos veidos:

- vizuālā notācija neatbalsta kādu vaicājumvalodas konstrukciju, tas nozīmē, ka tekstuālu vaicājumu nav iespējams automātiski vizualizēt un to nav iespējams arī manuāli vizualizēt, jo nav atbilstošas vizuālās notācijas, taču šādā gadījumā var izmantot citu vaicājuma formu, lai iegūtu tādu pašu rezultātu;
- pastāv atbilstoša vizuālā notācija, taču vizualizācijas algoritms neveiksmīgi vizualizē vaicājumu, šajā gadījumā vaicājumu ir iespējams vizualizēt manuāli.

Izveidotajā vaicājumu kopā ir iekļautas dažādas SPARQL vaicājumvalodas pamatkonstrukcijas, taču pēc autores domām to ir iespējams papildināt ar tādiem vaicājumiem, kas satur jaunu vērtību veidošanu ar *CONCAT* funkciju un vērtību piešķiršanu ar *BIND*, iekšējiem filtriem jeb gadījumus, kad aiz *FILTER* esošajā nosacījumā ir dots apakšfiltrs un īpašību ceļus.

Darba rezultāti ir praktiski izmantojami tālākā *ViziQuer* rīka pilnveidē, piedāvājot sistemātiski izveidotus vaicājumu piemērus, kurus vizualizācijas algoritmam ir jāspēj vizualizēt veiksmīgi, kā arī neatkarīgi no izstrādātāja ir novērtēta esošā rīka funkcionalitātes situācija.

IZMANTOTĀ LITERATŪRA UN AVOTI

[1] W3C, RDF Working Group, «RDF» 25.02.2014. [Tiešsaiste]. Pieejams: https://www.w3.org/RDF/. [Piekļūts 16.05.2020].

[2] W3C, «Query» [Tiešsaiste]. Pieejams: https://www.w3.org/standards/semanticweb/query. [Piekļūts 16.05.2020].

[3] W3C, «HTML+RDFa 1.1 - Second Edition» 17.03.2015. [Tiešsaiste]. Pieejams: https://www.w3.org/TR/html-rdfa/. [Piekļūts 16.05.2020].

[4] W3C, «Semantic Web» [Tiešsaiste]. Pieejams: https://www.w3.org/standards/semanticweb/. [Piekļūts 16.05.2020].

[5] Cisco, «Cisco Annual Internet Report (2018–2023) White Paper» 17.02.2020. [Tiešsaiste]. Pieejams: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. [Piekļūts 16.05.2020].

[6] H. Knublauch, D. Oberle, P. Tetlow, E. Wallace, J. Z. Pan un M. Uschold, «A Semantic Web Primer for Object-Oriented Software Developers, W3C Editor's Draft» 09.03.2006. [Tiešsaiste]. Pieejams: https://www.w3.org/2001/sw/BestPractices/SE/ODSD/. [Piekļūts 16.05.2020].

[7] W3C OWL Working Group, «OWL 2 Web Ontology Language Document Overview (Second Edition)» 11.12.2012. [Tiešsaiste]. Pieejams: https://www.w3.org/TR/owl2-overview/. [Piekļūts 16.05.2020].

[8] O. Corcho, F. Priyatna un D. Chaves-Fraga, «Towards a new generation of ontology based data access» *Semantic Web,* sēj. 11, nr. 1, pp. 153-160, 2020.

[9] I. Herman, «SPARQL is a Recommendation» 15.01.2008. [Tiešsaiste]. Pieejams: https://www.w3.org/blog/SW/2008/01/15/sparql_is_a_recommendation/. [Piekļūts 16.05.2020].

[10] W3C, «SPARQL 1.1 Overview» 21.03.2013. [Tiešsaiste]. Pieejams: https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/. [Piekļūts 16.05.2020].

[11] W3C, «SPARQL 1.1 Query Language» 21.03.2013. [Tiešsaiste]. Pieejams: https://www.w3.org/TR/2013/REC-sparql11-query-20130321/. [Piekļūts 16.05.2020].

[12] G. Bārzdiņš, S. Rikačovs, M. Veilande un M. Zviedris, «Ontological Re-engineering of Medical Databases» *Proceedings of the Latvian Academy of Sciences,* sēj. 63, nr. 4/5 (663/664), p. 156–158, 2009.

[13] K. Jēriņš, «ViziQuer grafiku kompilēšana par SPARQL» Latvijas Universitāte. Datorikas fakultāte, Rīga, 2010.

[14] R. Laganovskis, «ViziQuer pieprasījumu grafiskās ievadformas realizācija» Latvijas Universitāte. Datorikas fakultāte, Rīga, 2010.

[15] J. Hodakovska, «ViziQuer rīka realizācija tīmekļa vidē» Latvijas Universitāte. Datorikas fakultāte, Rīga, 2016.

[16] T. Ozols, «Izteiksmju sintakses priekšā teicējs ViziQuer rīkā» Latvijas Universitāte. Datorikas fakultāte, Rīga, 2018.

[17] O. Pešudovs, «Diagrammatisku vaicājumu veidošanas servisi tīmekļa vidē» Latvijas Universitāte. Datorikas fakultāte, Rīga, 2018.

[18] J. Hodakovska, «Lietotāja servisi vizuālā vaicājumu rīkā» Latvijas Universitāte. Datorikas fakultāte, Rīga, 2018.

[19] I. Stinkevičs, «Grafiskie vaicājumi relāciju datubāzēs» Latvijas Universitāte. Datorikas fakultāte, Rīga, 2018.

[20] T. Kirtovskis, «Praktiski vizuāli datu vaicājumi» Latvijas Universitāte. Datorikas fakultāte, Rīga, 2019.

[21] R. Kolduns, «Vizuālo vaicājumu rezultātu attēlošana gala lietotājam» Latvijas Universitāte. Datorikas fakultāte, Rīga, 2019.

[22] P. Brereton, B. A. Kitchenham, D. Budgen, MarkTurner un M. Khalil, «Lessons from applying the systematic literature review process within the software engineering domain» *Journal of Systems and Software,* sēj. 80, nr. 4, pp. 571-583, 2007.

[23] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey un S. Linkman, «Systematic literature reviews in software engineering – A systematic literature review» *Information and Software Technology,* sēj. 51, nr. 1, pp. 7-15, 2009.

[24] O. Ambrus, K. Möller un S. Handschuh, «Konduit VQB: A visual query builder for SPARQL on the social semantic desktop» Digital Enterprise Research Institute (DERI), National University of Ireland, Galway (NUIG), 2010.

[25] L. Dragan, A. Passant, T. Groza un S. Handschuh, «From Personal Notes to Linked Social Media» *Linked Data Meets Artificial Intelligence, Papers from the 2010 AAAI Spring Symposium, Technical Report SS-10-07*, Stanford, 2010.

[26] F. Haag, S. Lohmann, S. Siek un T. Ertl, «Visual Querying of Linked Data with QueryVOWL» Institute for Visualization and Interactive Systems, University of Stuttgart, Stuttgart, 2015.

[27] F. Benedetti, S. Bergamaschi un L. Po, «LODeX: A tool for Visual Querying Linked Open Data» *The 14th International Semantic Web Conference (ISWC-2015)*, Bethlehem, 2015.

[28] F. Haag, S. Lohmann un T. Ertl, «SparqlFilterFlow: SPARQL Query Composition for Everyone» *ESWC 2014: The Semantic Web: ESWC 2014 Satellite Events*, Stuttgart, 2014.

[29] A. Soylu, E. Kharlamov, D. Zheleznyakov, E. Jimenez-Ruiz, M. Giese un I. Horrocks, «OptiqueVQS: Visual Query Formulation for OBDA» *CEUR Workshop Proceedings*, 2014.

[30] Y. Mumtaz, «Collaborative Query Formulation in a Visual Query System» University of Oslo, Oslo, 2015.

[31] J. Aasman un K. Cheetham, «RDF Browser for Data Discovery and Visual Query» Franz Inc, Oakland, 2011.

[32] H. Vargas, C. B. Aranda un A. Hogan, «RDF Explorer: A Visual Query Builder for Semantic Web Knowledge Graphs,» *ISWC 2019 Satellites*, Auckland, 2019.

[33] G. V.-G. M. G. Simen Heggestøyl, «Visual Query Formulation for Linked Open Data: The Norwegian Entity Registry Case,» *NIK 2014*, Fredrikstad, 2014.

[34] M. S. Q. M. D. Z. S. D. A. H. Syeda Sana e Zainab, «SPARQL Query Formulation and Execution using FedViz,» *International Semantic Web Conference*, Bethlehem, 2015.

[35] A. S. Imen Sarray, «Assisted Composition of Linked Data Queries» *Proceedings of KEOD 2019*, Vienna, 2019.

[36] A. Soylu, E. Kharlamov, D. Zheleznyakov, E. Jimenez-Ruiz, M. Giese, M. G. Skjæveland, D. Hovland, R. Schlatte, S. Brandt, H. Lie un I. Horrocks, «OptiqueVQS: a Visual Query System over Ontologies for Industry» *Semantic Web journal,* sēj. 9, nr. 5, pp. 627-660, 2018.

[37] K. Čerāns, J. Bārzdiņš, A. Šostaks, J. Ovčiņņikova, L. Lāce, M. Grasmanis un A. Sproģis, «Extended UML Class Diagram Constructs for Visual SPARQL Queries in ViziQuer/web» *Proceedings of the Third International Workshop on Visualization and Interaction for Ontologies and Linked Data*, Vienna, 2017.

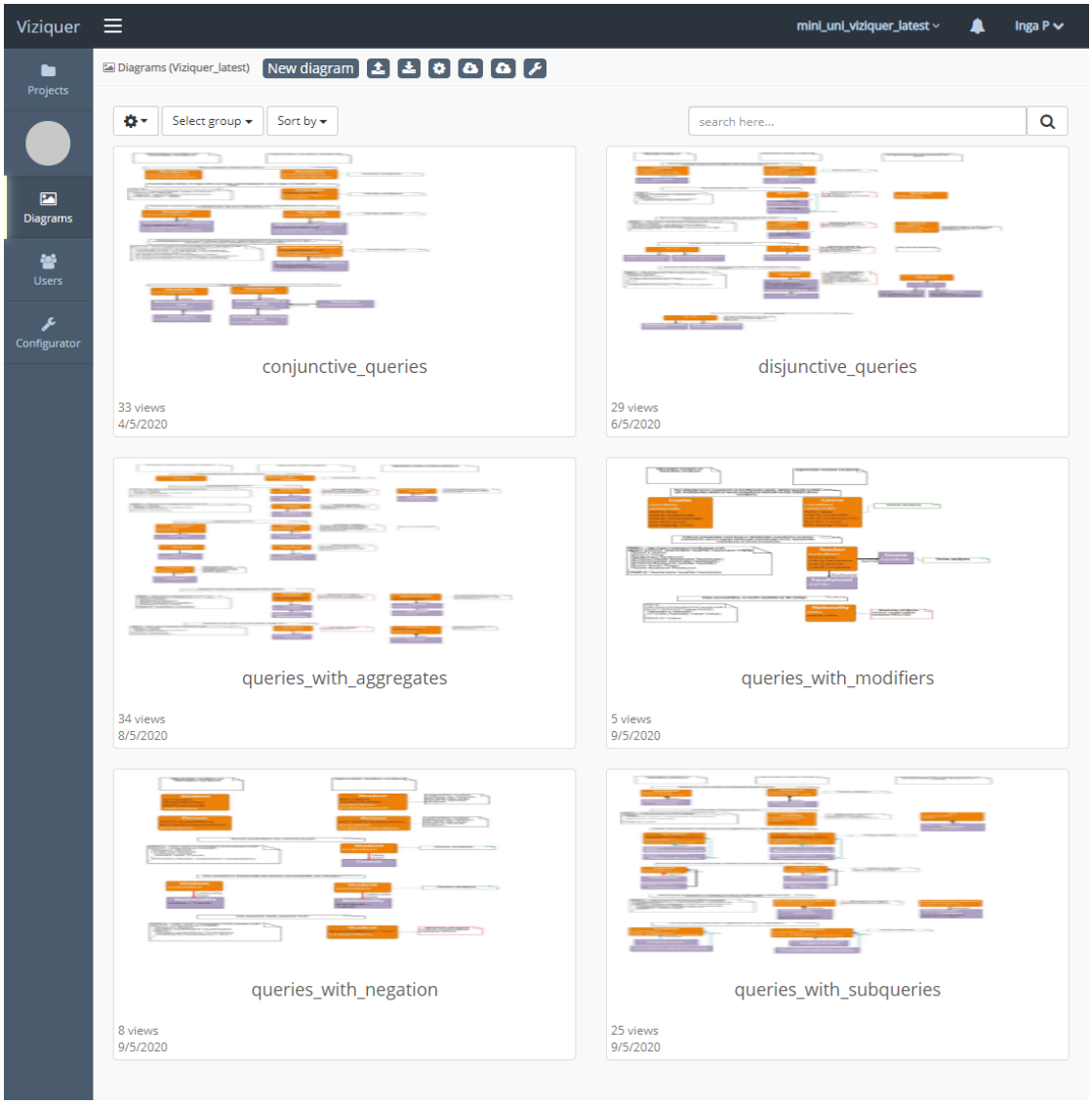
PIELIKUMI

1. pielikums. Atklāto nepilnību un risinājumu kopsavilkums

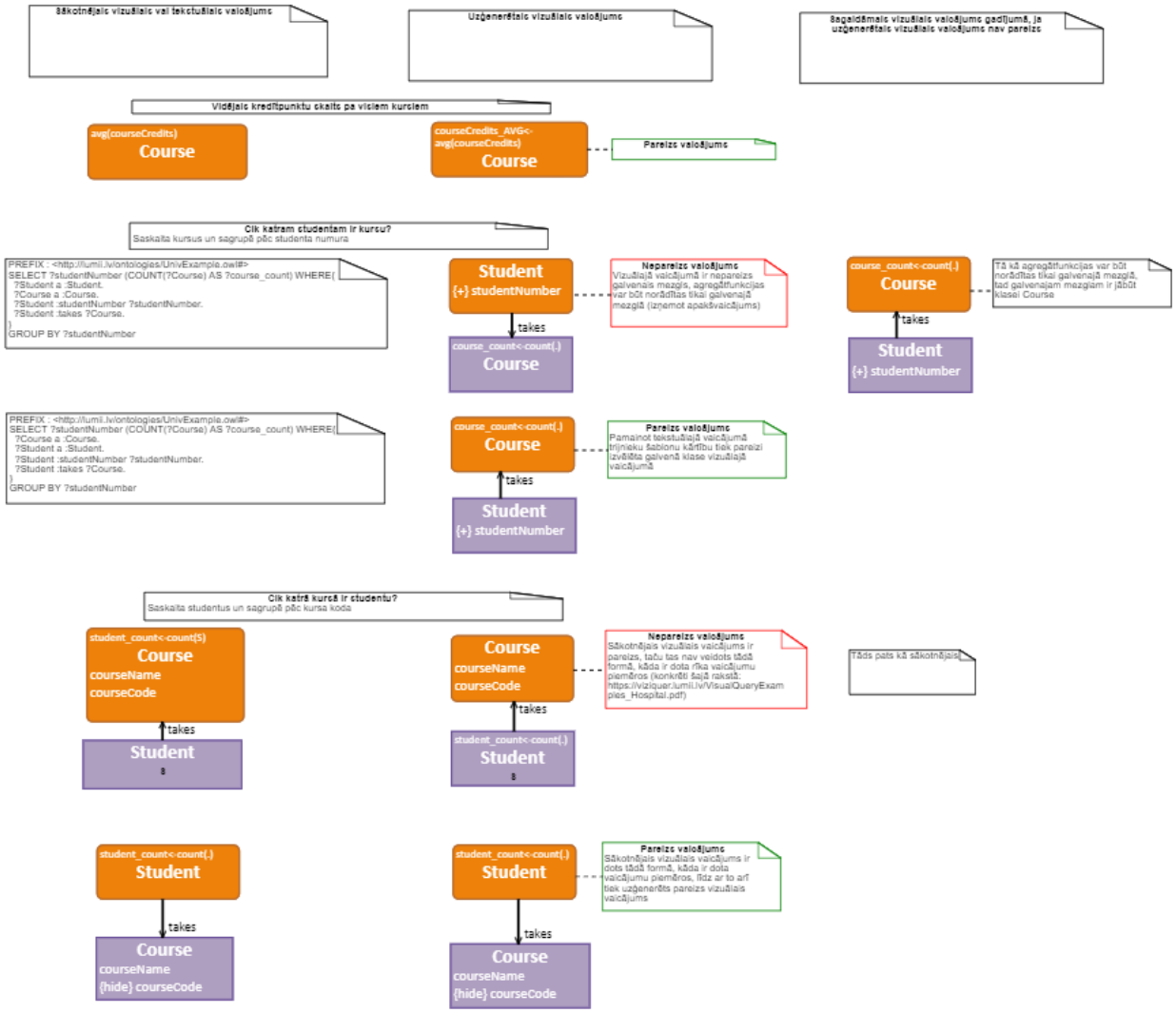
Vaicājuma klase	Atklātās nepilnības	Risinājums
Konjunktīvi vaicājumi	-	-
Disjunktīvi vaicājumi	Netiek pareizi vizualizēti neobligātie trijnieku šabloni (*OPTIONAL*), kur figūriekavās ir dots vairāk nekā viens trijnieku šablons.	Jāveic izmaiņas vizualizācijas algoritmā, lai vaicājumi ar vairākiem trijnieku šabloniem pie *OPTIONAL* tiktu pareizi vizualizēti.
	Pie *OPTIONAL* nav iespējams norādīt ierobežojumus izmantojot *FILTER*.	*ViziQuer* notācija nenodrošina šādu iespēju, risinājums varētu būt pievienot iespēju norādīt nosacījumu pie neobligātajiem atribūtiem. **Course** courseName {+} courseCode {hide} courseExtInfo courseCredits > 5
	ViziQuer nodrošina notāciju vizuālu vaicājumu veidošanai ar *UNION*, taču tekstuāli vaicājumi tiek neveiksmīgi vizualizēti.	Jāveic izmaiņas vizualizācijas algoritmā, lai vaicājumi ar *UNION* tiktu pareizi apstrādāti.
Vaicājumi ar apkopošanas funkcijām	Veiksmīga vizualizēšana ir atkarīga no vaicājumā dotās trijnieku šablonu kārtības. Trijnieku šablonu kārtība ietekmē to, kura klase tiks attēlota kā galvenais vaicājuma mezgls. Līdz ar to var būt gadījumi, kad apkopošanas	Veikt izmaiņas vizualizācijas algoritmā, lai tiktu pareizi noteikts galvenais mezgls tā, lai nerastos situācija, kad apkopošanas funkcija ir norādīta pie nosacījumu klases.

Vaicājuma klase	Atklātās nepilnības	Risinājums
	funkcijas nav norādītas pie galvenā mezgla.	
	ViziQuer vizuālā notācija neatbalsta *HAVING* klauzulu.	Vaicājumu ar *HAVING* klauzulu ir iespējams pārrakstīt citā formā – kā vaicājumu ar apakšvaicājumu.
Vaicājumi ar noliegumu	Iespējams uzlabot *EXISTS* attēlošanu vizuālajā vaicājumā – *EXISTS* netiek attēlots kā nosacījums, bet gan kā obligāts klases atribūts.	Attēlot *EXISTS* kā nosacījumu, nevis kā obligātu klases atribūtu.
	ViziQuer vizuālā notācija neatbalsta *MINUS*.	Darbā netika apskatīts alternatīvas formas vaicājums, taču šo konstrukciju ir iespējams aizstāt ar globāliem apakšvaicājumiem [37].
Vaicājumi ar risinājumu virknes modifikatoriem	*ViziQuer* vizuālā notācija neatbalsta *REDUCED*.	Vizuāli tas varētu tikt attēlots tāpat kā modifikators *DISTINCT*, taču pēc rīka autoru dotās informācijas [37], pastāv neskaidrības saistībā ar citām iespējamām implementācijām, piemēram, vaicājumu tulkošanu uz SQL.
Vaicājumi ar apakšvaicājumiem	Gadījumos, kad galvenais mezgls ir kā slānis apakšvaicājumu rezultātu modificēšanai nevis klase, vaicājums netiek pareizi vizualizēts	Jāveic izmaiņas vizualizācijas algoritmā, lai šādi gadījumi tiktu pareizi apstrādāti.
	Vaicājumi ar globāliem apakšvaicājumiem netiek pareizi vizualizēti.	Jāveic izmaiņas vizualizācijas algoritmā, lai globāli apakšvaicājumi tiktu pareizi apstrādāti.

2. pielikums. Ekrānuzņēmums no *ViziQuer* projekta



3. pielikums. Ekrānuzņēmums no *ViziQuer* projekta diagrammas



DOKUMENTĀRĀ LAPA

Maģistra darbs "Vizuālo vaicājumu ģenerēšana no tekstuālās formas" izstrādāts LU Datorikas fakultātē.

Darba teksta galīgā versija izgatavota 17.05.2020.

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

Autors: _____
(Autora paraksts un datums)

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

Darba vadītājs: _____
(Vadītāja paraksts un datums)

Darbs iesniegts **maģistrantūras sekretariātā** _____.
(Iesniegšanas datums)

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

Studiju metodiķe: _____.
(Metodiķes paraksts)

Recenzents: docents Dr. dat. Elīna Kalniņa
(Akad. amats, zin. grāds, vārds, uzvārds)

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____
(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____
(Sekretāra paraksts)