LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

DATU SHĒMAS IZGUVE NO WIKIDATA VIZUĀLO VAICĀJUMU ATBALSTAM

BAKALAURA DARBS

Autors: **Ginters Juris Vehi**
Studenta apliecības Nr.: gv17018
Darba vadītājs: profesors Dr. dat. Kārlis Čerāns

RĪGA 2021

ANOTĀCIJA

Semantiskajam tīmeklim paliekot arvien apjomīgākam rodas nepieciešamība pēc dažādiem rīkiem un lietojumprogrammām, kas palīdz analizēt šo lielo datu apjomu tai skaitā arī saistīto datu avotus. Un šiem rīkiem ir svarīgi zināt šo datu avotu shēmas, lai palīdzētu ar datu analīzi.

Šī bakalaura darba mērķis ir apskatīt un iepazīties ar populārākajiem semantiskā tīmekļa saistīto datu avotiem kā "Wikidata", "DBpedia" un izstrādāt skriptu, kas palīdzētu izgūt "Wikidata" shēmu.

Darba apjoms ir 40 lappuses, tas sastāv no 4 nodaļām un darbā ir izmantoti 19 dažādi literatūras avoti.

Atslēgvārdi: Semantiskais tīmeklis, saistīto datu avoti, Wikidata, DBpedia.

ABSTRACT

DATA SCHEMA EXTRACTION FROM WIKIDATA FOR GRAPHICAL QUERY SUPPORT

As the semantic web becomes even bigger and contains ever increasing amounts of information, where large portion of this information resides in linked data sources like “Wikidata” and “DBpedia”. There arises a need for tools and programs that would help in analyzing these data sources and for them a schema of these sources is needed.

The goal of this bachelor paper is to go over most popular linked data resources like “Wikidata” and “DBpedia” and develop a script for extracting the “Wikidata” schema.

This bachelor paper consists of 40 pages, which are grouped in 4 chapters. The work references 19 different information sources.

Keywords: Semantic web, linked data sources, Wikidata, Dbpedia

SATURA RĀDĪTĀJS

APZĪMĒJUMU SARAKSTS

RDF (“Resource Description Framework” angliski) – standarta modelis, kas tiek izmantos datu apmaiņai tīmeklī. Tā specifikāciju kopu uztur globālā tīmekļa konsorcijs. Datu aprakstīšana šajā modelī ir semantiskā tīmekļa iniciatīvas pamatā.

Wikidata – Atvērto strukturēto saistīto datu avots, kurš ir lasāms gan cilvēkiem, gan datoriem. Informācijas avots citiem “wikis”. Dati aprakstīti modelī, kas ir līdzīgs RDF.

Dbpedia – Projekts ar mērķi aprakstīt informāciju no “Wikipedia” strukturētā formātā. Viens no plašākajiem tīmekļa saistīto datu avotiem. Dati aprakstīti izmantojot RDF datu shēmu.

XML (“eXtensible Markup Language” angliski) – Iezīmēšanas valoda, kas apraksta kopu ar likumiem priekš dokumentu aprakstīšanas formātā, kas būtu gan cilvēku, gan datoru lasāms.

RDFS (“Resource Description Framework Schema” angliski) – ontoloģija, kas papildina RDF datu modeli, pievienojot tam papildus klases un īpašības.

OWL (“Web Ontology Language” angliski) – tīmekļa ontoliģijas valoda, kas ir izmantojama, lai papildinātu ontoloģijas, tajā skaitā RDF.

HTML (“HyperText Markup Language” angliski) – Standarta iezīmēšanas valoda, kas tiek izmantota, lai aprakstītu dokumentus, jeb tīmekļa vietnes, kas tiek attēlotas globālajā tīmeklī.

API (“Application Programming Interface” angliski) – Interfeiss, kas apraksta saistības starp dažādām lietotnēm vai pat dažādām aparatūrām.

SPARQL (“SPARQL Protocol and RDF Query Language” angliski) – RDF vaicājumu valoda, kas ir bāzēta uz SQL vaicājumu valodu, izmantota, lai izgūtu informāciju no RDF datu avotiem.

RIF (“Requirements Interchange Format”) – XML failu formāts, izmantots, lai apmainītos ar prasībām saistītu informāciju starp dažādām lietotnēm un sistēmām.

JSON (“JavaScript Object Notation”) – datu apmaiņas formāts, kas izmanto cilvēku lasāmu tekstu, lai saglabātu un apmainītos ar informāciju, kas ir saglabāta īpašību un vērtību pāros.

Lua – augsta līmeņa multi-paradigmu programmēšanas valoda, paredzēta iegultai izmantošanai lietotnēs.

Serializācija – datorikas kontekstā, process, lai pārtulkotu objektu vai datu struktūru formātā, ko var tālāk saglabāt failā, datu buferī vai pārraidīt tīklā.

Izmete – darba kontekstā saprotama kā datubāzes izmete, kur datubāzes tabulu struktūra un dati tiek eksportēti failā.

Turtle (“Terse RDF Triple Language” angliski) – RDF serializācijas formāts, balstīts uz “Notation 3 “ RDF serializācijas formātu.

GitHub – Projektu un lietotņu versiju kontroles repozitoriju uzturēšanas platforma. Kur “Git” versiju kontroles repozitoriji var tikt publiski izplatīti un ir pieejami.

PosgreSQL – Atvērtā pirmkoda relāciju datubāzes sistēma.

Python – Interpretēta augsta līmeņa objekt-orientēta programmēšanas valoda.

MySQL – Atvērtā pirmkoda relāciju datubāzes sistēma.

IRI (“Internationalized Resource Identifier” angliski) – tīmekļa protokolu standarts, kas apraksta tīmeklī pieejamus resursus.

IEVADS

Semantiskais tīmeklis ir daļa no globālā tīmekļa, kas ir lasāms un saprotams datoriem. Un tā kā mūsdienās ir daudz populāru automatizētu asistentu, kuriem ir jāiegūst informācija no tīmekļa, ir svarīgi ka tie spēj atrast un nolasīt nepieciešamos datus.

Tā kā semantiskā tīmekļa lielākās strukturēto datu bāzes, jeb saistīto datu avoti kā Wikidata un DBpedia paliek arvien apjomīgāki un vairāk informācija tiek aprakstīta strukturētā veidā RDF datu modelī. Rodas nepieciešamība pēc tādiem rīkiem kā "ViziQuer" vai "Optique VQs", kas palīdz veidot vizuālus, strukturētus datu analīzes vaicājumus pār RDF datiem. Bet lai šāda veida rīki palīdzētu ar vizuālo vaicājumu veidošanu tiem ir nepieciešamība zināt saistīto datu avotu shēmas, lai ieteiktu kā labāk veidot doto vizuālo vaicājumu.

Bakalaura darba pētāmā problēma ir saprast Wikidata saistīto datu avota shēmu un izstrādāt skriptu, kas izgūtu šī datu avota shēmu.

Bakalaura darba mērķis ir apskatīt un analizēt semantiskā tīmekļa konceptus, RDF datu modeli un semantiskā tīmekļa saistīto datu avotus kā DBpedia un Wikidata. Un izstrādāt skriptu, kas izgūtu Wikidata datu avota shēmu, kuru tad tālāk varētu izmantot RDF vizuālo vaicājumu rīkos kā "ViziQuer" vai "Optique VQs", lai palīdzētu ar vaicājumu priekšā teikšanu, pabeigšanu.

1. SEMANTISKAIS TĪMEKLIS

Semantiskais tīmeklis ir daļa no globālā tīmekļa, ko definē kopums ar standartiem, ko uztur globālā tīmekļa konsorcijs. Semantiskā tīmekļa idejas mērķis ir padarīt tīmeklī pasniegto informāciju dator lasāmu.

Semantiskais tīmeklis nav kā jauna, atsevišķa tehnoloģija, bet drīzāk kā jau esošā tīmekļa papildinājums. Semantiski dati un informācija, kas sniedz dator lasāmu "nozīmi" tiek uzlikta pa virsu cilvēkiem pasniegtajai informācijai.

Lai aprakstītu dažādus terminus un konceptus datoram saprotamā veidā tiek izmantotas tādas tehnoloģijas kā RDF, OWL, XML un citas. Ontoloģija sniedz iespēju aprakstīt dažādus objektus, to aprakstošos atribūtus, tā piederību augstāka līmeņa kopumam, piemēram, ka entītija "Latvija" ir instance no klases "valsts" un tai pieder atribūts "platība", kam vērtība būtu 64589 km^2.

Mums cilvēkiem tas var likties triviāli un pašsaprotami, ka piemēram, Latvija ir valsts, bet priekš datoriem mums ir specifiski jādefinē un jāapraksta šis koncepts.

1.1. Vēsture

Semantiskais tīmeklis kā koncepts radās jau tik agri kā vismaz 1994.gadā, kad vispasaules tīmekļa izgudrotājs un pašreizējais vispasaules tīmekļa konsorcija vadītājs – Tims Bērnerss-Lī izteica šo ideju pirmajā internacionālajā vispasaules tīmekļa sanāksmē, kas norisinājās no 1994.gadā no 25. līdz 27. maijam.

Šī ideja tālāk tika papildus aprakstīta rakstā "The Semantic Web", kas tika publicēts amerikāņu populārzinātniskajā žurnāla "Scientific American" 2001.gada maija izdevumā šī raksta autori bija Tims Bērnerss-Lī, Džeims Hendlers un Ora Lasilla. Tieši šajā rakstā tika nostiprināta šī ideja par dator lasāmu tīmekli, ko var izteikti redzēt jau no raksta virsraksta, ka jauna forma tīmekļa informācijai, kas ir nozīmīga datoriem radīs jaunu iespēju revolūciju("A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities"). [2]

Semantiskā tīmekļa ideja tika tālāk papildināta zinātniskā rakstā "The Semantic Web Revisited", kas tika publicēts "IEEE Intelligent Systems" zinātniskā žurnāla 2006.gada janvāra-februāra izdevumā. Šis raksts papildināja un aprakstīja semantiskā tīmekļa vīziju, šīs vīzijas progresu un tika apskatīti sastaptie izaicinājumi. [3]

Līdz šim semantiskais tīmeklis nav tik plaši un veiksmīgi attīstījies kā pats globālais tīmeklis, lai arī mūsdienu tīmeklī, kas populāri tiek dēvēts kā "Web 2.0" ir daudz semantisku, daļēji

strukturētu datu, piemēram, kā tagi pie publikācijām, video, mūzikas, bildēm. Arī informācija HTML ‘<meta>’ tagos ir semantiska informācija, kuru izmanto tīmekļa meklētāji. Strauji pieaugošā tīmekļa arhitektūra tikai paātrina semantiskā timekļa izaugsmi, jo lielākā daļa lielo tiešsaistes sistēmu piedāvā publiski pieejamus API, kas sniedz iespēju atsevišķi izstrādātām sistēmām sazināties.

1.2. Sastāvdaļas un standarti

Semantiskā tīmekļa standartizācija “Web 3.0” kontekstā ir globālā tīmekļa konsorcija atbildībā. Lai arī bieži ar terminu “Semantiskais tīmeklis” apraksta tajā izmantotās tehnoloģijas, kuras ir standartizējis globālā tīmekļa konsorcijs.

Semantiskajā tīmeklī izmantotās tehnoloģijas ar labi izstrādātiem standartiem:

- XML – datu reprezentācijas forma strukturētā tekstā, paredzēta lai atvieglotu informācijas koplietošanu tīmeklī.
- RDF – datu aprakstīšanas valoda paredzēta, lai aprakstītu tīmekļa bāzētus resursus un to savstarpējās attiecības.
- RDFS – RDF shēma, kas papildina RDF ontoloģiju kā vārdnīca aprakstot RDF bāzētu resursu atribūtus un klases ar papildus semantiku, lai aprakstītu šo klašu un atribūtu hierarhiju.
- SPARQL – protokols un uz SQL bāzēta RDF vaicājumu valoda paredzēta, lai iegūtu dažādu informāciju par semantiskiem tīmekļa datiem.
- RIF – likumu apmaiņas formāts, kas ir paredzēts, lai aprakstītu tīmekļa likumus, kurus datori var izpildīt.
- OWL – tīmekļa ontoloģijas valoda, kas var papildināt ontoloģiju tajā skaitā RDF, lai aprakstītu klases un atribūtus, kā klašu attiecības, kardinalitāti, vienādību utt.

Visas šīs sastāvdaļas kopā tiek izmantotas, lai īstenotu semantisku datu reprezentāciju globālajā tīmeklī. [4]

1.3. Pielietojums

Semantiskais tīmeklis un tā strukturētie dati mūsdienu tehnoloģiju bagātajā vidē tiek plaši izmantoti. Vairāki labi piemēri:

- Saistīto datu avoti kā DBpedia un Wikidata, kas sniedz iespēju programmām piekļūt informācijai par dažādām vietām, cilvēkiem, objektiem un notikumiem, līdzīgi kā mēs izmantojam Wikipedia tikai dator lasāmi.
- Inteliģentie asistenti kā "Cortana", "Siri", "Alexa", kas var iegūt informāciju no tīmekļa un veikt dažādus uzdevumus kā rezervēt lidojumam biļetes. Šie inteliģentie asistenti pamatā izmanto semantiskos datus, lai izpildītu šos uzdevumus.
- Datizrace – semantiskā tīmekļa strukturētie dati atļauj vieglāku datizraci un informācijas iegūšanu automatizētiem botiem un dažādām programmām.
- Datu vizualizācija – Dažādi rīki kā "Mimirix" sniedz iespēju vizuāli apskatīt un apstrādāt RDF datus, kas ir daudz intuitīvāk un vieglāk saprotami, kā arī sniedz iespēju vizualizēt dažādus konceptus, attiecības starp objektiem utt.

Šie protams nav vienīgie semantiskā tīmekļa pielietojumi, bet vieni no spožākajiem.

1.4. Ierobežojumi un nepilnības

Lai arī semantiskajam tīmeklim ir daudz plusu un tas sniedz vairākas iespējas kā dator lasāmu informāciju. Semantiskā tīmekļa konceptam ir vairāki ierobežojumi un nepilnības kā:

- **Nenoteiktība**

  Eksistē vairāki nenoteikti koncepti kā "plats", "šaurs", "vecs", "jauns" utt. Šis rodas no nenoteiktiem lietotāju vaicājumiem, mēģinājumiem apvienot dažādas zināšanu bāzes ar pārklājošiem bet nedaudz atšķirīgiem konceptiem kā arī dēļ lietotāju vaicājumu terminu saskaņošanas ar sistēmas terminiem.

- **Apjoms**

  Vispasaules tīmeklis satur neaptverami lielu daudzumu vietņu un informācijas. Piemēram, Wikidata vien eksistē klases kā "Cilvēks" ar 9 miljoniem instanču, "Zinātnisks raksts" ar 37 miljoniem ierakstu. Un ar semantisko tīmekli strādājošām sistēmām var nākties apstrādāt patiesi milzīgu apjomu ar datiem. Kā arī nevar garantēt, ka vietnēs un informācijas bāzēs nav dublicējušies dati.

- **Nekonsekvence**

  Apvienojot ontoloģijas no dažādiem resursiem vai arī izstrādājot lielāka apjoma ontoloģijas var rasties vairākas loģikas pretrunas. It īpaši slikti ir mēģināt tālāk šādus datus izmantot, lai izvestu vai pierādītu ko, jo šādā gadījumā pretrunas var jebko padarīt patiesu.

- **Informācijas patiesums**

  Līdzīgi kā ar publiski izstrādātām un papildinātām informācijas bāzēm kā Wikipedia nav veida kā garantēt, ka dotā informācija nāk no uzticama avota. Lai šo apietu ir iespējams izsekot identitāti entītijai, kas ir izveidojusi vai sniegusi doto informāciju.

- **Sarežģītība**

  Semantiskā tīmekļa un tajā izmantoto tehnoloģiju koncepti ne katram var likties viegli saprotami un uztverami. Strukturēti dati ir datoram saprotami, bet parastam cilvēkam tie tieši var būt grūtāk uztverami. Lai plaši pielietotu semantiskā tīmekļa idejas industrijā var nākties papildus apmācīt izstrādātājus un iesaistītās personas, kas reizēm var neapmaksāties.

- **Uzturēšana**

  Kā ‘Apjoms’ punktā jau tika minēts, globālais tīmeklis ir masīvs un lai aprakstītu visu informāciju datoriem saprotamā formātā ir apjomīgs darbs. Arī jauni izstrādātām vietnēm datu strukturēta aprakstīšana prasa papildus izstrādes līdzekļu, kas ne vienmēr tiks pilnīgi izdarīts.

- **Jaunas tehnoloģijas**

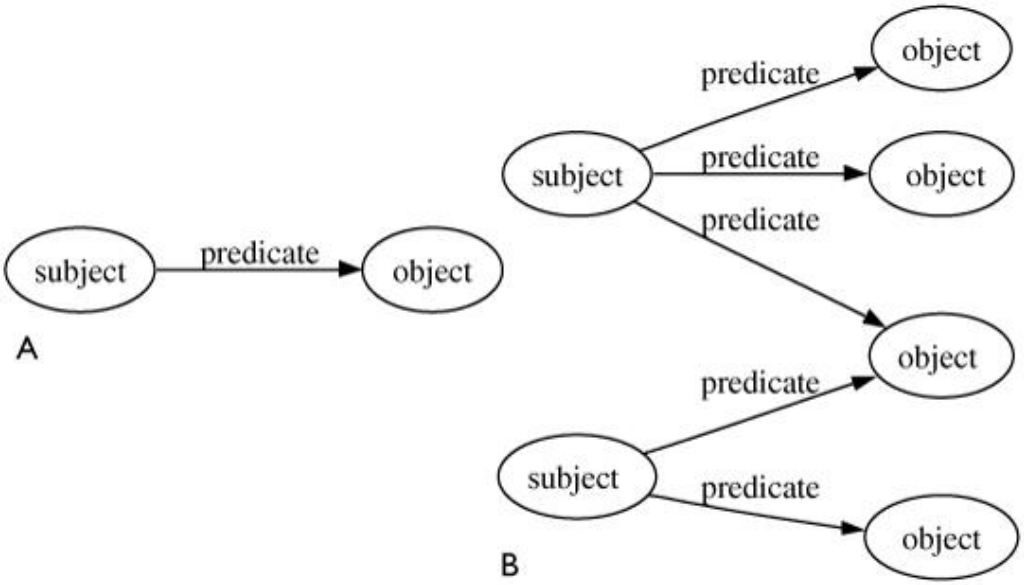
  Mūsdienās parādās vairākas jaunas tehnoloģijas kā mākslīgais intelekts, kas padara semantiskā tīmekļa konceptu absolūtu. Attīstās dabiskās valodas atpazīšanas un saprašanas tehnoloģijas, mākslīgā intelekta algoritmi nestrukturētu datu apstrādei, kas ievāc informāciju no vietnēm, samazinot vajadzību pēc dator lasāmiem strukturētiem datiem.

Pēc autora domām semantiskais tīmeklis ir lielisks un industrijai kritisks koncepts jaunu sistēmu izstrādē un informācijas pieejamības, apmaiņas papildināšanā. Bet autors uzskata, ka dēļ lielās papildus uzturamās galvenes virs pamata informācijas un datiem apvienojumā ar arvien attīstošām mākslīgā intelekta tehnoloģijām var padarīt semantisko tīmekli absolūtu un vairs ne tik noderīgu.

2. RESURSU APRAKSTĪŠANAS IETVARS

Resursu aprakstīšanas ietvars, jeb RDF īsāk ir globālā tīmekļa konsorcija specifikāciju kopums, kas oriģināli bija veidots kā metadatu datu modelis, jeb modelis, kas apraksta datus.

Tā pamata ideja ir aprakstīt resursus, pamatā tīmekļa resursus, izmantojot izteikumus, kas ir formā – subjekts : predikāts : objekts, kur subjekts apzīmē resursu, predikāts apraksta šī resursa īpašības un apzīmē attiecības, atkarības starp subjektu un objektu. Labs piemērs, kas aprakstītu doto ideju ir "Latvijas himna ir Dievs, svētī Latviju!", šajā piemērā subjekts ir "Latvija", predikāts ir "himna ir" un objekts ir "Dievs, svētī Latviju!".



Attēls 2-1 **RDF trijnieki un iespēja tos apvienot grafā**

Augstāk attēlotajā attēlā "A" daļā var izteikti redzēt RDF trijnieku(subjekts, predikāts, objekts), kas ir RDF pamatā. Un "B" daļā var redzēt, ka saliekot kopā vairākus šādus trijniekus var veidot kompleksus grafus, kur viens subjekts var būt saistīts ar vairākiem objektiem, un tajā pašā laikā viens objekts var saistīt vairākus subjektus, praktiski starp subjektu un objektu veidojot n:n attiecību.

Ir noderīgi pieminēt, ka RDF datu modelis ir tikai pamats un to var papildināt izmantojot ontoloģijas valodas kā RDFS(resursu aprakstīšanas ietvara shēma) vai OWL(tīmekļa ontoloģijas valoda), kuras papildina RDF modeli ar klasēm un īpašībām, lai pilnvērtīgāk aprakstītu ontoloģiju.

2.1. Vēsture

Resursu aprakstīšanas ietvara sākums ir meklējams jau 1997.gada 2.oktobrī, kad tika izdota šī modeļa pirmā versija. To izdeva globālā tīmekļa konsorcija grupa, kas iekļāva pārstāvjus no lieliem uzņēmumiem kā "IBM", "Microsoft", "Netscape", "Nokia" un citiem. [5]

Neilgi pēc tā 1999.gada 22.februārī globālā tīmekļa konsorcijs izdeva RDF rekomendācijas dokumentu, kas nozīmē, ka pēc nopietnas tehnoloģijas vai koncepta apskates un pilnveidošanas globālā tīmekļa konsorcijs iesaka doto modeli kā tīmekļa standartu. Šajā rekomendācijā tika aprakstīts RDF datu modelis, kā arī tā XML serializācija. [5]

2004.gada februārī šo rekomendāciju aizstāja kopums ar RDF modeļa specifikācijas dokumentiem, ko izdeva globālā tīmekļa konsorcijs ("RDF Concepts and Abstract", "The RDF Primer", "RDF/XML Syntax Specifications (revised)", "RDF Semantics", "RDF Vocabulary Description Language 1.0", "The RDF Test Cases").

Līdz beidzot 2014.gadā tika izveidota RDF datu modeļa 1.1 versija (1.0 iepriekš), to aprakstīja globālā tīmekļa konsorcija izdotu dokumentu kopums ("RDF 1.1 Concepts and Abstract Syntax", "RDF 1.1 Primer", "RDF 1.1 XML Syntax", "RDF 1.1 Semantics", "RDF Schema 1.1", "RDF 1.1 Test Cases"). [6]

2.2. RDF/RDFS Vārdnīca

Tālāk tiks uzskaitītas un aprakstītas klases, īpašības un atribūti, kas sastāda RDF vārdnīcu, kas ir aprakstītas RDF specifikācijā. [7]

Tālāk uzskaitītajā sarakstā saīsinājumi pirms klasēm, īpašībām un atribūtiem apraksta, kurai shēmai pieder dotais elements. RDF elementi pieder "http://www.w3.org/1999/02/22-rdf-syntax-ns#" interneta resursam, kamēr RDFS elementi pieder "http://www.w3.org/2000/01/rdf-schema#" interneta resursam.

2.2.1. RDF/RDFS Klases

- ***rdfs:Class*** – Apzīmē, ka dotais elements ir klase.
- ***rdfs:Datatype*** – Apzīmē datu tipus.
- ***rdfs:Resource*** – Apraksta resursus.
- ***rdfs:Container*** – Apraksta klases, kas ir konteineri.
- ***rdfs:Literal*** – Resurss, literāļu vērtībām, kas ir teksts vai skaitļi.

- ***rdf:List*** – Resursi, kas apzīmē sarakstu.
- ***rdf:Statement*** – Resursi, kas ir izteikumi.
- ***rdf:Property*** – Resursi, kas ir īpašības.
- ***rdf:Alt*** – Konteineri, kas satur alternatīvus.
- ***rdf:Bag*** – Konteineri, kas satur nesakārtotus elementus.
- ***rdf:Seq*** – Konteineri, kas satur sakārtotus elementus.
- ***rdfs:ContainerMermbershipProperty*** – Konteineru piederības īpašības.
- ***rdf:XMLLiteral*** – XML literāļu vērtības.

2.2.2. RDF/RDFS Īpašības

- ***rdfs:domain*** – Īpašība, kas apraksta klases domēnu.
- ***rdfs:range*** – Īpašība, kas apraksta klases vērtību diapazonu.
- ***rdfs:subPropertyOf*** – Apraksta, ka īpašība ir apakš īpašība citai īpašībai.
- ***rdfs:subClassOf*** – Apraksta, ka klase ir apakš klase citai klasei.
- ***rdfs:comment*** – Literālis, kas cilvēkam saprotami apraksta doto resursu.
- ***rdfs:label*** – Literālis, kas apraksta dotā resursa nosaukumu.
- ***rdfs:isDefinedBy*** – Īpašība, kas apraksta resursu izmantojot citu resursu.
- ***rdfs:seeAlso*** – Īpašība, kas norāda uz resursu, kas satur papildus informāciju par doto resursu
- ***rdfs:member*** – Īpašība, kas izskaidro ka viens resurss pieder citam resursam, visas īpašības, kas apraksta resursu piederību ietvariem ir šīs īpašības apakš īpašība.
- ***rdf:first*** – Īpašība, kas paskaidro, ka resurss ir dotā saraksta pirmais elements.
- ***rdf:rest*** – Īpašība, kas apraksta, ka saraksts ir dotā saraksta atlikušais apakš saraksts.
- ***rdf:subject*** – Īpašība, kas apraksta, ka resursa subjekts ir RDF izteikums.
- ***rdf:predicate*** – Īpašība, kas apraksta ka resursa predikāts ir RDF izteikums.
- ***rdf:object*** – Apraksta, ka resursa objekts ir RDF izteikums.
- ***rdf:value*** – Īpašība izmantota, lai aprakstītu ka resursa vērtība ir kāds cits resurss.
- ***rdf:type*** – Īpašība, kas apraksta resursa tipu, jeb piederību kādai klasei.

2.2.3. RDF Atribūti

- ***rdf:about*** – Apraksta resursu, kurš tiek definēts.

- ***rdf:Description*** – Ietvars resursa aprakstam.
- ***rdf:resource*** – Apraksta resursu, lai identificētu īpašību.
- ***rdf:datatype*** – Norāda elementa datu tipu.
- ***rdf:ID*** – Norāda elementa identifikatoru.
- ***rdf:li*** – Definē sarakstu.
- ***rdf:_n*** – Definē mezglu.
- ***rdf:nodeID*** – Norāda mezgla identifikatoru.
- ***rdf:parseType*** – Apraksta tipu kādā veidā elements ir jāapstrādā.
- ***rdf:RDF*** – RDF dokumenta sakne.
- ***xml:base*** – Norāda XML dokumenta pamatu.
- ***xml:lang*** – Norāda valodu elementa saturam.

2.3. Serializācijas formāti

Atšķirībā no citiem datu modeļiem RDF neierobežo viens serializācijas formāts, kā piemēram, XML, bet saistītos datus RDF modelī, kas sastāv no iepriekš minētajiem trijniekiem ir iespējams serializēt vairākos formātos. Tālāk apakšnodaļās tiks uzskaitīt un aprakstīti dažādi formāti, kuros ir iespējams serializēt RDF informāciju. Šie uzskaitītie formāti nav vienīgi, bet tikai daļa, kas ir atlasīti, lai atspoguļoto šo formātu daudzveidību. [8]

2.3.1. RDF/XML

1999.gadā globālā tīmekļa konsorcija izdotajā RDF rekomendācijā, tika definēta RDF serializācija XML formātā. Lai arī šis ir vecākais no serializācijas formātiem, tas iespējams nav pats piemērotākais, jo RDF un XML ir fundamentāli atšķirīgi koncepti, kur XML ir vairāk koka tipa dokuments, kamēr RDF ir uz trijniekiem bāzēts grafs. Izstrādātājiem, kas ir pieredzējuši ar XML šis formāts var izskatīties pazīstams, bet tādēļ ka tas pilnīgi neatspoguļo RDF trijnieku modeli, var izraisīt apjukumu. Ieteicams izmantot, ja ir nepieciešamība strādāt ar XML.

```xml
<?xml version="1.0"?>

<RDF>
  <Description about="https://www.w3schools.com/rdf">
    <author>Jan Egil Refsnes</author>
    <homepage>https://www.w3schools.com</homepage>
  </Description>
</RDF>
```

Attēls 2-2 **RDF/XML serializācijas formāta piemērs**

Augstāk redzamajā attēlā ir redzams šī serializācijas formāta piemērs, kur kā RDF elementa pamatā ir aprakstīta mājaslapa un norāde uz to un tās autors.

2.3.2. RDFa

RDFa praktiski ir RDF informācija iekš HTML dokumenta. Šādā veidā sniedzot iespēju papildināt tīmekļa vietņu saturu ar papildus semantisku kontekstu. Google spēj apstrādāt informāciju RDFa formātā, lai uzlabotu meklēšanas rezultātus, lai arī pats Google iesaka izmantot JSON-LD formātu. Pat globālā tīmekļa konsorcijs beidza atbalstīt šo formātu, jo tas tika izmantots ļoti maz un nepieciešamais kods, lai to lasītu bija salīdzinoši sarežģīts un arī Google nespēja to apstrādāt pilnīgi pareizi.

Šis formāts ir tikai ieteicams izmantot, lai papildinātu mājaslapas ar semantisku informāciju.

```xml
<div xmlns:dc="http://purl.org/dc/elements/1.1/"
  about="http://www.example.com/books/wikinomics">
  <span property="dc:title">Wikinomics</span>
  <span property="dc:creator">Don Tapscott</span>
  <span property="dc:date">2006-10-01</span>
</div>
```

Attēls 2-3 **RDFa serializācijas formāta piemērs**

Augstāk ir redzams šī formāta piemērs, kur HTML elementā, kas satur informāciju par grāmatu ir pievienoti RDF atribūti, lai padarītu šo informāciju semantiski draudzīgāku.

2.3.3. Notation3 (.n3)

Tims Bērnerss-Lī nebija apmierināts ar RDF/XML serializācijas formātu un vēlējās izveidot labāku formātu RDF serializācijai. Un šis ir Tima izveidotais formāts. Šis formāts ļoti atgādina RDF trijnieku modeli, tādēļ varētu būt viegli saprotams, ja ir pazīstams RDF. Šis formāts pielieto prefiksus("@prefix"), kas padara šo formātu salīdzinoši kompaktu. Šī formāta mīnuss ir iespējami sarežģīta RDF datu serializācija.

```
@prefix dc: <http://purl.org/dc/elements/1.1/>.

<https://en.wikipedia.org/wiki/Tony_Benn>
  dc:title "Tony Benn";
  dc:publisher "Wikipedia
```

Attēls 2-4 **N3 serializācijas formāta piemērs**

Formāta piemērā var redzēt, ka šis serializācijas formāts ir salīdzinoši kompakts un viegli lasāms. Prefiksi var nedaudz apgrūtināt lasāmība, bet tie ļoti samazina apjomu.

2.3.4. JSON-LD (.jsonld)

JSON-LD ir serializācijas formāts, lai attēlotu RDF informāciju JSON formātā. JSON-LD ir tikai kā JSON paplašinājums un tā dokumenti ir arī derīgi JSON dokumenti. Šis formāts ir viegli lasāms, bet tas ir salīdzinoši grūti apstrādājams. Ja informācija ir nepieciešama RDF formātā var nākties veikt papildus vaicājumus, lai pilnvērtīgi iegūtu visu informāciju. Šis formāts ir piemērots, ja nepieciešamie dati ir vajadzīgi JSON formātā.

```
{
  "@context": "https://json-ld.org/contexts/person.jsonld",
  "@id": "http://dbpedia.org/resource/John_Lennon",
  "name": "John Lennon",
  "born": "1940-10-09",
}
```

Attēls 2-5 **JSON-LD serializācijas formāta piemērs**

Augstāk redzamajā attēlā var redzēt šī formāta piemēru. JSON dokumentus šajā formātā var atpazīt pēc "@context" atribūta, kas tiek izmantots, lai pamata atslēgas pārvēstu saitēs uz RDF klasēm un īpašībām.

2.3.5. Turtle

Turtle, jeb "Terse RDF Triple Language" ir viens no populārākajiem RDF serializācijas formātiem. Turtle ir apakš sets Notation3 serializācijas formātam, kurš ir padarīts vienkāršāks, zaudējot papildus sintakses struktūras. Šis serializācijas formāts ir ļoti cilvēk lasāms, noderīgs, ja RDF informācija ir manuāli jārediģē. Tomēr, tas līdzīgi kā Notation3 ir sarežģīti apstrādājams, bet dēļ tā popularitātes eksistē vairākas bibliotēkas šī formāta atbalstam.

```
@prefix relPrefix: <http://www.perceive.net/schemas/relationship/> .

<http://example.org/#green-goblin> relPrefix:enemyOf <http://example.org/#spiderman> .
```

Attēls 2-6 **Turtle serializācijas formāta piemērs**

Augstāk redzamajā formāta piemērā ir aprakstīts pamata RDF trijnieks, ka "Green Goblin" ir "Spiderman" ienaidnieks. Šis varbūt nav tas labākais piemērs, jo tā kā Turtle ir apakš sets Notation3, šis piemērs ir arī derīgs Notation3 formātā.

3. SAISTĪTO DATU AVOTI

Saistīto datu avoti ir svarīga semantiskā tīmekļa sastāvdaļa. Šie datu avoti ir liela apjoma saistīto datu apkopojumi, kas satur visdažādāko informāciju dator lasāmā formātā. Piemēram, Wikidata un DBpedia gadījumos šie datu avoti satur pamata vispārējas zināšanas, līdzīgi kā Wikipedia, sākot ar datiem par personām, valstīm, vēsturiskiem notikumiem, zinātniskiem konceptiem utt. Bet protams, eksistē arī dažādi citi saistīto datu avoti, kā piemēram:

- "**Geonames**", kas satur ģeogrāfisku informāciju par vairāk kā 7,5 miljoniem vienumiem, kas ietver valstis, pilsētas, pasta indeksus un tam līdzīgi.
- **Globālās izpētes atpazīšanas datubāze**("Global Research Identifier Database (GRID)"), kas satur informāciju par brīvi pieejamām izglītības un izpētes organizāciju datubāzēm. Šī datubāze pašlaik satur informāciju par nu jau vairāk kā 100 tūkstošiem izpētes un izglītības institūtiem.
- **"DBTune"**, daļa no "atvērto publisko datu sasaistes semantiskajā tīmeklī" projekta. Satur strukturētus RDF datus par mūziku saistītu informāciju, kā informāciju par skaņdarbiem, akordiem, skaņu signāliem, albumiem un tam līdzīgi.

Saistīto datu avoti parasti arī piedāvā interfeisu caur kuru sazināties ar datu avotu un caur kuru var izgūt nepieciešamo informāciju, parasti šim tiek izmantots SPARQL, kas ir SQL līdzīga valoda, kas strādā ar datiem RDF formātā.

Lielākā daļa no šiem datu avotiem informāciju satur RDF vai RDF līdzīgā formātā. Daudzi no šiem datu avotiem, tai skaitā arī Wikidata un DBpedia satur atvērtos saistītos datus("Linked open data"), kur saistītie dati ir izlaisti zem atvērtās licences un tie ietver norādes uz citiem saistītajiem datiem. Pēc Tima Bērnerss-Lī domām skalā, kas novērtē globālajā tīmeklī pieejamos atvērtos datus, atvērtie saistītie dati ir visaugstākās kvalitātes [9].

Tālāk šajā nodaļā tiks apskatīti globālā tīmekļa divi lielākie atvērto saistīto datu avoti, tas ir DBpedia un Wikidata.

3.1. DBpedia

DBpedia ir projekts, kura mērķis ir no izvilkt strukturētu, semantisku informāciju par datiem no Wikipedia. DBpedia ir licencēts zem GNU Vispārējās publiskās licences ("GNU General Public Licence" angliski, jeb GPL), padarot šajā datu resursā esošo informāciju publiski pieejamu globālajā tīmeklī.

DBpedia izgūst strukturētu informāciju no Wikipedia rakstiem no to “infoboxes”. Wikipedia regulāri, apmēram, ikmēnesi publicē SQL izmeti (pieejama http://dumps.wikimedia.org/enwiki/ vietnē), publicējot informāciju par jauniem un papildinātiem rakstiem un DBpedia izmanto šos datus, lai papildinātu savu saistīto datu bāzi. [10]

Informāciju no DBpedia var izgūt caur to SPARQL servisu vietnē - https://dbpedia.org/sparql, bet ir iespējams arī izveidot savienojumu no programmas izmantojot DBpedia piedāvāto API. DBpedia arī piedāvā nolādēt RDF izmetes un izgūt datus no tām.

3.1.1. Vēsture

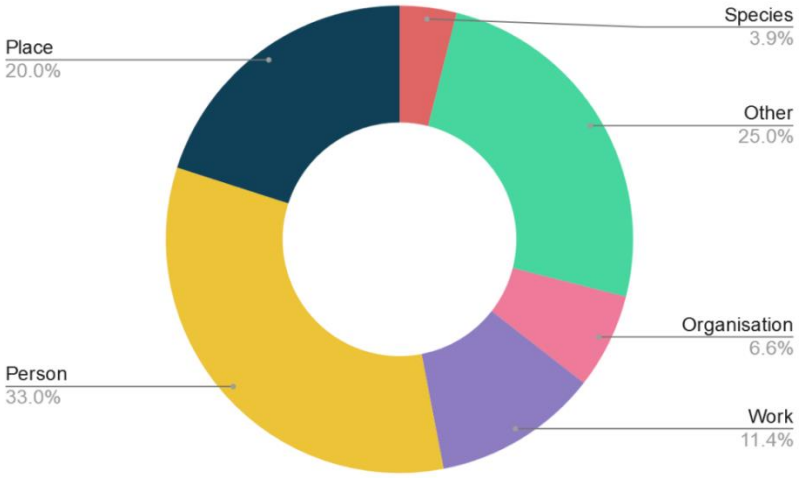
Projekts tika aizsākts ap 2007.gadu, un pie tā darbojās cilvēki no Leipcigas Universitātes un Berlīnes Brīvās universitātes. Pirmais publiski pieejamais datu kopums tika publicēts 2007.gada 10.janvārī un tas saturēja aptuveni 103 miljonus RDF trijnieku. [11]

Pēdējā stabilā DBpedia izdotā versija ir 2016-10 versija, kas tika izdota 2017.gada 4.jūlijā un tā saturēju nu jau līdz pat 9,5 miljardiem RDF trijnieku. Mūsdienās projektu uztur cilvēki no Leipcigas Universitātes un Manheimas Universitātes, bet šis projekts ir publiski izstrādāts un pie tā strādā daudz citi brīvprātīgie un tas tiek regulāri atjaunots un papildināts.

3.1.2. Shēma un ontoloģija

DBpedia izmanto Resursu Aprakstīšanas Ietvaru(RFD), lai aprakstītu to saturošos datus. DBpedia ontoloģija iekļauj vairāk nekā 768 klases un 3000 dažādas īpašības, kas apraksta šīs klases. DBpedia shēmu un tās ontoloģiju var izgūt caur DBpedia piedāvāto SPARQL servisu, vai arī ir pieejami ar ontoloģiju saistītās informācijas izmetumi, kurus var lejupielādēt. [12]

DBpedia lielākās klases ir “Vietas”, “Cilvēki”, ”Darbi”, “Organizācijas” un “Sugas”, kur vietu un cilvēku klases pārsniedz pat miljonu instanču. Zemāk redzamajā diagrammām var skaidri redzēt instanču daudzumu attiecībā pret šīm klasēm. [12]



Attēls 3-1 **DBpedia instanču daudzums pa klasēm**

DBpedia iesaka dažādus interesantus rīkus ar kuriem var izpētīt DBpedia ontoloģiju. Piemēram, ir pieejams interaktīvs klašu hierarhiju un to instanču daudzuma apskates rīks (pieejams http://78.46.100.7:9000 vietnē)

3.1.3. Pielietojums

DBpedia pats galvenais pielietojums ir kā datu avots, kas ir dator lasāms. Piemēram, ja izstrādājat lietotni un ir nepieciešamība iegūt datus par kādu objektu vai valstīm, cilvēkiem var lieliski izmantot DBpedia SPARQL servisu vai arī izveidot savienojumu uz to piedāvāto API un izgūt nepieciešamo informāciju. Piemēram, lietotne, kas iegūtu un attēlotu informāciju par lielākajām Latvijas pilsētām varētu izmantot zemāk redzamo vaicājumu, lai iegūtu šo informāciju.

Ja nav pieredzes ar SPARQL dotais vaicājums varētu likties nedaudz grūti saprotams, bet

```
SELECT DISTINCT ?Pilseta ?Populacija WHERE {
    ?Pilseta a dbo:City.
    ?Pilseta dbo:country dbr:Latvia.
    ?Pilseta dbo:populationTotal ?Populacija.
}
ORDER BY DESC(?Populacija)
LIMIT 10
```

Attēls 3-2 **DBpedia SPARQL vaicājuma piemērs**

praktiski dotajā vaicājumā tiek atlasītas unikālas pilsētas un to populācijas. Tiek atlasītas entītijas, kuras pieder pilsētu klasei(“dbo:City”) un kurām ir atribūts valsts(“dbo:country”) ar vērtību

“dbr:Latvia”. Papildus šīm pilsētām no kopējās populācijas atribūta (“dbo:populationTotal”) tiek paņemts populācijas daudzums. Līdz beidzot entītijas tiek sakārtotas pēc populācijas daudzuma un tiek paņemti top 10 rezultāti. Pašlaik, lai arī vaicājums atgrieza tikai 8 vērtības ar Rīgu, Daugavpili un Liepāju vadībā, jo DBpedia nav datu par citām Latvijas pilsētām.

Līdzīgi daudz citas sistēmas izmanto DBpedia datu izguvei, piemēram, vairākas žurnālistikas organizācijas kā “BBC”, “The New York Times” izmanto DBpedia atvērto saistīto datu projektos un lai organizētu sevis piedāvāto saturu.

DBpedia ir arī lielisks datu avots mākslīgā intelekta sistēmām, jo piemēram, IBM izstrādātā “Watson” sistēma, kas spēj atbildēt uz jautājumiem uzdotiem naturālā cilvēku valodā, izmanto DBpedia kā vienu no tā informācijas avotiem.

Neskaitot uzskaitītās sistēmas eksistē daudz citas sistēmas un lietotnes, kas izmanto informāciju no DBpedia.

3.2. Wikidata

Wikidata ir apjomīgākais globālā tīmekļa atvērto saistīto datu avots, kuru uztur “Wikimedia” fonds. Wikidata ir centralizēta datu noliktava, kurai var brīvi piekļūt citas sistēmas, tajā skaitā citi “wikis”, kurus uztur “Wikimedia” fonds, starp kuriem ir arī visplašāk pazīstamais tīmekļa datu avots – Wikipedia. Wikidata ideja ir glabāt dažādus datus, kurus var izmantot citi “wikis”, lai tie šo informāciju var neglabāt paši pie sevis, bet dinamiski ielādēt no Wikidata. Wikidata satur centralizētu informāciju par statistiku, datumiem, vietām un tam līdzīgi. [13]

Wikidata ir publiski papildināts projekts un informāciju tajā var ievietot ikviens, līdzīgi kā to var darīt Wikipedia. Lai arī lielu daļu no datiem papildina automatizētas sistēmas, kas masveidā izgūst datus no citām datu bāzēm. Wikidata ir labi atzīts un plaši izmantots citās “wikis”, jo piemēram, angliskā Wikipedia(“enwiki”) 5,5 miljoni lapu satur informāciju no Wikidata. [16]

3.2.1. Vēsture

Wikidata projekts tika izlaists 2012.gada 29.oktobrī. Sākotnējo projekta izstrādi finansēja apjomīgi ziedojumi no “Allen Institure for Artificial Intelligence”, “Gordon and Betty Moore” fonda un “Google”.

No paša sākuma vienīgā pieejamā funkcionalitāte bija valodas saišu centralizācija, kas atļāva ierakstiem pievienot pamata informāciju kā nosaukumu, aprakstu un alternatīvos nosaukumus. Vēlāk 2013.gadā projekts tika papildināts ar funkcionalitāti, lai ierakstus varētu

papildinātu ar īpašībām. 2015.gada 8.septembrī "Wikimedia" fonds papildināja Wikidata projektu ar SPARQL vaicājumu servisu, sniedzot iespēju veikt SPARQL vaicājumus pret Wikidata datiem. [14]

Šī darba rakstīšanas laikā, 2021.gadā Wikidata satur informāciju par vairāk nekā 93 miljoniem ierakstu.

3.2.2. Shēma un ontoloģija

Wikidata tieši neizmanto RDF datu modeli, bet Wikidata datu modelis vienkāršākajā līmenī ir labi atbilstošs RDF datu modelim. Līdzīgi kā RDF, Wikidata attēlo informāciju strukturētos trijniekos – priekšmets ("rdf:subject"), īpašība ("rdf:predicate") un vērtība, kas apraksta doto īpašību ("rdf:object"). Liela daļa RDF datu modeļa un RDFS shēmas īpašībām ir atbilstošas īpašības Wikidata datu modelī, kā piemēram:

- "rdf:type" īpašībai, kas apraksta, ka subjekts ir instance no kādas klases Wikidata ir atbilstoša īpašība "instance of (P31)".
- "rdfs:subClassOf" īpašibai, kas apraksta, ka subjekts ir apakšklase dotai klasei, atbilst Wikidata īpašība "subclass of (P279)".
- "rdfs:subPropertyOf" īpašībai, kas apraksta, ka subjekts ir apakš īpašība dotai īpašībai Wikidata gadījumā atbilst "subproperty of (P1647)" īpašība
- "rdfs:member" īpašībai, kas apzīmē, ka objekts ir daļa no subjekta resursa atbilst Wikidata īpašība "part of (P361)"

Wikidata projektā entītijas ir unikāli apzīmētas, kā piemēram "Q5" apzīmē cilvēka entītiju vai "Q729" apzīmē dzīvnieka entītiju un šīm entītijām ir "wd" predikāts, kas atbilst "http://www.wikidata.org/entity/" saitei. Wikidata gadījumā klases nav eksplicēti definētas, bet tās praktiski ir entītijas kuras apkopo grupu ar entītijām. Entītija ir klase, ja tai tiešā vai netiešā veidā ir vismaz viena instance, jeb entītija ir saistīta ar citu caur "intance of (P31)" īpašību. Klases var veidot vairāku līmeņu hierarhiju caur "subclass of (P279)" īpašību, tad šādā gadījumā zemākā līmeņa klašu instances ir instances arī augstākā līmeņa klasēm, lai arī ne tieši saistīti. Lai arī ir vairākas problēmas ar Wikidata klasēm, kā piemēram, vairākās vietās "instance of (P31)" un "subclass of (P279)" ir nepareizi pielietots, tādā veidā radot nepareizu iespaidu par klasēm. Papildus ir arī grūti viegli pateikt vai entītija ir klase, jo lai atlasītu visas klases vajag atlasīt visus subjektus, kuriem ir vismaz viena instance. [15]

Īpašības Wikidata ir salīdzinoši viegli saprotamas, tās apraksta datu vērtību izteikumā, labi īpašības piemēri ir "krāsa", "garums", "lokācija" utt.. Īpašības kopā ar to vērtību, jeb objektu veido izteikumu. Īpašības pašas ir aprakstītas Wikidata atsevišķās lapās un tās var būt saistītas ar citiem elementiem, tajā skaitā ar citām īpašībām caur "subproperty of (P1647)" īpašību, veidojot hierarhisku īpašību struktūru. Līdzīgi kā klasēm arī katrai Wikidata īpašībai ir unikāls identifikators, kā piemēram "autors ir" īpašībai atbilst "P50" identifikators. Īpašības ir vairāku veidu un tās apraksta dažādi predikāti kā "ps:", kas apraksta īpašības, kas saista vērtības ar izteikumiem, "pq:", kas paraksta īpašības, kas saista ierobežojumus ar izteikumiem. Bet visplašāk izmantotais un parastākais īpašības veids, kas apraksta datus piesaistot vērtību ir "wdt:", kas atbilst "http://www.wikidata.org/prop/direct/" saitei.

Wikidata ontoloģijā darba rakstīšanas laikā eksistē nedaudz virs 80 tūkstošiem klašu un 40 tūkstošiem īpašību. Kas ir daudz salīdzinājumā ar DBpedia klašu un īpašību skaitu.

3.2.3. Datu izguve

Eksistē vairākas metodes, kā iegūt datus no Wikidata :

- SPARQL vaicājumu serviss (pieejams https://query.wikidata.org vietnē). Ir iespējams iesūtīt SPARQL vaicājumus pret Wikidata datiem un iegūt atbildi visdažādākajos formātos (XML, JSON, HTML, Turtle). Uz šo izejas punktu ir iespējams izveidot savienojumu no lietotnes vai skripta caur "GET" vai "POST" vaicājumiem. [17]
- Ir iespējams izgūt datus arī no Wikidata datubāzes izmetēm https://dumps.wikimedia.org/wikidatawiki/ vietnē. Lai arī šīs izmetes ir milzīga apjoma un tās var būt grūti apstrādāt, jo piemēram jaunākā RDF izmete par entītijām ir tuvu 100GB liela.
- Trešā datu izguve attiecas uz "wikis", jo ir iespējams izgūt datus no Wikidata, lai dinamiski papildinātu informāciju citos informācijas avotos, izmantojot apstrādes funkcijas vai Lua(Vairākparadigmu programmēšanas valoda) skriptus. [18]

3.2.4. Pielietojums

Wikidata, kā jau iepriekš minēts, pamatā tiek izmantots citos "wikis" kā datu avots. Bet Wikidata ir arī publiski pieejams un izmantojams caur SPARQL vaicājumu servisu. Piemēram, izstrādājot skriptu var iegūt nepieciešamo informāciju no Wikidata caur SPARQL vaicājumu, ko var iesūtīt kā "POST" vaicājumu:

```
SELECT DISTINCT ?dziesmaLabel ?IzpilditajsLabel ?ZanrsLabel ?ValodaLabel WHERE {
    ?dziesma wdt:P31 wd:Q134556. #P31 - instance of, Q134556 - single
    ?dziesma wdt:P1344 wd:Q50729731. #P1344 - participant in, Q50729731 - Eurovision Song Contest 2021
    ?dziesma wdt:P1476 ?Dziesma. #P1476 - title
    ?dziesma wdt:P136 ?Zanrs. #P136 - genre
    ?dziesma wdt:P175 ?Izpilditajs. #P175 - performer
    ?dziesma wdt:P407 ?Valoda. #P407 - language of work or name
    SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }.
}
```

Attēls 3-3 **Wikidata SPARQL vaicājuma piemērs**

Dotajā vaicājumā var spilgti redzēt vienu no Wikidata shēmas nepilnībām, jo īpašības un entītijas tiek referencētas pēc to identifikatoriem, pēc kuriem uzreiz nevar pateikt, ko tas nozīmē. Tiešsaites Wikidata SPARQL vaicājumu rīkā uzbraucot uz šiem identifikatoriem parādās to nosaukums, bet atsevišķā skriptā vaicājumu nākas komentēt, lai tas būtu lasāmāks.

Augstāk redzamajā vaicājumā tiek atlasītas visas dziesmas, kas piedalās 2021.gada Eirovīzijā. Papildus dziesmām tiek atlasīti to izpildītāji, žanri, kā arī valoda kurā dziesma izpildīta. Vēl viena Wikidata neērtība ir, ka pamatā, ja vaicājumu rīks atgriež entītiju vai īpašību, tas atgriezīs vietni uz to IRI. Tādēļ papildus nākas izmantot īpašu Wikidata nosaukumu servisu("SERVICE wikibase:label"), lai iegūtu šo vērtību nosaukumus.

Wikidata ir pieejami daudz un dažādi rīki, gan datu rediģēšanai, vaicājumu veidošanai, datu vizualizācijai un daudz dažādas vizuālas informācijas attēlošanai. Wikidata rīku tīmekļa vietnē (https://www.wikidata.org/wiki/Wikidata:Tools tiešsaistē) ir saites uz šiem dažādajiem rīkiem, kuru skaits darba rakstīšanas laikā ir vairākos desmitos. Tālāk tiks aprakstīti un apskatīti vairāki interesantākie no šiem rīkiem:

- **Label Collector** (https://www.wikidata.org/wiki/User:YMS/LC tiešsaistē) - ļoti interesants rīks, kas palīdz ar Wikidata lapu rediģēšanu, piedāvājot daļēji automatizētu nosaukumu, aprakstu un aizstājvārdu iegūšanu priekš specifiska temata, no citiem Wikimedia projektiem, kā Wikipedia.
- **Platypus** (https://askplatyp.us/ tiešsaistē) Ļoti interesants rīks, kurā var uzdot daudz un dažādus pamata zināšanu vai ar matemātiku saistītus jautājumus un iegūt atbildes. Šis rīks analizē jautājumus un interpretē to nozīmi, lai sniegtu atbildi. Piemēram, uzdodot jautājumu par pašreizējo Amerikas prezidentu - "Who is the president of USA?", rīks atgrieza atbildi "Joe Biden", kas ir pat pareizi.

- **Crotos** (http://zone47.com/crotos/ tiešsaistē) Mākslas darbu meklēšanas un attēlošanas rīks, kas kā pamatu izmanto informāciju no Wikidata un Wikimedia. Var meklēt mākslas darbus pēc nosaukuma, autora, gada un ir iespējams iegūtos mākslas darbus filtrēt pēc to tipa, laika perioda. Ļoti noderīgs rīks mākslas entuziastiem.
- **Histropedia** (http://histropedia.com/timeline tiešsaistē) Rīks, kas lietotājam sniedz iespēju veidot, vai apskatīt citu veidotās laika joslas. Šajās laika joslās var ievietot daudz un dažādus vēsturiskus notikumus, cilvēkus, impērijas, mūziku, filmas utt. Aizraujošs rīks, kurš var noderēt, ja ir nepieciešams prezentēt notikumu vēsturisko gājumu vai arī vienkārši interesē vizuāli redzēt dažādas laika joslas.
- **Wikidata periodic table** (https://ptable.toolforge.org tiešsaistē) Ķīmiķiem noderīgs rīks, kas attēlo visus Wikidata aprakstītos ķīmijas elementus. Katram elementam ir attēlots to simbols, kā arī to atomiskā masa un norāde uz elementa Wikidata ierakstu.

Šie uzskaitītie rīki nav vienīgie Wikidata pieejamie rīki, bet tikai ļoti neliela daļa no visiem, kuri darba autoram likās interesantākie.

4. DATU SHĒMAS IZGUVE NO WIKIDATA

Šī izgūstamā Wikidata shēma ir kritiska kā atbalsts rīkiem, kas palīdz gala lietotājam vizuālu RDF vaicājumu veidošanā, sakot priekšā dažādu informāciju vaicājuma veidošanas procesā. Eksistē vairāki šādi rīki, kas izmantotu Wikidata shēmu, tai skaitā vizuālo vaicājumu sistēmas kā "Optique VQs" un "Viziquer", kā arī aspektu("facet") bāzētas un naturālas valodas bāzētas sistēmas.[19]

Tālāk šajā nodaļā tiks aprakstīta skripta izstrāde, izvēlētie risinājumi, izmantotie vaicājumi un to paskaidrojumi. Nodaļā tiks iekļauti un aprakstīti vairāki koda gabali, bet pilnais pirmkods ir pieejams GitHub repozitorijā (https://github.com/vehiginters/wikidata_export tiešsaistē).

4.1. Skripta rīku un valodu izvēle

Dotais skripts tika izvēlēts izstrādāt kā Python skriptu. Jo Python ir augsta līmeņa programmēšanas valoda un atļauj salīdzinoši vienkārši aprakstīt sarežģītas kodu struktūras. Papildus paša skripta veiktspēja nebija pati svarīgākā, jo lielāko daļu laika skripts tā vai tā pavadīs gaidot uz vaicājumu izpildi iegūstot datus Wikidata vaicājumu servisā, kā arī kamēr ievietojot iegūto informāciju mērķa PostgreSQL datubāzē.

Kā pagājušajā rindkopā jau pieminēta, izvēlētā mērķa datubāze ir PosgreSQL relāciju datubāze. Tika izvēlētā tieši šī datubāzes sistēma, jo tā ir bezmaksas atvērtā pirmkoda datubāze un bija viegli pieejama, bet doto skriptu būtu iespējams samērā viegli pielāgot arī cita veida relāciju datubāzes sistēmai kā MySQL vai Oracle.

Ir vērts pieminēt, ka tika izvēlēta tieši relāciju tipa datubāze, nevis RDF tipa datubāze dēļ autora mazās pieredzes ar RDF datubāzēm un vairāku gadu pieredzes ar relāciju datubāzēm.

4.2. API savienojumu izveide

Skriptam nākas izveidot savienojumus gan ar ienākošo datu avotu, jeb Wikidata SPARQL vaicājumu servisu, gan arī ar izejošo datu avotu, jeb mērķa PostgreSQL dabubāzi.

4.2.1. Mērķa datubāzes savienojums

Lai izveidotu savienojumu ar PostgreSQL datubāzi un tālāk arī izpildītu vaicājumus uz to, tiek izmantots "psycopg2" Python bibliotēka, kas sanāk ir vienīgā šī skripta neiebūvētās bibliotēkas atkarība. Lai iegūtu savienojumu tiek izmantota dotā funkcija:

```
DB_CON = None
def getDbCon(params):
    # Get a connection to sparSql database using given parameters
    global DB_CON
    if DB_CON is None:
        try:
            print('Connecting to the PostgreSQL database...')
            DB_CON = psycopg2.connect(**params)
        except (Exception, psycopg2.DatabaseError) as error:
            raise Exception("Failed to connect to PostgreSQL database - {}".format(error))
    return DB_CON
```

Attēls 4-1 **Datubāzes savienojuma iegūšanas funkcija**

Funkcija izmanto globālu datubāzes savienojuma mainīgo, kurš tiek izmantots, lai pateiktu vai iepriekš nav jau izveidots savienojums un to nedarītu atkārtoti. Funkcija izmanto no konfigurācijas faila iegūtus datubāzes savienojuma parametrus.

4.2.2. Wikidata vaicājumu servisa savienojums, kļūdas kodu apstrāde

Kamēr savienojums ar Wikidata vaicājumu servisu tiek veikts caur Python iebūvēto bibliotēku "requests", lai nosūtītu "POST" vaicājumus uz https://query.wikidata.org/sparql saiti. Atbilde tiek saņemta JSON formātā un pārveidota Python vārdnīcā, ko tālāk var apstrādāt. No vārdnīcas izmantošanas nāk pluss, ka pie lieliem datu apjomiem ir viegli piekļūt noteiktiem elementiem.

Sūtot pieprasījumu uz Wikidata, pieprasījuma galvenā tiek norādīts, lai saņemtu atbildi JSON formātā, kā arī tiek norādīts "User-Agent" parametrs, kas identificē doto automatizēto skriptu. Wikidata API izmantošanas pamācībā tiek stingri piekodināts automatizētiem skriptiem sevi identificēt, savādāk pastāv risks, ka dotā IP adrese var nonākt Wikidata SPARQL servisa melnajā sarakstā. [17]

Pie šī savienojuma ir viena no skripta vājajām vietām, jeb nepilnībām, ka no datu avota vaicājumi var saņemt atbildē līdz pat 1 miljonam ierakstu un tie nāk vienā kopumā, ko nevar atdalīt. Un tad ievietojot šos datus Python vārdnīcā sanāk aizņemt, lai arī uz neilgu brīdi, līdz pat pāri 1GB brīvpiekļuves atmiņas, bet pēc autora domām dēļ mūsdienu sistēmu brīvpiekļuves atmiņu apjoma šī nav pārāk liela problēma, plus datus ir vieglāk apstrādāt vienā gabalā iekš skripta, nevis sarežģīti mēģināt iegūt tos pa daļām.

Saņemot atbildi no Wikidata SPARQL servisa tiek apstrādāts saņemtais atbildes statusa kods un pēc tā skripts veic attiecīgas darbības:

- 429 atbildes kods vai 503 atbildes kods, ka pēdējā minūtē ir pārsniegts vaicājumu limits (30 minūtē) vai arī, ka serviss ir pārslogots – Abos gadījumos galvenē vajag parādīties

"Retry-After" parametram, kas pasaka pēc cik ilga laika var mēģināt atkal, ko skripts tad pagaida un mēģina atkal.

- 502 atbildes statusa kods, ka ir neizdevies savienojums ar mērķa servisu – Šajā gadījumā skripts vienkārši pagaida neilgu laiku un mēģina atkal
- 500 atbildes statusa kods, ka vaicājums pārsniedza vaicājuma laika ierobežojumu(60 sekundes) – Šajā gadījumā skripts neko daudz nespēj darīt, tādēļ vienkārši piefiksē neizdevušos vaicājumu un turpina.

4.3. Shēmas izguves plāns

Skripta pamata izguves plāns ir:

1. Izgūt visas īpašības, izskaitīt cik attiecībās dotās īpašības ir izmantotas un iegūt šo īpašību nosaukumus
2. Ievietot iegūtās īpašības mērķa datubāzē
3. Atsevišķi izgūt visas klases (entītijas ar vismaz vienu instanci), izskaitīt visu šo klašu instanču un apakš klašu skaitu, kā arī iegūt šīm klasēm to nosaukumus
4. Ievietot iegūtās īpašības mērķa datubāzē
5. Iegūt ienākošās un izejošās klašu un īpašību atkarības, kur īpašība ir ieejoša vai izejoša dotai klasei. Atkarībām tiek izskaitīts cik gadījumos dotā īpašība ir saistīta ar kādu no klases instanci. Papildus izejošām atkarībām tiek izskaitīts, cik klases instanču gadījumos trijnieka objekts ir entītija(objekts ar IRI) un ne literālis. Šis solis ir atkarīgs no 3. jo tas izmanto klašu sarakstu.
6. Ievietot iegūtās klašu un īpašību atkarības mērķa datubāzē
7. Iegūt klašu-klašu atkarības, praktiski tiek atrastas visas klašu atkarības, kur viena klase ir apakšklase citai klasei. Šis solis arī ir atkarīgs no 3. soļa, jo izmanto klašu sarakstu.
8. Ievietot iegūtās klašu-klašu atkarības mērķa datubāzē.

Skatoties uz shēmas izguves plānu var redzēt, ka vairāki soļi ir neatkarīgi no citiem. Tādēļ skripts tiek izstrādāts atdalot šos soļus atsevišķi katru savā funkcijā un galvenajā skripta funkcijā tad šīs funkcijas tiek izsauktas. Šis sniedz skripta izpildes konfigurēšanas iespēju, kur var izkomentēt no skripta dotā momentā nevajadzīgos soļus vai arī vēlāk šo varētu iztaisīt konfigurējamu caur konfigurācijas failu, lai norādītu kuras daļas izpildīt.

4.4. Klašu izguve

Priekš klašu izguves pirmais ir nepieciešamību iegūt sarakstu ar visām Wikidata esošajām klasēm. Pašlaik ir zināms, ka tās nav pārāk liela daudzuma, tas ir nedaudz pāri 80 tūkstošiem. Tādēļ ir iespējams izgūt visas klase vienā vaicājumā, lai arī Wikidata paliekot lielākām šis nākotnē var prasīt vairākus vaicājumus, dēļ Wikidata vaicājumu 60 sekunžu limita.

```
SELECT ?class (COUNT(?y) as ?instances) WHERE {
   ?y wdt:P31 ?class.
}
GROUP BY ?class
ORDER BY DESC(?instances)
```

Attēls 4-2 **Klašu un to instanču skaita SPARQL vaicājums**

Augstāk attēlā redzamais vaicājums tiek izmantots, lai iegūtu visas Wikidata klases. Šo nodrošina "Instance of (P31)" īpašība, pasakot izgūt entītijas, kurām ir vismaz kāda instance("?y" mainīgais dotajā vaicājumā). Papildus šīm klasēm tiek izskaitīts to instanču skaits("?instances" mainīgajā). Šis saraksts ar klasēm tālāk skriptā tiek vairāk kārtēji izmantots, lai nav vairākas reizes jāiegūst šis saraksts. Šis instanču skaits tālāk skriptā tiek izmantots, lai grupētu klases kopā, jo instanču skaits vaicājumos, kuros tiek apskatītas klašu instances, kā tālāk redzamajos klašu īpašību vaicājumos, kopējais klašu instanču skaits korelē ar vaicājuma izpildes laiku.

Tālāk papildus tiek iegūts klašu apakšklašu skaits ar sekojošo vaicājumu:

```
SELECT ?class (COUNT(?y) as ?subclasses) where {
   ?y wdt:P279 ?class.
}
GROUP BY ?class
ORDER BY DESC(?subclasses)
```

Attēls 4-3 **Klašu un to apakšklašu skaita SPARQL vaicājums**

Augstāk attēlā redzamais vaicājums ir ļoti līdzīgs instanču skaita vaicājumam, vienīgais šeit tiek izmantota "subclass of (P279)" Wikidata īpašība, lai iegūtu klases, kurām ir kāda apakšklase ("?y" mainīgais vaicājumā), kuras tiek arī izskaitītas ("subclasses" mainīgajā). Pašā skriptā tālāk tiek iets cauri klasēm ar apakšklasēm un ja tās eksistē klašu vārdnīcā(tika iegūta ar iepriekšējo klašu instanču vaicājumu), tad šajā vārdnīcā dotajai klasei tiek precizēts apakšklašu skaits.

Šis iegūtais apakšklašu skaits noder gan kā aprakstoša klases īpašība, gan tālāk skriptā izmantots vaicājumos, kuros ir apskatītas klases apakšklases, lai grupētu klases kopā, līdzīgi kā tas tiek darīts ar instanču skaitu.

Tā kā izgūstot klašu sarakstu, tiek iegūts tikai tās identificējošās IRI. Tad papildus vaicājumos nākas izgūt šo klašu angliskos nosaukumus("labels"). Šis netika veikts vaicājumā kopā ar klašu saraksta iegūšanu, jo nosaukumu iegūšana no Wikidata ir samērā ilgstoša un nav pat iespējams iegūt visu klašu nosaukumus vienā vaicājumā. Tādēļ skriptā klases tiek grupētas vienumos līdz 15 tūkstošiem. Šādā veidā pašreizējajām 80 tūkstošu klasēm šo var paveikt 6 vaicājumos.

```
SELECT ?class ?classLabel WHERE {
   VALUES ?class { <classList> }
   SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
```

Attēls 4-4 **Klašu nosaukumu ieguves SPARQL vaicājums**

Augstāk redzamais vaicājums tiek izmantots, lai iegūtu klašu nosaukumus. "<classList>" mainīgajā skripts ievieto salīmētu "string" vērtību ar klašu IRIs, interesanti ir pieminēt, ka šajā klašu sarakstā limits ir 15 tūkstoši klašu, nevis tādēļ, ka vaicājums pārāk ilgi izpildās(uz ilgumu limits ir ap 20 tūkstošiem klašu), bet tādēļ, ka izveidotais vaicājums pārsniedz Wikidata SPARQL vaicājumu servisa pieņemto vaicājumu garumu.

Kad ir iegūti klašu nosaukumi skriptam vien atliek ievietot visu šo informāciju par klasēm mērķa datubāzē izmantojot sekojošo vaicājumu:

```
INSERT INTO sample.classes(iri, cnt, display_name, local_name, is_unique, subclasses)
VALUES('{}', {}, '{}', '{}', true, '{}');
```

Attēls 4-5 **Klašu ievietošanas SQL vaicājums**

Augstāk redzamais vaicājums skriptā tiek izmantots kā pamats, kur skripts aizpilda "{}" vietas ar vērtībām. Skripts līmē vairākus šādus vaicājumus par katru klasi kopā un izpilda mērķa datubāzē. "local_name" mainīgais ir Wikidata iekšējie entītiju identifikatori("Q1", "Q562" utt.) skripts pats tos iegūst no klases IRI. Kad visas klases ir ievietotas mērķa datubāzē, datubāzes savienojuma tranzakcija tiek pilnīgi nodota("commited").

Kopumā visu klašu ieguve un ar to saistītās informācijas ieguve, kā arī to ievietošana mērķa datubāzē skriptam prasa aptuveni nedaudz pāri 10 minūtēm. Priekš skripta šis varētu likties ilgi,

bet šī skripta kontekstā tas ir salīdzinoši ļoti ātri, to varēs vēlāk redzēt pie attiecību ieguves, kuras aizņem daudz ilgāku laiku.

4.5. Īpašību izguve

Īpašību izguve bija nedaudz vienkāršāka, kā klašu izguve dēļ mazāka apjoma īpašību nekā klašu. No sākuma vienā vaicājumā tiek izgūts pilns saraksts ar īpašībām un to pielietojumu daudzumu trijnieku attiecībās.

```
SELECT DISTINCT ?property (COUNT(?item) as ?useCount) WHERE {
    ?item ?property ?propValue
}
GROUP BY ?property
ORDER BY DESC(?useCount)
```

Attēls 4-6 **Īpašību un to pielietojuma skaita izguves vaicājums**

Izmantojot augstāk redzamo vaicājumu ir iespējams izgūt visas Wikidata īpašības, kā arī to pielietojuma skaitu. Izgūtās vērtības skriptā tālāk tiek ievietotas vārdnīcā.
Izmantojot iegūto īpašību sarakstu šīm īpašībām tālāk tiek iegūti to nosaukumi līdzīgi kā tas tika darīts klasēm.

```
SELECT DISTINCT ?property ?propLabel WHERE {
  VALUES ?property { <property_list> }
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
  ?prop wikibase:directClaim ?property
}
```

Attēls 4-7 **Īpašību nosaukumu izguves SPARQL vaicājums**

Skripts īpašību nosaukumus izguva grupējot vairākas īpašības kopā, līdz pat 15 tūkstošiem īpašību. Īpašību identificējošie IRIs tika apvienoti kopā “string” vērtībā, ko skripts tālāk ievieto vaicājuma “<property_list>” mainīgajā. Ar īpašībām ir problēma, ka tām nevar tiešā veidā no paša “?property” iegūt nosaukumu izmantojot Wikidata nosaukumu servisu. Wikidata īpašību nosaukumi ir piesaistīti šo īpašību entītijām(“?prop” mainīgais vaicājumā), un ne pašām īpašībām(“?property” mainīgais vaicājumā). Šādi nosaukumi tiek izgūti tikai “:wdt” tipa īpašibām, šī ir vieta, ko varētu tālāk papildināt un apskatīt, lai izlemtu ko darīt ar citu tipu īpašībām.

Tālāk šis izgūtais īpašību saraksts, kurš tagad ir papildināts arī ar īpašību nosaukumiem tiek izmantots, lai ievietotu iegūtās vērtības mērķa datubāzē:

```
INSERT INTO sample.properties(iri, cnt, display_name)
VALUES('{}', {}, '{}');
```

Attēls 4-8 **Īpašību ievietošana mērķa datubāzē**

Mērķa datubāzē tiek ievietota sekojošā informācija par katru īpašību: to identificējošais IRI, to pielietojuma skaits, kā arī to nosaukums. Tālāk izstrādājot skriptu šeit būtu nepieciešamība vēl apstrādāt īpašību IRIs un izgūt no tiem īpašību tipus un to Wikidata identifikatorus ("P31", "P579" utt.), kurus arī vajag ievietot mērķa datubāzē. Lai arī šī tipu un Wikidata identifikatoru izgūšana var būt salīdzinoši sarežģīta dēļ vairākiem īpašību tipiem(virs 10).

Īpašību un ar tām saistītās informācijas izguve un ievietošana mērķa datubāzē iet vēl ātrāk kā klasēm, dēļ mazāka īpašību skaita. Visas īpašības tiek apstrādātas aptuveni 5 minūtēs.

4.6. Klašu-klašu attiecību izguve

Klašu savstarpējās attiecības, kur viena klase ir apakš klase citai klasei, tiek izgūtas izmantojot pie klašu izguves iegūto klašu vārdnīcu un sekojošo vaicājumu:

```
SELECT DISTINCT ?class ?subclass WHERE {
  ?subclass wdt:P279 ?class.
  VALUES ?class { <class_list> }
}
```

Attēls 4-9 **Klašu attiecību izguves vaicājums**

Skripts secīgi iet cauri klašu vārdnīcai un grupē tās kopā attiecīgi pēc klašu apakšklašu skaita, jo klašu apakšklašu skaits korelē ar augstāk redzamā vaicājuma izpildes laiku. Šeit un daudzās citās vietās skripts vaicājumos grupē kopā pēc iespējas vairāk klases, lai maksimizētu vaicājumu izpildes laiku dēļ ierobežotā vaicājumu skaita minūtē pret Wikidata.

Līdz ko skripts savāc vairāk par 50 tūkstošiem attiecību, tas mēģina ievietot šīs attiecības mērķa datubāzē izmantojot sekojošo vaicājumu:

```
INSERT INTO sample.cc_rels(class_1_id, class_2_id, type_id)
SELECT (SELECT id from sample.classes WHERE iri = '{class1Iri}') AS cl_id,
(SELECT id from sample.classes WHERE iri = '{class2Iri}') AS cl2_id,
(SELECT id from sample.cc_rel_types WHERE name = 'sub_class_of')
HAVING (SELECT id from sample.classes WHERE iri = '{class1Iri}') IS NOT NULL
AND (SELECT id from sample.classes WHERE iri = '{class2Iri}') IS NOT NULL;
```

Attēls 4-10 **Klašu attiecību ievietošanas SQL vaicājums**

Tā kā iekš mērķa datubāzes klases un īpašības tiek identificētas pēc atsevišķām tabulu identifikatora vērtībām, bet pašā skriptā un Wikidata tās tiek identificētas pēc IRI, tad ievietojot klašu attiecības mērķa datubāzē šie klašu identifikatori tiek atlasīti no klašu tabulas dotām klašu IRI vērtībām. Līdzīgi attiecības veida identifikators tiek atlasīts no īpašas klašu attiecību tipu tabulas, kur attiecība ir "apakš klase". Bet šīs atkarības tiek ievietotas tikai, ja šādi IRIs eksistē mērķa datubāzes klašu tabulā.

Klašu attiecību iegūšana iet salīdzinoši ilgu laiku, jo kopumā eksistē 80 tūkstoši klašu un aptuveni pirmās 200 klases tiek grupētas kopā pa 1 līdz 10 klasēm, bet pēdējās klases ar dažām apakšklasēm tiek grupētas kopā līdz pat 15 tūkstošiem klašu. Bet kopumā šis aizņem aptuveni 1 stundu, iegūstot ap 80 tūkstošiem attiecību.

4.7. Klašu-īpašību attiecību izguve

Klašu un īpašību attiecību izguve strādā līdzīgi klašu attiecību izguvei, vienīgais šeit ir nepieciešams atsevišķi iegūt gan klasē ienākošās īpašības, kur dotas klases instances ir objekts, gan no klases izejošās attiecības, kur dotās klases instances ir subjekts.

Priekš izejošo attiecību iegūšanas un ienākošo īpašību iegūšanas tiek izmantots ļoti līdzīgs vaicājums.

```
SELECT ?property ?class (COUNT(?y) AS ?propertyInstances) WHERE {
   ?x wdt:P31 ?class.
   ?x ?property ?y. #?y ?property ?x. For incoming relations
   VALUES ?class { <class_list> }
}
GROUP BY ?property ?class
```

Attēls 4-11 **Klašu un īpašību attiecību iegūšanas vaicājums**

Starp izejošām un ienākošām attiecībām šajā vaicājumā atšķiras tikai "?x ?property ?y" rindiņa, kur priekš ienākošām attiecībām "?x" tiek samainīts ar vietām ar "?y". Mainīgajā "?x" tiek atlasītas doto klašu instances, priekš kurām tālāk skatās ar tām saistītajām īpašībām. Papildus "?propertyInstances" mainīgajā tiek izskaitīts, cik daudz klases instances ir saistītas ar doto īpašību. Arī šajā vaicājumā skripts grupē kopā pēc iespējas vairāk klases, bet šeit klases tiek grupētas pēc to instanču skaita, jo vaicājuma izpildes laiks korelē ar klašu kopējo instanču skaitu.

Kad tiek iegūtas vismaz 50 tūkstoši attiecības, tās tiek ievietotas mērķa datubāzē ar sekojošo vaicājumu:

```
INSERT INTO sample.cp_rels(class_id, property_id, type_id, cnt, object_cnt)
SELECT (SELECT id from sample.classes WHERE iri = '{classIri}') AS cl_id,
(SELECT id from sample.properties WHERE iri = '{propIri}') AS pr_id,
(SELECT id from sample.cp_rel_types WHERE name = '{propertyDirection}'),
{cnt},
{objectCnt}
HAVING (SELECT id from sample.classes WHERE iri = '{classIri}') IS NOT NULL
AND (SELECT id from sample.properties WHERE iri = '{propIri}') IS NOT NULL;
```

Attēls 4-12 **Klašu un īpašību attiecību ievietošanas SQL vaicājums**

Šis vaicājums līdzīgi kā klašu attiecību vaicājums iegūst nepieciešamos klašu un īpašību identifikatorus no attiecīgajām tabulām pēc klašu un īpašību IRIs. Klases un īpašības attiecības tips tiek atlasīts no "cp_rel_types" tabulas, kur attiecības tipa nosaukums ir "Outgoing" priekš izejošām attiecībām un "Incoming" priekš ienākošām attiecībām. Objektu daudzums visām attiecībām šeit tiek aizpildīts ar 0, kas tiek vēlāk atjaunots priekš izejošām attiecībām.

Tālāk tikai priekš izejošām attiecībām tiek atsevišķā vaicājumā iegūts attiecību skaits priekš dotas klases un īpašības, kur objekts, jeb "?y" mainīgais ir entītija, jeb ne literāļa vērtība, tāda kurai ir savs IRI. Šis skaits tiek iegūts ar sekojošo vaicājumu:

```
SELECT ?property ?class (COUNT(?y) AS ?objectCnt) WHERE {
   ?x wdt:P31 ?class.
   ?x ?property ?y.
   FILTER  isIRI(?y)
   VALUES ?class { <class_list> }
}
GROUP BY ?property ?class
```

Attēls 4-13 **Klašu un īpašību attiecību objektu skaita iegūšanas vaicājums**

Šis vaicājums ir ļoti līdzīgs parastajam klašu un īpašību attiecību iegūšanas vaicājumam, vienīgais šajā vaicājumā ir pievienots papildus filtrs, kas atlasa tikai vērtības, kur objektam, jeb "?y" mainīgajam ir IRI. Izpildot šo vaicājumu klases arī tiek grupētas kopā pēc to instanču skaita, lai maksimizētu vaicājuma izpildes laiku. Kad tiek iegūts objektu daudzums priekš vismaz 50 tūkstošiem attiecību, šis objektu daudzums tiek atjaunots mērķa datubāzē.

Klašu un īpašību attiecību ieguve ir garākā skripta daļa, jo sanāk iziet cauri visām klasēm 3 reizes, lai iegūtu ienākošās attiecības, izejošās attiecības un objektu daudzumu priekš izejošām

attiecībām. Kopumā šī skripta daļa iet aptuveni 5 stundas, kopā izgūstot aptuveni 5 miljonus attiecību.

4.8. Izgūtās shēmas pielietojums

Šo izgūto shēmu, kā iepriekš darbā ir minēts var pielietot RDF vaicājumu veidošanas un analīzes rīkos, kur ir svarīgi spēt ātri izgūt Wikidata klases, īpašības vai to attiecības. Lokāla datubāze, kas satur Wikidata shēmu sniedz ātrāku atbildes laiku, nekā vaicāšana pret pašu Wikidata.

Lai pierādītu šo pieņēmumu, apskatīsim piemēru, kur kā par piemēru rīkam ņemsim "ViziQuer" RDF vaicājumu analīzes rīku. Šajā rīkā var veidot RDF vaicājumus, kur rīks palīdz ar vaicājumu veidošanu, piemēram, sakot priekšā kādas īpašības ir iespējamas dotai klasei. Lai iegūtu visas iespējamās īpašības dotai klasei, šajā piemērā klasei "Sala (wdt:Q23442)", bija iespējams iegūt saistīto īpašību IRI sarakstu vaicājot pret pašu Wikidata ar vaicājumu:

```
SELECT DISTINCT ?property WHERE {
  ?x wdt:P31 ?class.
  ?x ?property ?y. # ?y ?property ?x. for incoming relations
  VALUES ?class { <http://www.wikidata.org/entity/Q23442> } # Q23442 - Island
}
```

Attēls 4-14 **Vaicājums, lai iegūtu īpašību sarakstu no Wikidata**

Šis vaicājums paņēma 26026ms, jeb 26 sekundes, lai iegūtu izejošās īpašības un 5417ms, jeb 5 sekundes, lai iegūtu ienākošās īpašības, kopā paņemot aptuveni 30 sekundes.

Lai iegūtu īpašību IRI sarakstu no lokālās datubāzes priekš "Sala" klases, tika izmantots vaicājums:

```
SELECT iri FROM sample.properties
WHERE id IN
    (SELECT property_id FROM sample.cp_rels
     WHERE class_id = (
        SELECT id from sample.classes where iri = 'http://www.wikidata.org/entity/Q23442'));
```

Attēls 4-15 **Vaicājums īpašību ieguvei no lokālās datubāzes**

Šis vaicājums pret lokālo datubāzi izpildījās un atgrieza rezultātus 45ms, kas ir nesalīdzināmi ātrāk nekā to pašu īpašību iegūšana no paša Wikidata.

4.9. Skripta nepilnības un iespējami uzlabojumi

Darbā izstrādātais skripts ir tikai kā pirmā versija šim shēmas izgūšanas skriptam. Un tam vēl joprojām ir vairākas nepilnības un lietas, kuras vajag papildināt:

- Skripts pašlaik tikai izdrukā konsolē neizdevušos vaicājumus, kas pārsniedza laika limitu vai arī pēc 3 reižu pārmēģināšanas neizpildījās. Bet šādus vaicājumus un iespējams citu aprakstošu informāciju vajadzētu pierakstīt "log" failā.
- Viena no pašlaik lielākajām šī skripta nepilnībām, ir, ka tas spēj tikai izgūt visu Wikidata shēmu, bet ne atjaunināt to. Skriptā būtu nepieciešams izstrādāt papildus režīmu, ko varētu iekonfigurēt, lai skripts tikai atjaunina shēmu ar kopš pēdējās izpildes Wikidata pievienotām klasēm, īpašībām un to attiecībām.
- Līdz beidzot būtu noderīgi optimizēt skriptu, lai tas uz brīžiem neaizņemtu tik daudz resursu, tas ir, pāri 2GB brīvpiekļuves atmiņas.

SECINĀJUMI

Viens no vislielākajiem šķēršļiem ar kuriem darba autors saskārās darba izstrādes laikā, bija darba autora mazā pieredze ar semantisko tīmekli un RDF datu modeli, jo šie koncepti bija tikai teorētiski apgūti un nebija praktiskas pieredzes ar tiem. Šis nedaudz apgrūtināja Wikidata shēmas izguves skripta modelēšanu un tā izstrādi.

Darba autors uzskata, ka izstrādātais shēmas izguves skripts ir pilnvērtīgi strādājošs un tas spēj izgūt Wikidata saistītā datu avota shēmu, kas var noderēt daudzos rīkos, kas analizē vai darbojās ar Wikidata shēmu, bet ir vēl daudz aspekti pie kuriem jāpiestrādā, lai optimizētu un pilnveidotu šo skriptu. Pie kā darba autors arī turpinās strādāt.

Šī darba izstrādes gaitā tika iegūtas neapšaubāmi noderīgas zināšanas par semantisko tīmekli, RDF datu modeli un saistīto datu avotiem. Kā arī tika iegūta praktiska pieredze strādājot ap šiem konceptiem. Tālākā darba autora profesionālajā karjerā var ļoti noderēt šīs zināšanas, piemēram, kā programmatūrai iegūt un apstrādāt semantisku informāciju un datus.

Bakalaura darbā izgūtā Wikidata shēma ir kā tikai pamats, lai ar tās palīdzību tālāk varētu veidot un analizēt vizuālos vaicājumus pār Wikidata informāciju dažādos vizuālo vaicājumu rīkos.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. “Semantic Web” World Wide Web Consortium - [atsauce 08.05.2021]. Pieejams internetā:https://www.w3.org/standards/semanticweb/
2. “The Semantic Web” Scientific American, 2001.gada maijs - [atsauce 08.05.2021]. Pieejams internetā: https://www-sop.inria.fr/acacia/cours/essi2006/Scientific%20American_%20Feature%20Article_%20The%20Semantic%20Web_%20May%202001.pdf
3. N. Shadbolt, T. Berners-Lee and W. Hall, "The Semantic Web Revisited," in IEEE Intelligent Systems, vol. 21, no. 3, pp. 96-101, Jan.-Feb. 2006. [atsauce 10.05.2021] Pieejams internetā: https://ieeexplore.ieee.org/document/1637364
4. “Components of the Semantic Web” Russell Kay, 2006.gada 27 februāris - [atsauce 10.05.2021] Pieejams internetā: https://www.computerworld.com/article/2562361/sidebar--components-of-the-semantic-web.html
5. “RDF Concepts and Abstract Syntax Publication History” World Wide Web Consortium - [atsauce 11.05.2021]. Pieejams internetā:https://www.w3.org/standards/history/rdf-concepts
6. “RDF 1.1 Concepts and Abstract Syntax” World Wide Web Consortium, 2014.gada 25.februāris - [atsauce 11.05.2021]. Pieejams internetā:https://www.w3.org/TR/rdf11-concepts/
7. “XML RDF” W3Schools - [atsauce 11.05.2021]. Pieejams internetā:https://www.w3schools.com/xml/xml_rdf.asp
8. “What’s the best RDF serialization format?” Joep Meindertsma, 2019.gada 29.jūnijs - [atsauce 12.05.2021]. Pieejams internetā:https://ontola.io/blog/rdf-serialization-formats/
9. “Linked Data” Tims Bērness-Lī, 2006.gada 27.jūlijs - [atsauce 14.05.2021]. Pieejams internetā:https://www.w3.org/DesignIssues/LinkedData.html
10. “DBpedia and the live extraction of structured data from Wikipedia” Mohamed Morsey, Jens Lehmann, Soren Auer, Claus Stadler, 2012.gada aprīlis.

11. “DBpedia: A Nucleus for a Web of Open Data” Soren Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak un Zachary Ives, 2007.gads.
12. “DBpedia Ontology” DBpedia asociācija, 2021.gads - [atsauce 14.05.2021]. Pieejams internetā:https://www.dbpedia.org/resources/ontology/
13. “Wikidata:Introduction” Wikimedia projekta fonds, 2021.gada 29.marts - [atsauce 15.05.2021]. Pieejams internetā:https://www.wikidata.org/wiki/Wikidata:Introduction
14. “Wikidata Announcing the release of the Wikidata Query Service” Dan Garry, 2015.gada 8.septembris - [atsauce 15.05.2021]. Pieejams internetā:https://lists.wikimedia.org/hyperkitty/list/wikidata@lists.wikimedia.org/message/N2HPRCYIWGLM2IDTNCHQLNY574H5ZEQR/
15. “Wikidata:WikiProject Ontology/Modelling” Wikimedia projekta fonds, 2020.gada 11.augusts - [atsauce 15.05.2021]. Pieejams internetā:https://www.wikidata.org/wiki/Wikidata:WikiProject_Ontology/Modelling
16. “Wikidata Entity Usage Project” Wikimedia Metrics - [atsauce 17.05.2021]. Pieejams internetā:https://grafana.wikimedia.org/d/000000176/wikidata-entity-usage-project?orgId=1&var-project=enwiki
17. “Wikidata Query Service/User Manual” Wikimedia projekta fonds, 2021.gada 2.maijs - [atsauce 17.05.2021]. Pieejams internetā:https://www.mediawiki.org/wiki/Wikidata_Query_Service/User_Manual
18. “Wikidata:How to use data on Wikimedia projects” Wikimedia projekta fonds, 2021.gada 18.aprīlis - [atsauce 18.05.2021]. Pieejams internetā:https://www.wikidata.org/wiki/Wikidata:How_to_use_data_on_Wikimedia_projects
19. Kārlis Čerāns, Jūlija Ovčiņņikova, Uldis Bojārs, Mikus Grasmanis, Lelde Lāce, Aiga Romāne. “Schema-Backed Visual Queries over Europeana and other Linked Data Resources” To appear in Poster and Demo Proceedings, ESWC 2021

Bakalaura darbs „Datu shēmas izguve no wikidata vizuālo vaicājumu atbalstam” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti.

Autors: (elektr. paraksts*)* Ginters Juris Vehi 28.05.2021.

Rekomendēju darbu aizstāvēšanai

Vadītāj*s*: profesors Dr. dat. Kārlis Čerāns (elektr. paraksts*)* 28.05.2021.

Recenzents: asociētā profesore Dr. dat. Vineta Arnicāne

Darbs iesniegts Datorikas fakultātē 31.05.2021.

Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

__.06.2021. prot. Nr. ___.

Komisijas sekretārs: docents Dr. dat. Ivo Odītis (elektr. Paraksts*)*>