Latvijas Universitāte
Datorikas fakultāte

RDB2OWL izteiksmju modeļa redaktors TDA vidē

Bakalaura darbs

Autore:Gerda Diteriha
st.apl.nr.:gd08020
darba vadītājs:Kārlis Čerāns
Latvijas Universitātes Matemātikas un informātikas institūta
vadošais pētnieks

Rīga, 2012

Anotācija

Šī bakalaura darba ietvaros tika izstrādāts(un joprojām ir izstrādes procesā) RDB2OWL izteiksmju modeļa redaktors TDA vidē. Lai veiksmīgi to varētu veikt, autore iepazinār ar RDB2OWL izteiksmju valodu, kas ļauj definēt attiecības starp relāciju datu bāzēm un tām atbilstošām ontoloģijām. Izstrādei tikai izmantots OWLGrEd redaktors, kas izmanto OWL sintaksi ontoloģiju aprakstīšanai un Lua programmēšanas valoda ar lQuery papildinājumu, kas ļauj veidot vizuālas lietotājformas. Rezultātā tika iegūts pagaidām ļoti vienkāršs izteiksmju redaktors ar savu metamodeli un iespēju piesaistīt citu metamodeli(Šajā gadījumā RDB2OWL), tajā pašā laikā saglabājot redaktora universalitāti.

Atslēgas vārdi : RDB2OWL, RDF, OWL, Semantiskais tīmeklis

Abstract

During the development of bachelor's work, author started the development of RDB2OWL expression model redactor in TDA. To make it happen, the author had to learn about RDB2OWL mapping language, that allows to define relations between relation data bases and ontologies. Also OWLGrEd redactor, which uses the OWL syntax for defining the ontologies was used for the development, so was Lua programming language with lQuery library, which allows us to visualize user-forms. As a result we got very simple expression redactor, with its own meta-model and possibility to link it to another meta-model(in this case RDB2OWL), in the mean-time remaining as universal redactor.

Key words: RDB2OWL, RDF, OWL, Semantic web

Satura rādītājs

Attēlu saraksts

Terminu skaidrojums

RDF - Resursu aprakstīšanas ietvars (*Resource Description Framework*) metadatu modelis

URI - Vienotais resursu identifikators(*Uniform Resource Identificator*) Interneta protokolu

elements, kas sastāv no īsas simbolu virknes, kas atbilst noteiktai sintaksei.

SQL - (Structured Query Language"strukturēto vaicājumu valoda") -vaicājumu valoda, kas paredzēta datu manipulēšanai relāciju datubāžu pārvaldības sistēmās.

Ontoloģija ir zināšanu formāls attēlojums, izmantojot klašu un to īpašību formā ietvertus jēdzienus

SPARQL-*SPARQL Protocol and RDF Query Language.*Vaicājumu valoda, kas ļauj atlasīt datus no RDF datu bāzes.

TDA platforma- (*Transformation-Driven Architecture*) –sistēmu/rīku būves platforma.

UML – Universālā modelēšanas valoda(*Unified Modeling Language)*, tā ir grafisko valodu kopa, kas ir balstīta uz metamodelēšanu .

OWL - Tīmekļa ontoloģijas valoda(W3C standarts).

Ievads

Ņemot vērā semantiskā tīmekļa attīstības ātrumus, jaunu standartu kā RDF, OWL, SPARQL ieviešanu, ir svarīgi nodrošināt iespēju "vecos tehnoloģiju" datus izmantot arī "jaunajās tehnoloģijās". Šajā darbā autore koncentrējs uz datu bāzēm, respektīvi relāciju datu bāzēm(vecā tehnoloģiju) un RDF/OWL(jaunā tehnoloģija). Lai arī mainās tehnoloģijas, dati,ko uzkrājušas un uzglabājušas "vecās tehnoloģijs" joprojām ir aktuāli un saprotama ir vēlme migrēt uz jaunākajām tehnoloģijām.

Vajadzība pēc relāciju datu bāzu un RDF datu bāzu atbilstībām ir loģiska. Šeit talkā nāk RDB2OWL,kas tiks apskatīts šajā bakalaura darbā, tālākajās nodaļās.

Sākumā 1. nodaļā apskatīsim iemeslu kādēļ RDB2OWL ir svarīgs – saistību ar "reālo dzīvi", jeb ieskats semantiskajā tīmeklī un tā uzbūvē.

Tālāk tiks apskatīti rīki un tehnoloģijas, kas bija jāizmanto/jāapgūst, lai varētu tapt šis bakalaura darbs - sākot ar redaktoru un programmēšanas valodu un beidzot ar RDB2OWL loģiskas izprašanu.

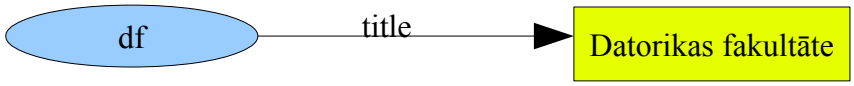
4. nodaļā jau tiks apskatīts rezultāts, aprakstīti algoritmi, viss kas saistās tieši ap pašu redaktoru un tā darbības principiem. Kā redaktors strādā tagad un kā tam būtu jāstrādā nākotnē.

1.Semantiskā tīmekļa raksturlielumi

Semantiskais tīmeklis ir globālā tīmekļa "nākamā pakāpe" – WEB 3.0. Šajā tīmeklī datiem tiek pievērsta daudz lielāka nozīme un pielietojamība. Dati tiek padarīti "gudri" jeb datiem piešķir zināmu semantiku. Semantiskajā tīmeklī(idēlā variantā) ikkatras interneta vietnes datiem ir piešķirta semantika, kuru saprot arī dators. Tāpat viss globālais tīmeklis drīzāk kļūst par globālu grafu, kurā ir iespējams veiksmīgi iegūt datus no vairākām šādām grafu virsotnēm. Brīdī, kad dati ir "gudri" t.i., tīmeklis var veikt secinājumus no šiem datiem,datu izmantojamība paplašinās. Datu "gudrību" nodrošina RDF datu bāzes un OWL pieraksts.[4]

RDF ir Resursu aprakstīšanas ietvars (*Resource Description Framework*) metadatu modelis, kurš ir balstīts uz ideju, ka tiek izteikti apgalvojumi par resursiem formā "subjekts-predikātsobjekts". Subjekts ir resurss, "lieta", kas tiek aprakstīta. Predikāts ir aprakstāmais resursa aspekts. Objekts ir konkrētajā aspektā ar subjektu saistītais resurss vai vērtība.

Atslēgas vārds runājot par RDF ir šie trijnieki(subjekts, predikāts, objekts). Aprakstīsim datorikas fakultāti ar RDF trijnieku. Grafiski tas izskatās šādi:



Drawing 1: RDF trijnieka piemērs [4]

RDF Datu bāzes - Tā ir samērā jauna datu bāzu tehnoloģija,kurā informācija par resursiem tiek glabāta grafa veidā izmantojpt RDF. Atšķirībā no relāciju datu bāzēm, RDF DB dati tiek salikti "vienā maisā" un pierakstīti RDF trijnieku veidā. Šādi "maisā" var salikt neierobežoti daudz RDF trijniekus,kas ir spējīgi aprakstīt praktiski jebko. RDF arī ir tas, kas tīmeklī veido "gudros" datus. [4]

Ja šāda datu "samešana vienā maisā" neliekas skaista, jo varbūt rodas izluzijas,ka dati ir nesakārtoti, tad šajā pašā "maisā" var ielikt datu bāzes struktūras aprakstu - definēt datu bāzes ontoloģiju un RDF Shēmas jeb klases[4]. Ontoloģija apraksta struktūru – klases un attiecības starp tām un RDF shēma RDF resursus attiecina uz vajadzīgo klasi. Šis ir būtisks ieguvums attiecībā pret relāciju datu bāzēm, jo RDF DB pieļauj ne tik "ķēpīgu" struktūras maiņu. Datus neietekmē datu bāzes uzbūves maiņa. Relāciju datu bāzē tā jau ir problēma.

Vaicājumus RDF datu bāzēs vizbiežāk veic izmantojot SPARQL un tā tas it arī RDB2OWL

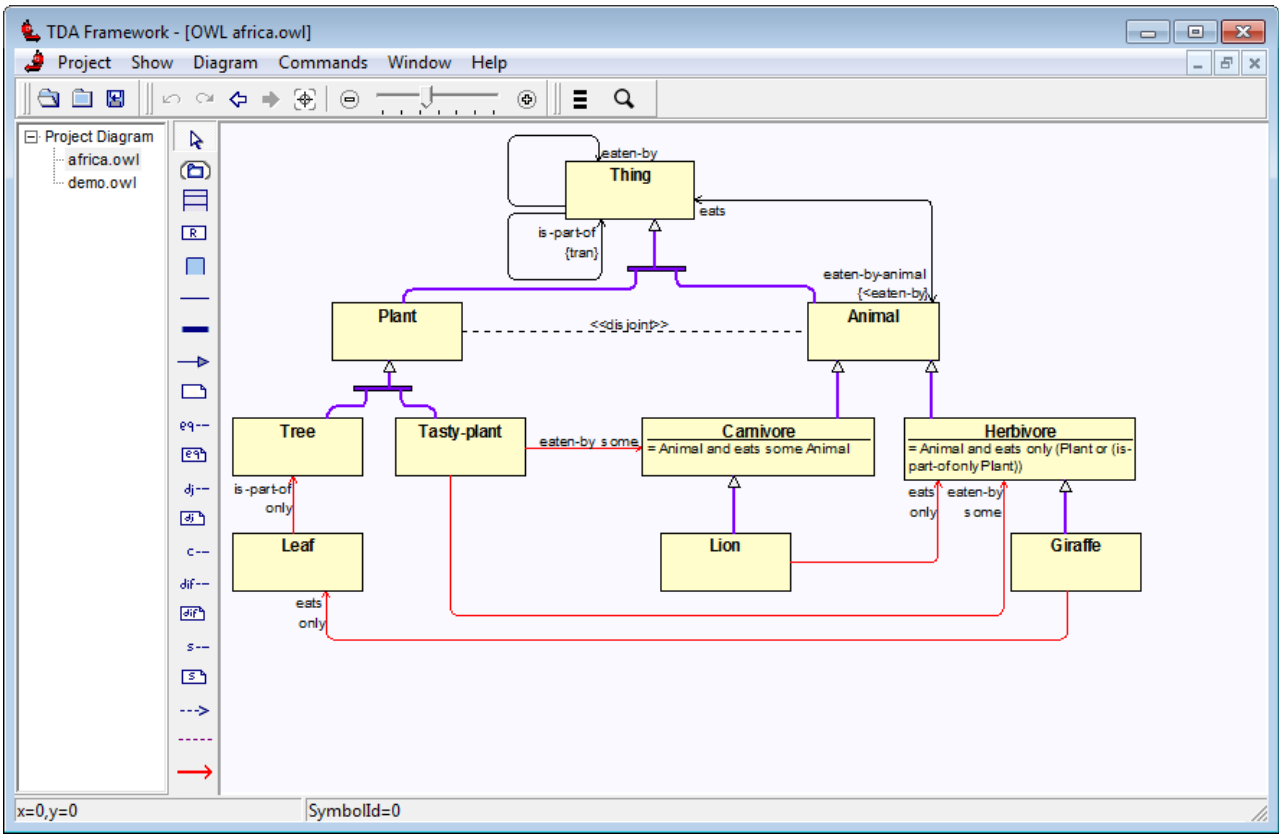
gadījumā. SPARQL jeb SPARQL Protocol and RDF Query Language W3C ir atzinis par standartu un rekomendē. SPARQL saprotams ir ļoti līdzīga citām SQL tipa valodām, tāpat kā citām valodām datu atlase notiek ar SELECT. Piemēram:

```
PREFIX df: <http://www.lu.lv/faculties/df#>
SELECT ?title
WHERE { <http://www.lu.lv/faculties/df> df:title ?title }
```

Šajā piemērā mēs vēlamies atlasīt objekta http://www.lu.lv/faculties/df propertiju - title. Rezultātā mēs saņemsim tekstu :"Datorikas fakultāte".

2.OWLGrED

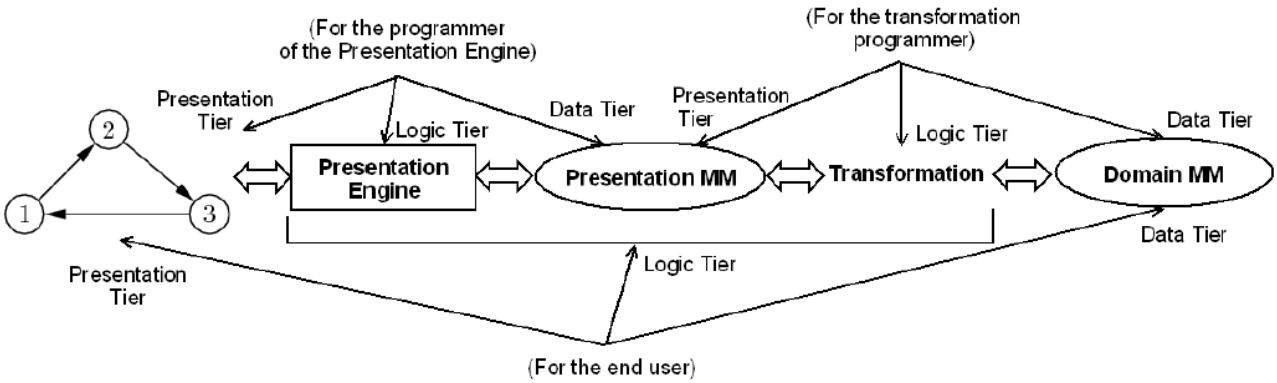
OWLGrEd ir uz UML valodas balstīts grafisks ontoloģiju redaktors, kas vizualizē OWL ontoloģijas. Tas ne tikai spēj vizualizēt ontoloģijas izmantojot paplašinātās UML klases diagrammu notācijas valodu, bet arī ļauj rediģēt ontoloģijas, ar daudz vairāk iespējām kā lielākā daļa līdzīgu rīku. Tāpar OWLGrEd ir savienojams ar Protege 4.[2]



Drawing 2: Atvērts OWLGrEd

Ontoloģijas parasti tiek aprakstītas OWL sintaksē, kas ir ļoti ērts apraksta veids datoram, taču diezgan nepatīkams cilvēkam. Tādēļ tiek radīti grafiskie redaktori. OWLGrEd nebūt nav pirmais, taču visnotaļ viens no ērtākajiem, ja ne pats ērtākais un cilvēkam draudzīgākais. OWLGrEd ir izvēlēts UML un UML paplašināto klašu attēlojumveids, jo konceptuāli OWL un UML ir līdzīgi. Taču OWL joprojām ir vairāk un plašākas iespējas, tādēļ UML tiek paplašināts ar papildus grafiskām vizuālizācijas iespējām, lai varētu veiksmīgi attēlot OWL semantiku grafiskā, cilvēka acij patīkamā veidā.

Tieši ar šī rīka palīdzību ir tapis šis bakalaura darbs. Tika izmantots OWLGrEd rīks,kas izstrādātais ar TDA rīku būves pieeju, kas atbalsta MVC uzbūves principus. Domēnu var izskatīt par moduli(module), prezentācija kā skats(view) un process kā kontrolieris(controller) [6]



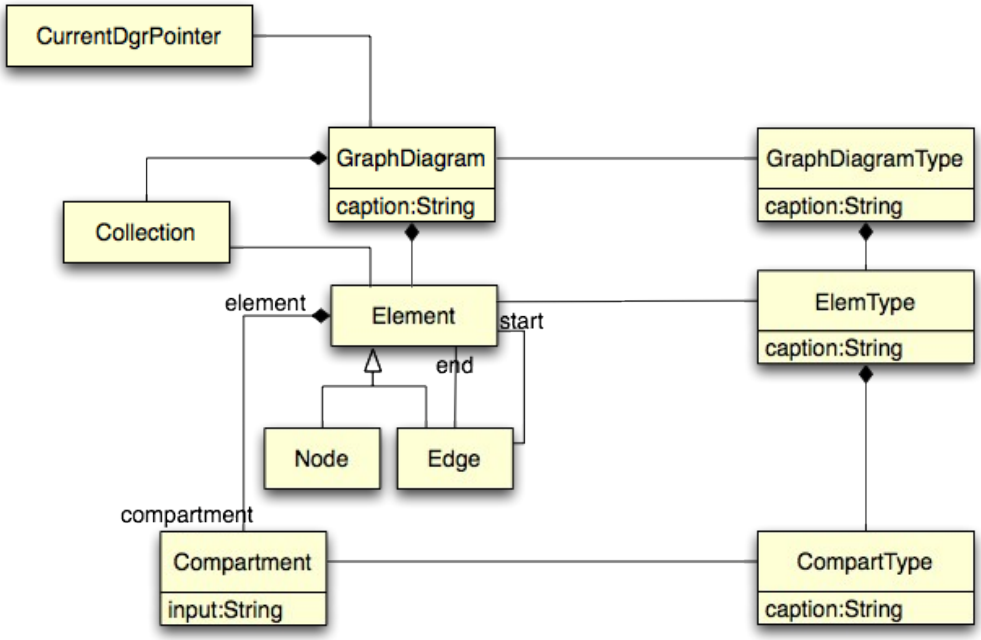
Drawing 3: Attiecības starp gala lietotāja attēlojumu un domēnu

[6]

2.1.TDA

Transformation driven architecture (TDA) – Transformāciju vadītā arhitektūra – uz modeļu transformācijām, saskarnes meta-modeļiem ar atbilstošiem dzinējiem un komandu un notikumu mehānismu bāzēta sistēmas būve[8]

TDA balstās uz metamodeļiem – piemēram vairākiem prezentācijas modeļiem, kā piemēram *Graph m Diagram Engine* un *Dialog Engine*, kas ir savienoti ar pārējo ar komandām un notikumiem[6]



Drawing 4: Graph Diagram Engine

- GraphDagram instances - apraksta katra elementa reālu attēlojumu
- GraphDiagramType - apraksta katra elementa tipu
- GraphDiagramStyle(kas arī var tikt piekārtots GraphDiagram) - apraksta katra elementa attēlojuma stilu

TDA būtiskākās īpašības[6]

- Datus glabā repozitorijā(Sesame, JgraLab utt),kam piekārtota kāda API saskarne. Turklāt var būt tikai viens repozitorijs.
- API jābūt pieejamama augsta līmeņa programmēšanas valodā(piemēram JAVA vai C++), kurā arī tiktu aprakstīta prezentācijas engine
- Metamodelis programmas palaišanas laikā nemainās, mainās tikai instances.
- Tikai viens modulis drīkst piekļūt repozitorijam.

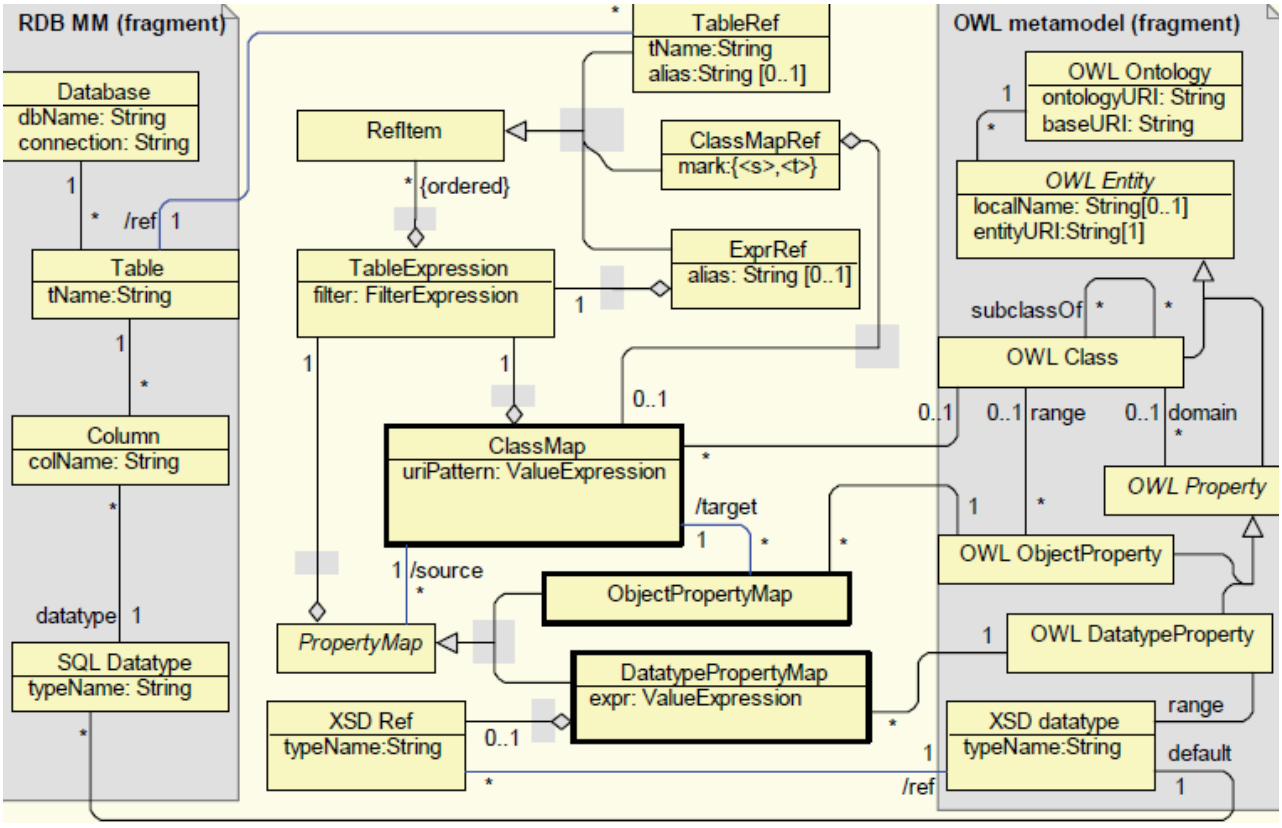
3.RDB2OWL

RDB2OWL ir datu bāzu specifikācijas valoda, kas ir paredzēta RDB-uz-RDF/OWL attēlojumiem, atbalstot datu bāzes un ontoloģijas atbilstības cilvēkam saprotamā veidā.RDB2OWL attēlojumi arī izmanto OWL ontoloģijas struktūru. RDB2OWL ir vienkārš paņēmiens relāciju datu bāzu attēlošanai OWL ontoloģijas veidā.[1]

RDB2OWL līdzīgi kā OWL attēlojumu specifikācijai izmanto klašu ontoloģijas anotācijas, kur glabā informāciju par datu bāzi.

Ir 3 RDB2OWL valodas "slāņi"

- the Raw language (nav domāta gala lietotājam)
- the Core language (uz šo pamtā balstās šis bakalaura darbs)
- Extended language



Drawing 5: RDB2OWL Raw

Tā kā rodas problēmas ar pilnā RDB2OWL metamodeļa attēlošanu, un ņemot vērā,ka pagaidām redaktors neapstrādā visas klases, par piemēriem turpmāk tiks izmantots pamatā Raw language metmodelis(metamodeļa attēliem tikai).

Metamodelis – Raw language(Attēlots Raw metamodelis tikai tā iemesla dēļ,ka tas ir mazāks pēc apjoma un tik un tā dod ieskatu RDB2OWL uzbūvē)

Pilnais RDB2OWL meta modelis ir atrodams pielikumā.

RDB2OWL valodas īpašības :

- Relāciju DB kolonnu un atslēgu saglabāšana/izmantošana iespēju robežās
- Vienkārša un intuitīva, cilvēkam viegli saprotama sintakse attēlojumu izteiksmēm vienkāršos gadījumos, kā arī iespēja aprakstīt daudz sarežģītākus gadījumus
- Gan iebūvētas, gan lietotājdefinētas funkcijas
- Paplašināta attēlošana, piemēram, vairākklašu konceptulizācija(iespēja,klases sadalīt pēc "vairākiem griezumiem")
- Arī advancētāku attēlojumu definēšanas iespējas, piemērsam, iespēja izmantot SQL līmeņa papildu struktūras(piemēram, lietotāja definētas tabulas un pagaidu tabulas, kā arī SQL skati), saglabājot principu,ka RDB paliks “read-only”

Tālāk attēlā ir piemērs kā relāciju datu bāzes uzbūve atbilst ontoloģijai OWLGrEd redaktorā(Atkal vienkāršības labad piemērā ir RDB2OWL Raw)

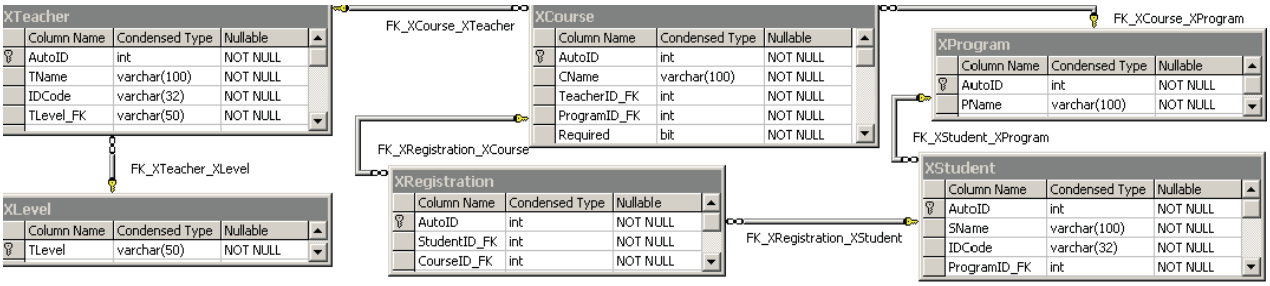


Figure 3. A mini-university database schema

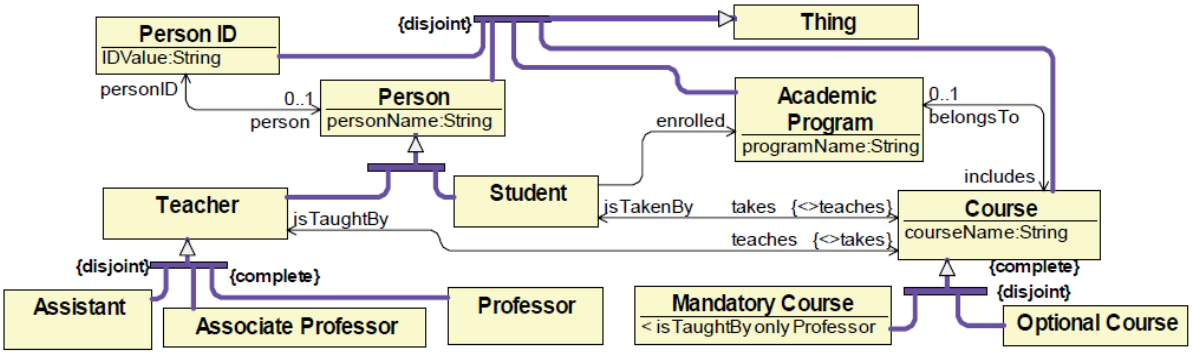


Figure 4. OWL ontology corresponding to the mini-University

Drawing 6: RDB atbilstības ontoloģijai

RDB2OWL sintakse ir ļoti līdzīga SQL, taču neiekļauj visas SQL iespējas. Tākā RDB2OWL izteksmes veido hierarhisku struktūru, kur katru līmeni var atpazīt pēc piešķirtā aizstājvārda, tiek izmantots, tā sauktais, fqn(angliski - fully qualified names) kolonnu apzīmēšanai formā:

an.(an-1.(an-2. … .(a1.(a0.c))…)),

kur

c – datu bāzes tabulas kolonnas vārds,

a0 – tabulas nosaukums,

a1..an - prefiksi

prefikss ir aizstājvārds vai "class map" iezīme.[1]

RDB2OWL sintakses piemēri[1]:

- *Person*
- *Person P, Address A; P.AdrID_FK=A.AdrID*

- *(Person P, Address A; P.AdrID_FK=A.AdrID) PA, (Company, WorksFor; CID=CID_FK); PA.(P.PersonID)= WorksFor.PersonID_FK*
- *<s>, <t>; <s>.AdrID_FK = <t>.AdrID*

Tāpat ir iespējams pievienot arī informāciju par URI : *{uri=(<item1>,…,<itemk>)},* kur katrs "item" ir vērtības izteksme (parastu parasta teksta vērtība, vai norāde uz datubāzes tabulas kolonnu); kas apraksta konversiju uz URI pierakstu formu un visu item konkatenāciju.
Piemēri[1]:

- *Person {uri=('XPerson',PersonID)}*
- *Person P, Address A; P.AdrID_FK=A.AdrID {uri=('XPerson',PersonID)}.*

3.1.R2B metamodelis

RDB2OWL metamodelī(manā darbā OWLGrEd metamodelī atzīmēts ar prefisku R2B#) ir 3 daļas. Ir relāciju datu bāzes daļa – aprakstīta ierastās relāciju datu bāzs struktūrs, ontoloģijas daļa - aprakstīta ontoloģija,kas atbilst konkrētajai relāciju datubāzei un visa pārējā metmodeļa daļa nodrošina atbilstību starp RDB un RDF.

Vienkāršā variantā, ar ko autore arī ir sākusi - metamodelī ir, piemēram, tabulu un kolonnu aprakstu("Table", Column) relāciju datu bāzes "galā" un OWLOntology, OWLClass, utt, OWL, "galā", kā arī ClassMap,ko nodēvēsim par galveno sasaistes nodrošin'tāju starp relāciju datu bāzi un tai atbilstošu ontoloģiju.

Sīkāk tiks apskatītas tikai tās RDB2OWL klases, kuras līdzšinējā darbā autorei vajadzēja izmantot, vai kuras ir pietiekami "plašas", lai būtu vēlams kāds ieskats par tām.

Tabula1 : RDB2OWL apraksts

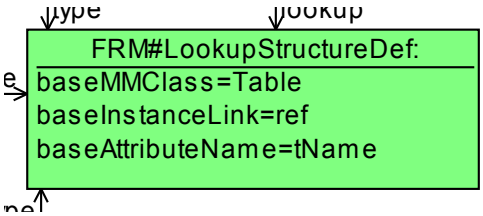
Klase	**Atribūts/Process**	**Apraksts**
OWLOntology	-	Apzīmē konkrētu ontoloģiju
OWLOntology	ontologyURI	Ontoloģijas piekļuves URI
OWLClass	-	Veido ontoloģiju – ontoloģijas klase
OWLProperty	-	Veido ontoloģiju - atribūtvērtība

ClassMap	-	Galvenā sasaiste starp RDB un RDF/OWL
Table	-	Relāciju DB tabulas apraksts
Table	Tname	String vērtība, tabulas vārds
Column	-	Relāciju DB tabulas kolonnas apraksts
Column	ColName	String vērtība, kolonnas nosaukums
TableRef	-	Satur norādi uz aktuālo “Table”
BinaryFilterItem	-	Nodrošina filtrēšanu ar “<, >, ==, <= utt”

4.RDB2OWL izteiksmju modeļa redaktors

Bakalaura darba ietvaros tika izstrādāts metamodelis pēc kura instancēm redaktors spēj ģenerēt gatavas lietojumformas, ar kuru tālāk ir iespējams veidot RDB2OWL izteiksmes. Redaktors tika rakstīts TDA vidē izmantojot Lua un papildinājumu lQuery.

Konkrētais redaktors izmanto "savu" metamodeli un tiek sasaistīts ar RDB2OWL metamodeli, lai gan redaktors tiek veidots tā, lai viegli būtu sasaistāms vajadzības gadījumā arī ar citiem metamodeļiem. Faktiski šim redaktoram ir 2 daļas – Universālā daļa un RDB2OWL izteiksmu veidošanas daļa. Šim nolūkam klasēm,kas apraksta formas izskatu ir paredzētas arī procedūras,kas tiek rakstītas atsevišķā failā. Ne metamodelī, ne kodā netiek norādītas konkrētas sasaiste ar RDB2OWL metamodeli(Atskaitos prefisku R2B#, kas tiek izmantots kā globāls mainīgais un kuru var viegli mainīt). Norādes uz RDB2OWL metamodeli vienkāršos gadījumos ir atrodamas instanšu atribūtos



Drawing 7: FRM# instances piemērs

Šajā gadījumā programmas kodā tiek nolasītas attiecīgās vērtības, lai piekļutu vajadzīgakai RDB2OWL klasei. Protams, ir arī gadījumi,kad viss nav tik viegli un vērtību iegūšanai ir sarežģītāks algoritms,kas prasa ņemt vērā konkrēti RDB2OWL struktūru. Šādiem gadījumiem arī tiek izmantotas procedūras. Līdz ar to redaktora universālā daļa paliek "neskarta", tam tikai tiek piesaistītas konkrētas procedūras.

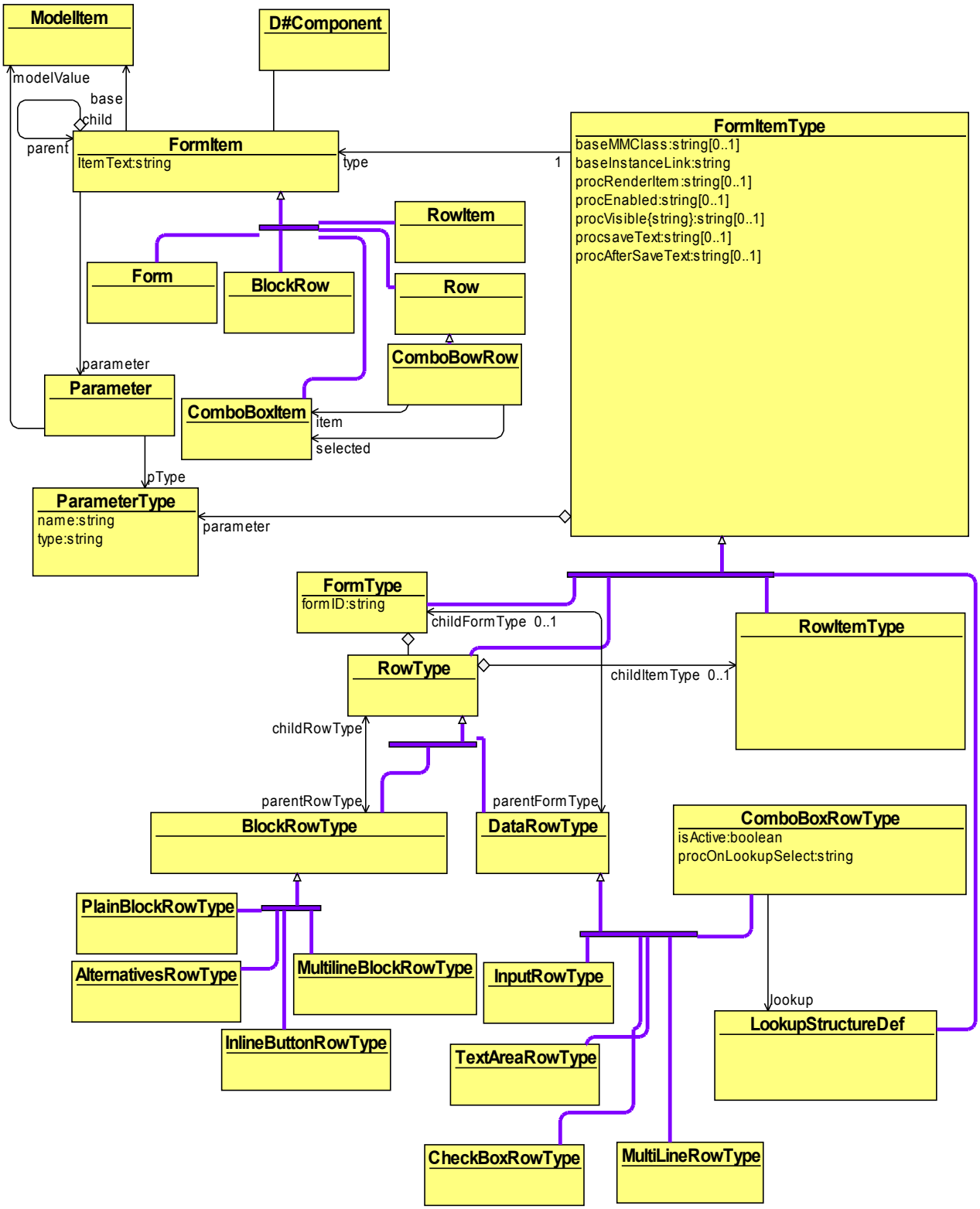
4.1."FRM" Metamodelis

Šis ir metamodelis(TDA vidē izveidotas ar prefiksu FRM#) uz kura tiek balstīts viss redaktors.

Konkrētajā metamodelī ir atsevišķi izveidotas klases tām instancēm,kas tiek izveidotas ar

lQuery paīdzību(nākotnē iespējams ar konfiguratoru) – formas definēšanas instances un "run-time" instancēm nepieciešamās klases, kuras tiek izvedotas pēc nepieciešamības pēc programmas palaišanas. Šim metamodelim ir arī sasiate konkrētajā gadījumā ar RDB2OWL izveidoto metamodeli(ModelItem šeit ir kā virsklase visam RDB2OWL metamodelim) - programmas darbināšanas laikā notiek sasaiste ar RDB2OWL instancēm.

Tāpat ir sasaiste arī ar D#Component, kas nodrošina vajadzīgās informācijas vizuālu attēlojumu TDA vidē.



Drawing 8: Redaktora metamodelis

4.2.Metamodeļa apraksts

FormItemType – virsklase visām klasēm,kuru instances izmantojamas attēlojumformas aprakstīšanai, saistās ar meatmodeļa run-time daļu.

- baseMMClass – RDB2OWL metamodelī esošās klases nosaukums
- baseInstanceLink – links pa kuru atrast attiecīgo RDB2OWL klasi
- procRenderItem – procedūras nosaukums, kuru izsaukt jaunu instnču ģenerēšanai
- procEnabled – nosacījumi, lai attiecīgais vienums būtu aktīvs, to varētu "lietot"
- procVisible - nosacījumi, lai attiecīgais vienums būtu redzams, respektīvi, tiktu izveidota viņam atbilstoša D#Component
- procSaveText – procedūra tiek izsaukta tekstuālām komponentēm, kad notiek darbība(Add/Edit/delete) "īpašos gadījumos", piemēram, ievēloties comboBox vērtību,ja konkrētajai vērtībai nav atbilstošas instaces RDB2OWL metamodelī.
- ProcAfterSaveText – papildina procSaveText procedūru, lai veiktu darbības pēc procSaveText izpildes

Tālāk tiek apskatītas klases, kuru instances tiek izvidotas "run-time" izpildes laikā

FormItem

- itemText – Komponentēm, kurām ir vizuāli attēlojama vērtība

Form – FormItem apakšklase

Row - FormItem apakšklase

RowItem - FormItem apakšklase

BlockRow - FormItem apakšklase

ComboBoxRow - Row apakšklase

ComboBoxItem - FormItem apakšklase

Parameter – sarežģītākos gadījumos vara neiztikt ar "itemText", tad arī piesaistam Parameter klasi

ParameterType – Iepriekšminētā parametra tips

Tālāk apskatām klases, kuru instances tiek izveidotas pirms programmas palaišanas – lai definētu attēlojumformu.

FormType – sastāv no RowType tipa vērtībām, tā teikt – apzīmē visu formu, kas jāttēlo

RowType – atsevišķa formas komponente, satur RowItemType tipa vērtības(atbilst D#HorizontalBox, D#VerticalBox utt komponentē,)

Tālāk ir klases,kas atbilst dažādiem blokiem(apakšklases RowType) :

BlockRowType – Bloks,kas ietver sevī citas komponentes

PlainBlockRowType – BlockRowType apakšklase

AlternativesBlockRowType - BlockRowType apakšklase

InlineButtonRowType - BlockRowType apakšklase

MultiLineBlockRowType - BlockRowType apakšklase

RowItemType – veido RowType

ComboBoxRowType

- isActive – bool tipa vērtība, nosaka vai komponente ir aktīva
- procOnLookupSelects – apraksta izsaucamo procedūtu,kad notiek vērtības izvēle ComboBox tipa komponentē

DataRowType – atbilst dažādām datu attēlošanas komponentēm – teksta lodziņiem utt

InputRowType - Atbilst "Input" tipa laukam.

TextAreaRowType - Atbilst “TextArea” tipa laukam.

CheckBoxRowType - Atbilst “CheckBox” tipa laukam.

MultilineRowType - Atbilst “MultilineTextBox” tipa laukam.

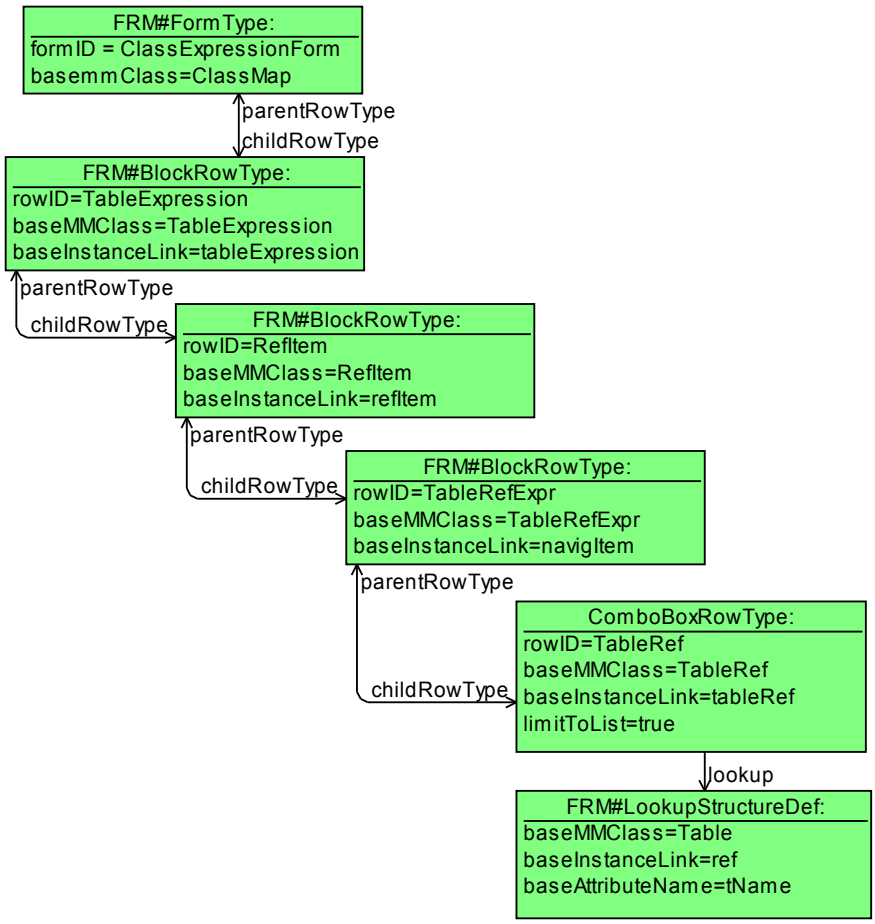
LookupStructureDef – Satur informāciju par vērtībām,kuras jāpiemeklē ComboBox tipa komponentei

Tālākās klases apzīmē FRM metamodeļa sasaiti ar citiem lielākiem metamodeļiem

ModelItem – Virsklase visām RDB2OWL klasēm, norāda sasaisti ar RDB2OWL metamodeli

D#Component – Norāda sasaisti ar visām D# komponentēm, nodrošinot vizuālu attēlojumu TDA vidē.

4.3. Testa piemēra apstrāde



Drawing 9: Redaktoram "apstrādājamā" isntance

Tā kā paredzētais konfigurators nav pabeigts, tad visas testa instances tika veidotas izmantojot lQuery. Sākotnēji ar lQuery(normālā gadījumā ar konfiguratoru) izveido "vēlamo formu" jeb saveido FormItem instasnces, pēc kurām būtu iespējams ģenerēt formu. Attēlā ir redzams piemērs instancei, pēc kuras ir iespējams no RDB2OWL metamodeļa daļas atrast visas "Table" instanču "tName" atribūtus – citiem vārdiem savācam tabulu nosaukumus relāciju datu bāzē.

Šī instance principā liek mums RDB2OWL metamodeļa daļā meklē "ClassMap" instancei pieasistīto "TableExpression" ar saiti "tableExpression", tālāk pa saiti"refItem" atrod "RefItem" instanci, tālāk ar saiti "navigItem" uz "TableRefExpr". Šī jau ir ComboBoxRowType instance, kas nozīmē,ka būs jāveido ComboBox. Tā vērtības meklējam tālāk skatoties lookup linka galā esošajā "LookupStructureDef" instancē. Šajā vienkāršajā gadījumā iztiekam bez procedūrām, jo meklējam vienas konkrētas klases, noteiktu atribūta vērtību, respektīvi "Table" klasei atribūtus "tName".

Tālāk būs apskatīts arī sarežģītāks piemērs, kas prasīs procedūras pielietojumu.

Algoritms,kas izveido formu no šīs instances

Palaižot programmu, tiek meklētas visas FRM#FormType instances, tā tiek apstrādāta un tālāk rekursīvā veidā tiek apstrādātas visas tai "piederošās" instances. Pa linku "child" atrodam 1. Row tipa instanci un rekursīvi sameklējam visas pa zaru "child"(comboox arī lookup) atrodamās instances.

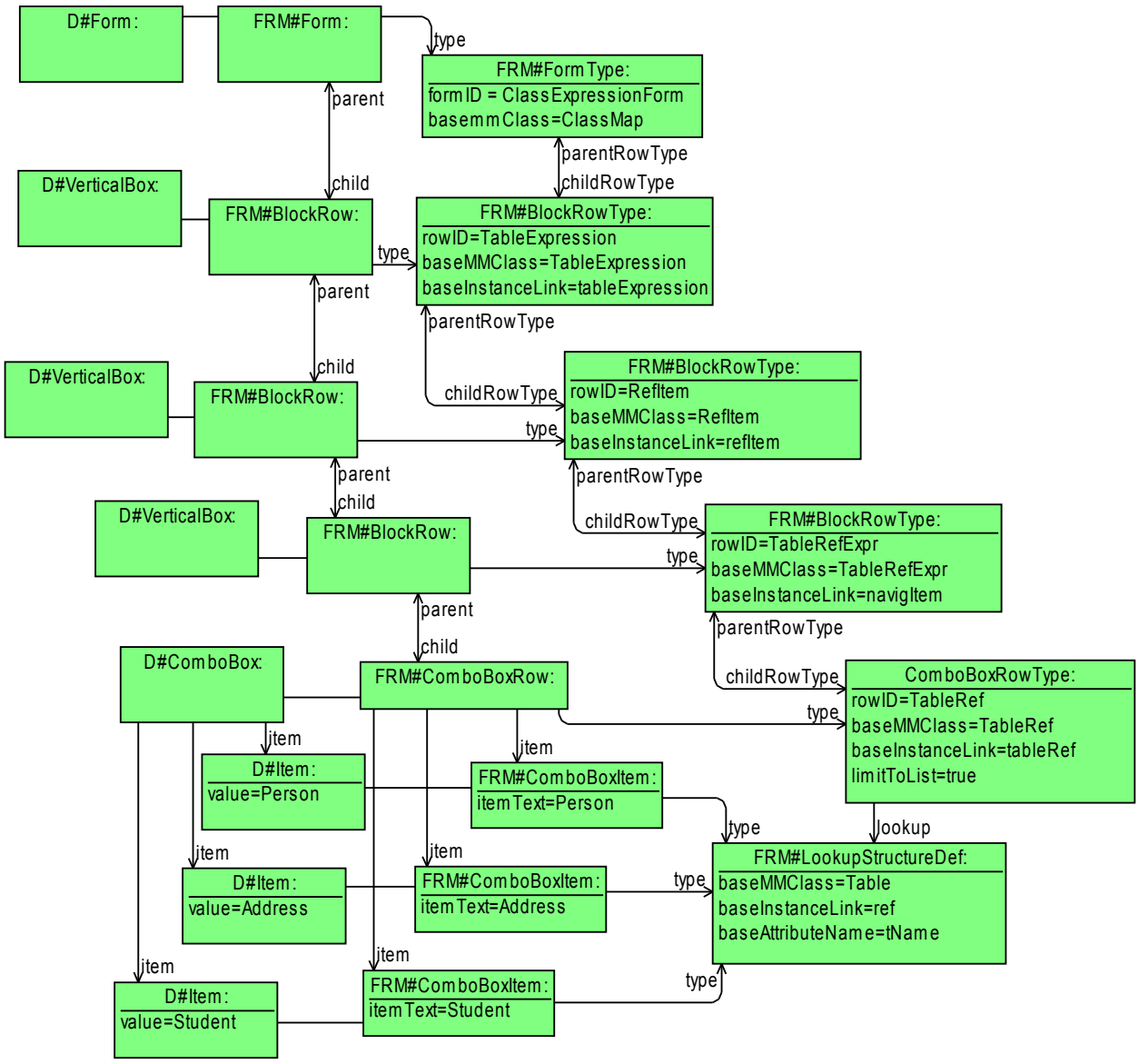
Pirmais, ko redaktors dara, esošo instaču kopumu "apaudzē" ar jaunām instancēm. Pirmkārt tiek izveidotas "run-time" instances(FRM#), pēc tam arī Instances,kas nepieciešamas formas vizuālajam attēlojumam(D#Component), piemēram FRM#FormType piesaista FRM#Form un tam piesaista D#Form.

Tāpat arī "run-time" FRM# instances tiek savā starpā savienotas ar parent<-->child attiecību.

RDB2OWL izteiksmes izveidošana

Tātad galvenais,kas jāspēj šim redaktoram ir saveidot RDB2OWL izteiksmi, ko pagaidām ļoti ļoti vienkāršām izteiksmēm tas arī spēj. Pagaidām algoritms nav ļoti universāls, ir rakstīts pamatā balstoties uz konkrētajiem testa piemēriem(Tabulas un kolonnas). Tas darbojas konkrētā gadījumā, taču nav testēts vispārīgākā un plašākā gadījumā.

Tāpat šis redaktors tuvākajā laikā atbalstīs arī iespēju rediģēt jau izveidotu izteiksmi – t.i., uz izmaiņām redaktors reaģēs veicot izmaiņas instancēs. Sākumā izteiksmes rediģēšana nebūs "patvaļīga", arī tam tiks izveidota atsevišķa saskarne, taču ideālā variantā jāizveido parseris,kas spēj lietotāja "ar roku" ievadītu izteiksmi "sadalīt" pa instancēm metamodeļos.



Drawing 10: Ar redaktora piezīmētajām instancēm

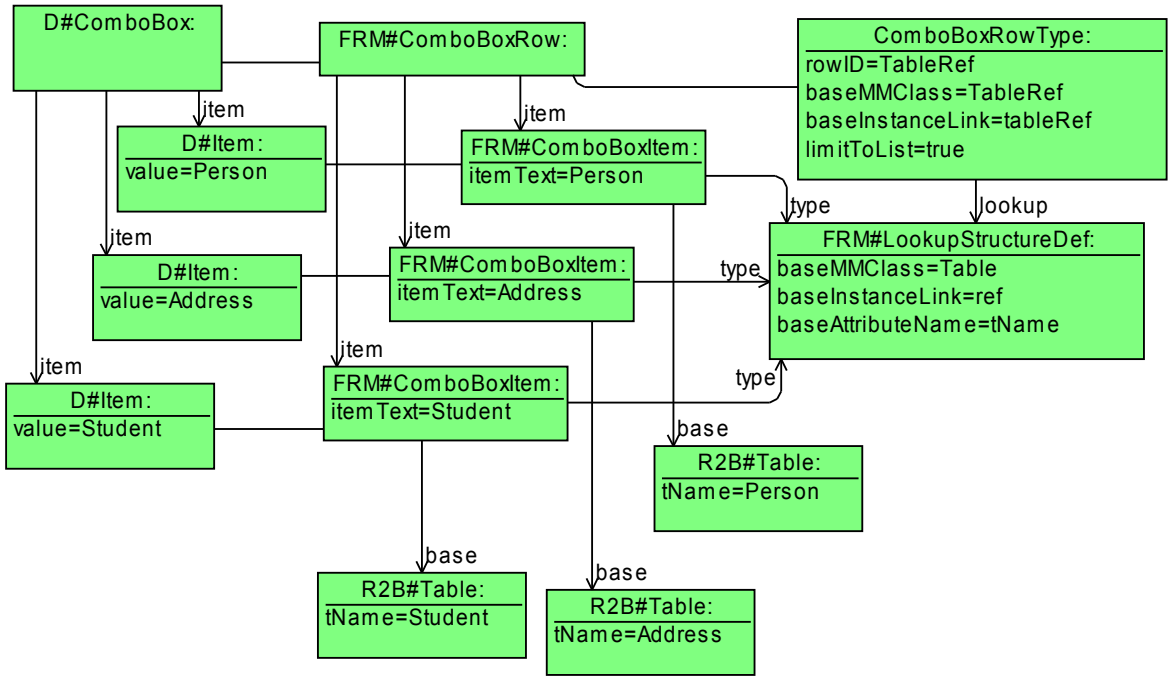
Šajā brīdī ir vērts pieminēt,ka izteiksmes veidošanas ziņā redaktoram ir 2 daļas – vienkāršotā(simple) un papildinātā(advanced). Šajā brīdī mēs darbojamies tikai "vienkāršotajā režīmā", taču arī vienkāršās izteiksmes ir iespējams attēlot papildinātajā daļā.

Šajā konkrētajā piemērā RDB2OWL metamodelī ir jau izveidotas 3 "Table" instaces –

Person, Address un Student. Redaktors attiecīgi tad izveido 3 FRM# un trīs D# instances katrai šai vērtībai

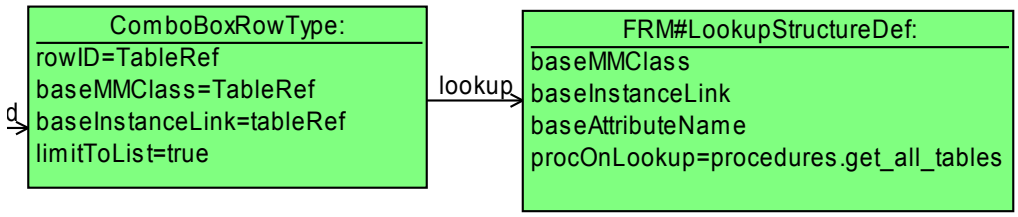
RDB2OWL instanču piesaistīšana ar linku “base” var notikt uzreiz, vai arī pēc kādu darbību izdarīšanas. Autores piemērā ModelItem atlasītajām tabulām tiek piesaistītas tikai pēc noklikšķināšanas uz ComboBox.

Tāpat pēc CoboBox vērtības izvēles, redaktors nodrošina “pareizu” ComboBoz uzvedību, atzīmējot kura vērtība šajā brīdī tiek attēlota(novelk linku “selected” iz vajadzīgo FRM#VomboBoxItem).



Drawing 11: ar "base" instancēm

Apskatīsim ļoti līdzīgu, bet nedaudz sarežģītāku piemēru, kur mēs vēlamies atlasīt ne tikai Table, bet arī “OWLClass” instances “localName” vērtības. Tas mums prasītu norādīt instancē “LookUpStructureDef” divus baseMMClass, baseInstanceLink un divus baseAttributeName. Šādas iespējas nav. Vajadzības gadījumā, protams, varētu nodrošināt iespēju šiem atribūtuem padot vairākas vērtības, tacu ir saprtoams,ka būs gadījumi,kad ar to vien nepietiks, tādēļ šajā brīdī jau nolemjam ieviest procedūras.



Drawing 12: Ar procedūru

Šajā gadījumā baseMMClass,baseInstanceLink un baseAttributeName atstājam tukšu, taču padodam vērtību procOnLookUp. Pagaidām ir ieviests arī šāds atribūts,kas izsauc procedūru get_all_tables no faila procedures.lua. Šī procedūra tikai atgriež masīvu ar klasēm un atribūtiem,kurus nepieciešams atlasīt.

```
function get_all_tables(value)
if (value == 'all_tables') then
          -- Get table values from Tables.tName utt
               local all_classes = {{"Table", "tName"},
                                         {"OWLClass", "localName"}
                                   }

               return all_classes
     end
end
```

Vai tiešām ir vajadzība pēc procedūrām, kas tikai atgriež masīvu ar datiem, ir diskutabls jautājums. Taču "LookuIpStructureDef" klasei ir paredzēts arī atribūts "procBuildItems", kurā norādītā procedūra drīkst veikt patvaļīgus pieprasījumus, tā teikt – patvaļīga procedūra. Jautājums par atribūtiem procedūrām ir "atvērts".

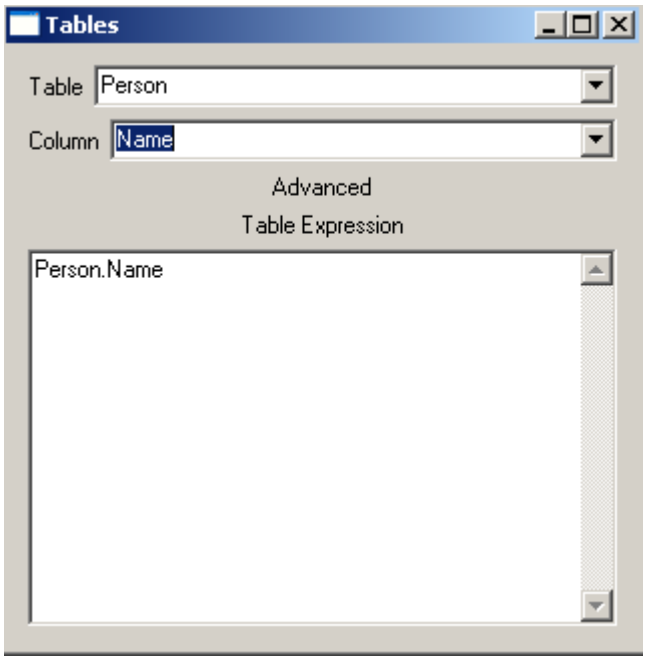
Kā tas izskatās redaktorā

Instances un algoritmi kā tās tiek apstrādātas ir viena lieta, bet kā tas vizuāli izskatās? Konkrētajā piemērā(Drawing 13) tiek savākti visi Table(atbilstoši pirmajam piemēram) ComboBox

vērtības komponentē pretī nosaukumam "Table"(Pagaidām izmanto RDB2OWL klases nosaukumu), līdzīgi zemāk ir atlasītas visas kolonnas,kas pieder tabulai "Person", šajā gadījumā tikai viena.

Ar šīm divām kompoonentēm konkrētajā gadījumā beidzas "Simple" daļa un tālāk jau ir paplašinātā daļa. Jebkuru vienkāršo izteiksmi ir iespējams attēlot arī paplašinātajā daļā,kas arī tiek darīts "Table Expression" lodziņā. Pagaidām tas neizskatās visai interesanti, taču "apakšā" stāv krietni lielāks mehānisms, ko nupat apskatījām un šāds ir vizuālais rezultāts iepriekš demonstrētjām instancēm.

Izstrādes procesā ir iespēja arī rediģēt uzreiz lauku Table Expression(lai izmaiņas tiktu ņemtas vērā), kā arī laukam "Column" norādīt filtrēšanas vērtību.



Drawing 13: Redaktora piemērs

Šī lietojumforma parāda mums no Rdb2OWL modeļa savāktos datus un ataino RDB2OWL izteiksmi, šeit mēs visuāli redzam abas - "Simple" un "Advanced" daļas. Lielākā daļa RDB2OWL izteiksmju "iekļausies" vienkāršajā daļā.

4.4.Lua programmēšanas valoda

Lai izstrādātu RDB2OWL izteiksmju redaktoru, tika izmantota programmēšanas valoda Lua un tās papildinājums lQuery.

Lua nav ļoti populāra, taču tā ir ātra, dinamiski veidota skriptēšanas valoda. Tajā nav daudz datu tipu un maiinīgajiem pirms lietošanas nav jānorāda datu tips. Masīvu un saraktus Lua apraksta izmantojot datu tipi "table",kas darbojas kā asociatīvais masīvs.

Tāpat Lua atbalsta arī objektorientēto pieeju. Šī kursa darba ietavros Lua tika izmantos arī papildinājums lQuery.[9]

4.4.1.lQuery papildinājums

lQuery ir bibliotēka, kas domāta darbam ar repositoriju objektu kolekcijām. Tas sastāv no divām daļām : metožu kopas un selektoru izteiksmju kopas – filtrēšanai un navigēšanai pa metamodeļa klašu instancēm. Lquery nav nekāda veida kontroles struktūras vai metožu definīcijas konstrukcijas, jo tā ir implementāta Lua un izmanto Lua kotroles struktūras un metožu definīcijas konstrukcijas.[9]

lQuery piemēri

Metamodeļa papildināšana:

```
lQuery.model.add_class("FRM#FormItemType")
    lQuery.model.add_property("FRM#FormItemType", "baseMMClass")
```

Ar šo kodu tiek izveido jauna metamodeļa klase "FRM#FormItemType" un tai atribūts "baseMMClass"

Instances izveidošana:

```
local comboBox1 = lQuery.create("FRM#ComboBoxRowType", {
                                    rowID = "TableRef",
                                    baseMMClass = "TableRef",
```

```
                                    baseInstanceLink = "tbleRef"
                                    })
```

Filtrēšana:

local this_instance=lQuery(“D#TextArea”)has(/compartType[id='Attributes'])")

tiek atlasīta instance, kas atbilst nosacījumiem :

- Tā ir D#TextArea klases instance
- Tais ir saite “compratType” ar atribūtu id == 'attributes'

Navigēšana:

līdzīgi kā filtrēšana

local this_instance = parent_instance:find("/child")

this_instance tiek atrasta ejot pa “child” saiti no instancs parent_instance

5.Rezultāti

Šī bakalaura darba rezultātā autore ir ieguvusi ieskatu RDB2OWL uzbūvē, RDB2OWL izteksmēs, nostiprinājusi zināšanas par semantisko tīmekli tā attīstības gaitu un pastāvošajām problēmām, jaunākajiem lQuery papildinājumiem, kā arī OWLGrEd uzbūvi un izpētījusi TDA. Gan redakktora metamodelis, gan pats redaktors atrodas izstrādes procesā, taču ir "ielikti pamati" pilnīgai redaktora izstrādei.

5.1.Izveidotais redaktors

Kā "taiustāms" rezultāts pagaidām ir iegūts redaktors, kurš pieļauj vienkāršo izteiksmju definēšanu RD2OWL sintaksē. Ar vienkāršiem testa piemēram nav atklāti nekādas būtiskas nepilnības, taču sarežģītākus piemērus pagaidam redaktors nespēj attēlot.

5.2.Problēmas un nākotnes ieceres

Darba gaitā autore saskārās ar tādu problēmu,ka lQuery navigācijas iespējas piedāvā rakstīt ļoti garus nosacījumos. Tas rada vēlmi saīsināt lQuery sintaksi, lai tā būtu vieglāk pārskatāma arī cilvēkam. Tāpat iespējams nākotnē būs nepieciešama lQuery papildināšana, lai varētu veiksmīgi un precīzi attēlot RDB2OWL metamodeli OWLGrEd rīkā.

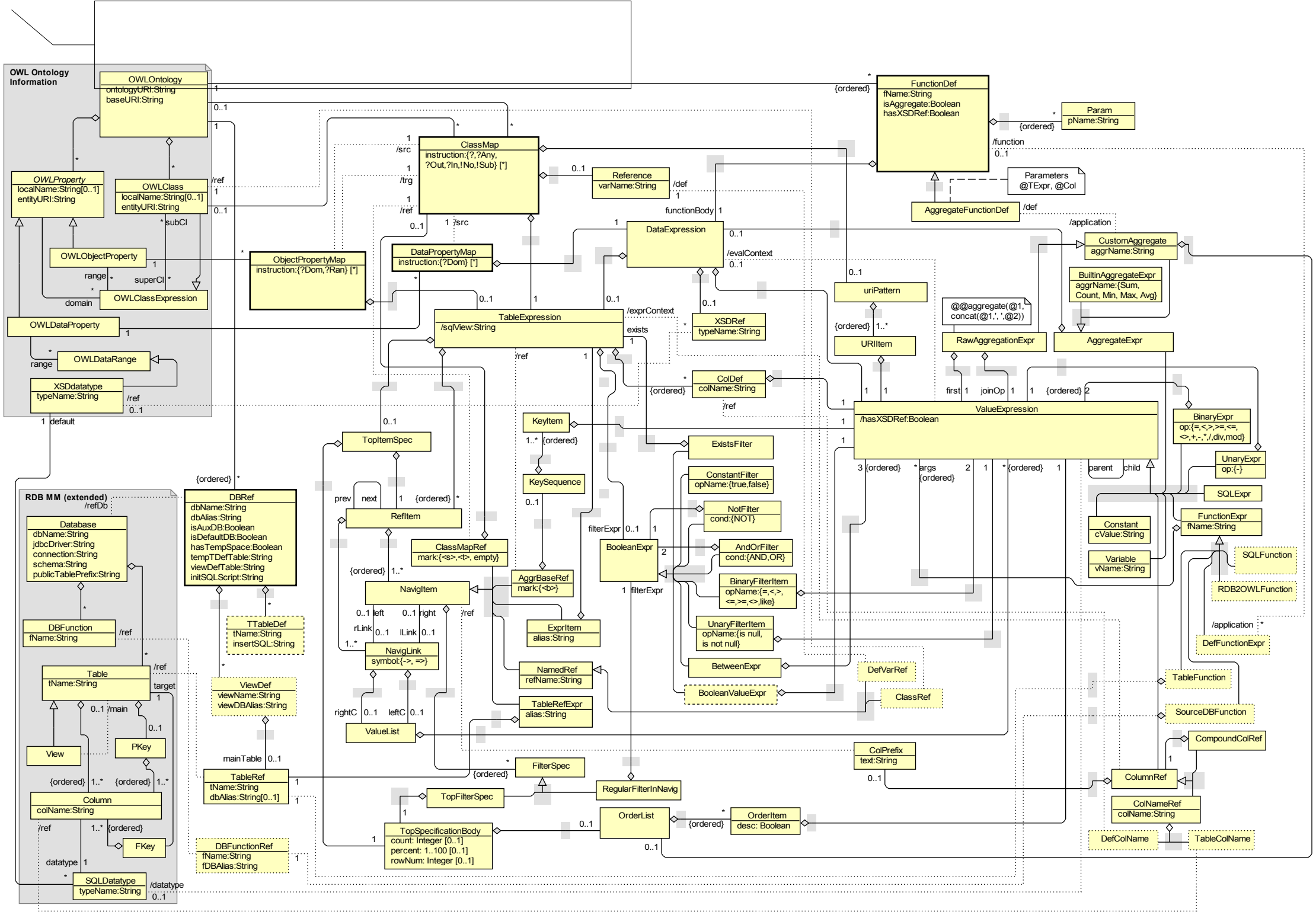
Ir paredzēts un cerība pabeigt "vienkāršas" RDB2OWL isteiksmes veidošanu, kā arī būtu nepieciešams parseris, kurš ļautu cilvēkam ievadīt RDB2OWL izteiksmi un tā tiktu "sadalīta" pa nepieciešamajām instancēm balsltoties uz metamodeli.

5.3.Secinājumi

Ir ielikts pamats RDB2OWL izteiksmju redaktoram TDA vidē. OWLGrEd ir uzskatāms par piemērotu redaktoru šāda rīka izstrādei. Lai arī darba gaitā autore saskārās ar dažām problēmām un nepilnībām, atrastās problēms jau tika prognozētas pirms darba uzsākšanas, līdz ar to būtisku šķēršļu darba turpināšanai nav, ar piebildi,ka kopumā darbs līdz šim nav ritējis raiti un turpmāk ir jāvelta vairāk pūļu, lai šis redaktors tiktu pabeigts.

6.Izmantotā literatūra

1. **RDB2OWL: a RDB-to-RDF/OWL Mapping Specification Language**, Čerāns Kārlis, Būmans Kārlis
2. **OWLGrEd: a UML Style Graphical Editor for OWL**, Bārzdiņš Jānis, Čerāns Kārlis, Liepiņš Renārs, Sproģis Artūrs
3. **OWL Web Ontology Language [tiešaiste].Pieejams internetā** http://www.w3.org/TR/owl-features/
4. "**Lietotāja saskarnes definēšana RDF datubāzēm**", kursa darbs, Diteriha Gerda, 2011
5. **Terminoloģijas vārdnīca.** [tiešsaiste]. Pieejams internetā: http://termini.letonika.lv
6. **The Transformation-Driven Architecture,** Bārzdiņš Jānis, Kozlovičs Sergejs, Rencis Edgars
7. **"The Configurator in DSL Tool Building"**, Sproģis Artūrs
8. **"A Dialog Engine Metamodel for the Transformation-Driven Architecture"**, Kozlovičs Sergejs
9. **"lQuery presentation"** Liepiņš Renārs


OWL Ontology Information
OWLOntology
ontologyURI:String
baseURI:String
OWLProperty
localName:String[0..1]
entityURI:String
OWLClass
localName:String[0..1]
entityURI:String
subCl
OWLObjectProperty
range
superCl
domain
OWLClassExpression
OWLDataProperty
range
OWLDataRange
XSDdatatype
typeName:String
default
ClassMap
instruction:{?,?Any,
?Out,?In,!No,!Sub} [*]
/src
/trg
/ref
ObjectPropertyMap
instruction:{?Dom,?Ran} [*]
DataPropertyMap
instruction:{?Dom} [*]
TableExpression
/sqlView:String
/exprContext
exists
Reference
varName:String
/def
functionBody
DataExpression
/evalContext
XSDRef
typeName:String
ColDef
colName:String
KeyItem
KeySequence
TopItemSpec
RefItem
prev
next
ClassMapRef
mark:{<s>,<t>, empty}
NavigItem
left
right
rLink
lLink
NavigLink
symbol:{->, =>}
AggrBaseRef
mark:{<b>}
ExprItem
alias:String
NamedRef
refName:String
TableRefExpr
alias:String
ValueList
rightC
leftC
FilterSpec
RegularFilterInNavig
TopFilterSpec
TopSpecificationBody
count: Integer [0..1]
percent: 1..100 [0..1]
rowNum: Integer [0..1]
OrderList
OrderItem
desc: Boolean
ExistsFilter
ConstantFilter
opName:{true,false}
NotFilter
cond:{NOT}
BooleanExpr
AndOrFilter
cond:{AND,OR}
BinaryFilterItem
opName:{=,<,>,
<=,>=,<>,like}
UnaryFilterItem
opName:{is null,
is not null}
BetweenExpr
BooleanValueExpr
filterExpr
DefVarRef
ClassRef
uriPattern
URIItem
@@aggregate(@1,
concat(@1,', ',@2))
RawAggregationExpr
AggregateExpr
CustomAggregate
aggrName:String
BuiltinAggregateExpr
aggrName:{Sum,
Count, Min, Max, Avg}
ValueExpression
/hasXSDRef:Boolean
first
joinOp
args
parent
child
BinaryExpr
op:{=,<,>,>=,<=,
<>,+,-,*,/,div,mod}
UnaryExpr
op:{-}
SQLExpr
Constant
cValue:String
Variable
vName:String
FunctionExpr
fName:String
SQLFunction
RDB2OWLFunction
DefFunctionExpr
TableFunction
SourceDBFunction
CompoundColRef
ColumnRef
ColNameRef
colName:String
DefColName
TableColName
ColPrefix
text:String
FunctionDef
fName:String
isAggregate:Boolean
hasXSDRef:Boolean
Param
pName:String
Parameters
@TExpr, @Col
AggregateFunctionDef
/function
/application
RDB MM (extended)
/refDb
DBRef
dbName:String
dbAlias:String
isAuxDB:Boolean
isDefaultDB:Boolean
hasTempSpace:Boolean
tempTDefTable:String
viewDefTable:String
initSQLScript:String
Database
dbName:String
jdbcDriver:String
connection:String
schema:String
publicTablePrefix:String
DBFunction
fName:String
Table
tName:String
target
main
View
PKey
Column
colName:String
FKey
datatype
SQLDatatype
typeName:String
/datatype
TTableDef
tName:String
insertSQL:String
ViewDef
viewName:String
viewDBAlias:String
TableRef
tName:String
dbAlias:String[0..1]
mainTable
DBFunctionRef
fName:String
fDBAlias:String

Bakalaura darbs
„RDB2OWL izteiksmju modeļa redaktors TDA vidē”

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

Autore: Gerda Diteriha _____
(Autores paraksts)

Ar savu parakstu apliecinu, ka esmu lasījis augšminēto bakalaura darbu un atzīstu to par piemērotu/nepiemērotu (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu bakalaura studiju programmas gala pārbaudījuma komisijas sēdē.

Darba vadītājs: Kārlis Čerāns _____
(Vadītāja paraksts)

Darbs iesniegts Datorikas Fakultātē _____
(Iesniegšanas datums)

Ar savu parakstu apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.

Metodiķe: Ārija Sproģe _____
(Metodiķes paraksts)

Recenzents: Audris Kalniņš _____
(Recenzenta paraksts)

Darbs aizstāvēts bakalaura darbu gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____, vērtējums _____
(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____
(Sekretāra paraksts)