FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Západočeská Univerzita v Plzni

Fakulta Aplikovaných Věd

Databázové systémy
a metody zprac.inf.2

Semestrální práce
Přejmenování RDF instance dle svých vlastností

David Bohmann - A17N0064P
bohmannd@gmail.com

10. prosince 2019

Welcome to RDF Node renaming program

Please select properties that you want to include in node name.

For each property there is an example in brackets

Type	Original ID	Properties	New name example
_links	☐ Include original ID	self self:href self:type type	[base]-self-118ddd03-2c70-4b6c-806b-8faf4c824c76-type-_links
character_link	☑ Include original ID	battle_tag id key key:href	[base]-id-4937072-key:href-4937072?namespace=prod-key:type-key-type-character_link-[id]
clan_link	☑ Include original ID	clan_name clan_tag decal_url icon_url	[base]-clan_tag-c5c!-[id]
key	☐ Include original ID	href type	[base]
ladder	☑ Include original ID	team:ties team:type team:wins type	[base]-team-count:199-type-ladder-[id]
league	☐ Include original ID	league_key:season_id league_key:team_type league_key:type type	[base]-league_key:season_id-33-league_key:team_type-0-type-league
league_key	☐ Include original ID	league_id queue_id season_id team_type	[base]
		id	

Obsah

1 Zadání

Pro semestrální práci jsem si zvolil zadání číslo 4(a).

Současný stav:

Na bázi ontologie definovány součástí kompozitního klíče anotačními vlastnostmi. Není podpora využívání zanořených vlastností. Skript pak neinteraktivně přejmenuje uzly. Chci zkusit změnit paradigma - řešit situaci na bázi dat nikoliv ontologie.

Úkol:

1. Vytvořit interaktivní aplikaci, která na vstupu dostane RDF soubor a na konci vrátí upravený RDF soubor.
2. Aplikace analýzou vstupného souboru zjistí, jaké vlastnosti (včetně zanořených) se běžně vztahují k jednotlivým RDF třídám (rdf:type).
3. (a) Formulářovým rozhraním umožnit volbu RDF třídy -¿ zobrazí se roletové menu s běžnými vlastnosti -¿ uživatel si vybere kombinaci vlastností jako kompozitní primární klíč -¿ přejmenování
   (b) Skript se pokusí automaticky navrhnout vhodné přejmenování

Data:
[A] `http://home.zcu.cz/~kryl/DBM2/data/20171009-eu-6.ttl`
[B] `http://home.zcu.cz/~kryl/DBM2/data/ibds.ttl`

2 Analýza

2.1 RDF

RDF (Resoure Description Format) je obecný rámec dat, která popisují zdrojový dokument tak, že je jeho popis čitelný jak lidsky, tak strojově. RDF je standardizovaný formát, který umožňuje vyjadřovat popisné informace o WWW zdrojích. Zároveň je to i formát grafový, takže veškerá data v RDF lze zapsat pomocí grafu s orientovanými hranami, které je možné zapsat jako množinu trojic.

Zdroj, který lze popsat pomocí RDF, musí být jednoznačně identifikovatelný pomocí URI.

RDF každému zdroji přiřazuje výraz ve tvaru *subjekt - predikát - objekt*. Zde *subjekt* určuje, o jaký zdroj se jedná, *predikát* nějakou jeho vlastnost a *objekt* hodnotu této vlastnosti.

2.2 Apache Jena

Apache Jena je knihovna, která (kromě mnoha dalších využití) umožňuje v jazyce Java pracovat s RDF. Jena umí vytvořit model dat, ve kterém spravuje následující informace využité v semestrální práci:

- `resource` - Subjekt a všechny informace o něm
- `statement` - Výraz pro daný subjekt

- `predicate` - Predikát v daném výrazu
- `statement` - Objekt v daném výrazu

3 Implementace

Program je implementován v jazyce Java jako webová aplikace postavená na Spring Boot. Pro zobrazování dynamického obsahu webových stránek je použit šablonovací nástroj Thymeleaf. Ze dvou možností zadání jsem zvolil tu, ve které má uživatel možnost zvolit nové názvy.

Implementaci lze rozdělit na tři hlavní části.

- Zpracování souboru
- Výběr vlastností pro přejmenování
- Přejmenování

Pro vytvoření celé aplikace jsme se snažil dodržovat zásady Spring MVC frameworku (s jistými omezeními, například aplikace vůbec nepoužívá databázi). Pro předávání informací uživateli slouží třída `RdfController`, o logiku celého programu se stará `RdfService`.

3.1 Zpracování souboru

Aplikace na začátku přijímá od uživatele soubor s RDF modelem v následujících formátech

- TURTLE (contentType „application/x-turtle“, přípona `.ttl`)
- RDF/XML (contentType „application/xml“, přípona `.xml`)
- N-TRIPLE (contentType „application/n-triples“, přípona `.nt`)

Apache Jena z daného souboru vytvoří model reprezentující kompletní graf všech trojic v RDF.

Program následně projde všechny subjekty v modelu a zjistí informaci o jejich typu a jednotlivé typy ukládá do objektu `RdfType`.

Může se stát, že subjekt nemá žádnou informaci o svém typu. V tom případě je pro další práci přeskočen. Naopak se může stát, že subjekt může mít více typů (subjekt může být zároveň osoba a pacient). V tom případě je vybrán pouze první typ.

Objekt `RdfType` reprezentuje daný typ subjektu. Obsahuje jméno a seznam objektů `RdfPredicate`.

Objekt `RdfPredicate` reprezentuje predikát. Obsahuje jméno a informaci o tom, zda byl zvolen pro přejmenování subjektu(boolean `selected`). Dále pro jednoznačnost obsahuje vygenerované UUID. Predikát nemusí obsahovat informaci o typu subjektu, protože je zařazen v seznamu predikátů u daného typu a porovnávání následně bude probíhat dle vygenerovaného ID.

`RdfPredicate` dále obsahuje vzorové jméno objektu (`exampleName`). Toto je použito ve chvíli, kdy si uživatel volí, jaké predikáty v novém názvu použít. GUI mu interaktivně zobrazuje, jak bude nový název vypadat, včetně vzorového jména objektu (uživatel pozná, zde se jedná o URI, nebo např. jméno či číselný údaj).

Poslední informací je boolean `multiple`. Pokud má subjekt pro daný predikát více hodnot, jsou tyto hodnoty sloučeny do jednoho predikátu s informací o počtu těchto hodnot a původní výrazy odstraněny. (Zde se jedná pouze o úpravu pro uživatelskou volbu k přejmenování, původní model je ponechán beze změn.) U nového predikátu je informace `multiple` nastavena na `true`.

Procházení seznamu výrazů daného subjektu probíhá rekurzivně do hloubky 2. Každý nalezený predikát je přidán do seznamu predikátů. Pokud je v daném výrazu objekt další uzel (nejedná se o literál), jsou i pro tento uzel přidány do původního seznamu jeho predikáty. To samé pro objekty těchto výrazů až do dané hloubky. Při vkládání názvu predikátů je jméno predikátu rodiče uvedeno jako prefix oddělený dvojtečkou.

Zde se může stát, že jeden subjekt daného typu obsahuje nějaké predikáty a druhý subjekt stejného typu obsahuje jiné predikáty. Algoritmus prochází všechny subjekty tak, aby při výběru vlastností pro přejmenování bylo možné vybrat všechny možné predikáty, ikdyby daný predikát obsahoval třeba pouze jeden subjekt. Tohle ale vede k riziku toho, že ikdyž se má subjekt přejmenovat, nemusí obsahovat žádný predikát, který byl uživatelem zvolen a není tedy nic, co by určovalo nový název. V tomto případě je ponechán původní název (vygenerované ID).

Následně `RdfController` zabalí celý seznam do DTO a předá kontrolu zpět uživateli.

3.2 Výběr vlastností pro přejmenování

V této chvíli se uživateli zobrazí tabulka s jednotlivými typy subjektů. U každého typu je zobrazen seznam predikátů v multiselect boxu. Aplikace také nabízí možnost zaškrtnout volbu pro přidání původního ID subjektu na konec nového názvu. Tuto volbu může uživatel použít, pokud nechce slučovat více subjektů do jednoho (někdy je to žádaná změna, jindy ne, přidání původního ID tedy slouží pro zachování jednoznačnosti).

Poslední sloupec slouží pro zobrazení ukázkového názvu přejmenovaného subjektu. Obsah se zde dynamicky mění na základě uživatelem zvolených možností.

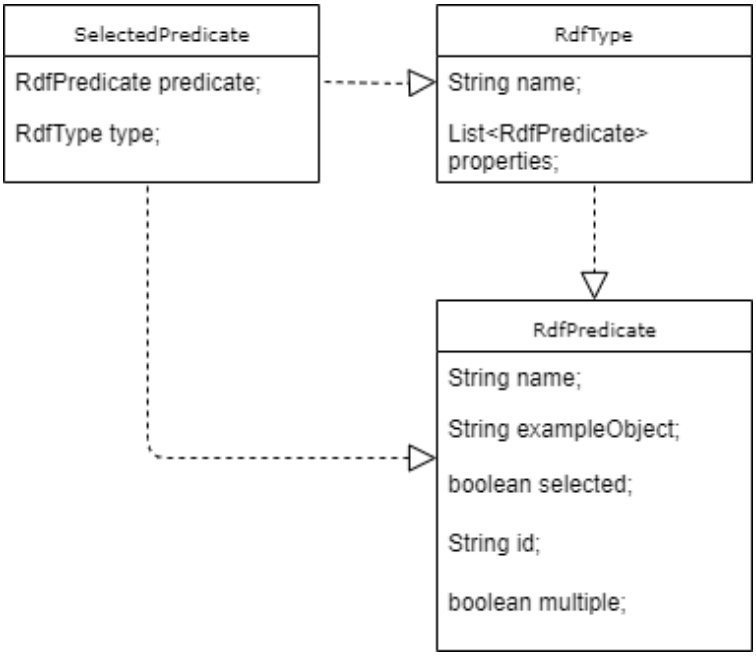
Uživatel zvolí predikáty, které chce zahrnout v novém názvu subjektu. Po vyplnění odklikne přejmenování.

3.3 Přejmenování

Následuje zřejmě nejsložitější část programu. `RdfController` obdrží od uživatele seznam ID predikátů, které má použít při přejmenování a informaci, u kterých typů má přidat původní ID.

Tyto informace jsou předány `RdfService`. V prvním kroku dojde k internímu označení predikátů k přejmenování. Program projde všechny predikáty a pokud se jejich ID shoduje s ID ze seznamu uživatelem zvolených predikátů, dojde k vytvoření objektu `SelectedPredicate`. Tento objekt udržuje informaci o `RdfType` a

`RdfPredicate` (viz Obr. 1).



Obrázek 1: UML diagram pro třídy `SelectedPredicate, RdfType, RdfPredicate`

Zároveň se u `RdfPredicate` změní hodnota `selected` na `true` (toto slouží pro případ, že dojde k návratu na stránku přejmenování, pokud uživatel při kontrole zjistí, že bude chtít provést další změny, původní změny budou již označeny).

Následně dochází k procházení subjektů v modelu. Pro každý subjekt jsou na startu nastaveny informace o přejmenování a přidání původního ID (`rename` a `appendOrigId`) na `false`. Program vytvoří `uriBase` z původního názvu subjektu a zjistí typ daného subjektu.

Pro každý predikát (i zanořené predikáty) daného subjektu dochází k porovnání s predikáty zvolenými k přejmenování a pokud jsou splněny následující dvě podmínky, dochází k přidání predikátu k novému jménu subjektu.

1. Název predikátu subjektu je stejný jako název predikátu zvoleného k přejmenování

2. Název typu subjektu je stejný jako název typu u predikátu zvoleného k přejmenování

Nové jméno může obsahovat více predikátů v názvu. Pro každý zvolený predikát je subjektu přidáno jméno predikátu a hodnota objektu (během cyklu se nový název ukládá do StringBuilderu).

Následuje ověření, zda nový název typu daného subjektu má obsahovat původní ID pro určení jednoznačnosti. Pokud ano, je hodnota `appendOrigId` na `true`.

Následně dochází k samotnému přejmenování. Program zkontroluje, zda je hodnota `rename` nastavena na `true`. Pokud ano, subjekt bude přejmenován. Při zvolení více vlastností se nový název vytváří podle abecedního pořadí predikátů. Následně se kontroluje, zda je hodnota `appendOrigId` na `true`, v tom případě se k novému

názvu tvořenému `uriBase` a všemi výrazy, které byly uživatelem zvoleny, přidá na konec původní ID.

Program využije upravené metody, kterou jsem převzal z knihovny Apache Jena, `ResourceUtils.renameResource()`, kde danému subjektu přiřadí nové jméno. Metoda odstraní všechny trojice, kde se nachází původní subjekt a nahradí je novými trojicemi s novým subjektem. Toto přináší riziko, že více subjektů může při nejednoznačnosti nového jména být sloučeno do jednoho (všechny typy aut mohou být označeny jako "auto", když při přejmenování nevyužijeme predikát "značka"nebo "SPZ"). Proto v převzaté metodě pro každý subjekt, který bude přejmenovaný, předávám do HashMap následující informace:

- key - nový název
- value - List obsahující původní názvy

Před finálním předáním nového souboru je uživatel přesměrován na stránku, kde vidí všechny změny, které se při přejmenování provádějí. Zde vidí nové vygenerované názvy subjektů, zároveň vidí, který subjekt (nebo více) se na tento název přejmenuje (pokud více než 1, proběhne ke sloučení).

Uživatel má možnost se vrátit na formulář pro přejmenování a provést další změny, nebo může změny potvrdit a v tu chvíli dojde k vygenerování nového souboru pro uživatele ke stažení. Je tedy čistě na uživateli, zda případnou nejednoznačnost názvů povolí, či ne. Stránka zároveň umožňuje stáhnout přehled změn v *.csv formátu*.

3.4 Problémy a slepé uličky, návrhy na vylepšení

Při testování na daných souborech jsem došel do stavu, kdy jsem použil všechny predikáty pro přejmenování subjektu a nový název stejně nebyl jednoznačný a došlo by ke sjednocení více subjektů do jednoho. Někde to může být žádaná vlastnost, ale někde také nemusí. Proto jsem musel zavést možnost připojit původní ID. Další vylepšení by mohlo být zbavit se nutnosti používat původní ID.

Dlouho jsem bojoval s tím, jak uživateli vrátit detailnější informace o průběhu přejmenování. Nakonec jsem převzal metodu `RenameResources` z `ResourceUtils` a přidal do ní mapování informací o přejmenovaných subjektech (původní a nový název).

Problém byl také v tom, že občas chci do třídy `RdfType` seznamu predikátů přidat více predikátů stejného typu, a někdy naopak ne, případně doplnit predikát, který v seznamu ještě není. Například pro typ `ladder` chci přidat všechny výrazy s predikátem `team` a následně je sjednotit. Ale pokud bych v grafu narazil na další objekt typu `ladder`, už tam nechci přidat další predikát `id` nebo `name`. Pokud tam ale ještě `name` od prvního subjektu není (z nějakého důvodu nebylo uvedeno), tak ho chci doplnit. Pro řešení jsem nakonec musel použít dva Listy. List, do kterého přidávám více stejných typů, a následně List, do kterého přidám typ pokaždé jen jednou. Při kontrole, zda už List daný typ obsahuje zároveň kontroluji, zda daný typ obsahuje všechny predikáty. Pokud některý predikát chybí, dodám ho do typu v Listu.

S uživatelským rozhraním jsem se pral dlouho. Nakonec jsem musel použít vlastnoručně napsané jQuery funkce pro správné chování (označení již zvolených predikátů, interaktivní zobrazování nového názvu.).

Na základě diskuse padla myšlenka pro větší uživatelský zásah v kroku přejmenování. Momentálně funguje pevně daná struktura nového názvu:
`[base]-predikát 1-objekt 1(-predikát 2...predikát n-objekt n-)([id])`,
přičemž pořadí je určeno dle abecedního pořadí názvů predikátů. Velkým vylepšením by byla možnost nechat na uživateli strukturu nového názvu, což by mohlo zahrnovat například:

- přehazování součástí nového názvu
- uživatelsky definovaný název predikátu (nemuselo by být „league:race:team_id“ ale například „id_týmu“)
- Vytvoření vlastního pravidla pro jednoznačnosti - nemuselo by se přidělovat původní ID

V momentální situaci toto vylepšení možné není, protože celý projekt byl od začátku postaven tak, aby se veškerá logika děla pouze na backendové části a frontend byl pouze pro demonstraci funkčnosti. Pro přejmenování se posílají pouze informace o zvolených predikátech pro přejmenování a na backendu se následně procházejí všechny predikáty a do nového názvu se vybírají pouze označené predikáty. V případě uživatelem navolené „kostry“ názvu by bylo třeba kompletně změnit algoritmus pro přejmenování a do kostry pouze doplňovat objekty daných predikátů. Popravdě si nedokážu moc představit, jak by tento princip mohl fungovat.

3.4.1 Komplexita

Program používá velké množství cyklů a podmínek, proto si myslím, že do dalšího vývoje by bylo vhodné se zaměřit na snížení komplexity. Momentálně na testovacích souborech program reaguje téměř ihned a z uživatelského pohledu není znát žádné zdržení.

Při zpracování souboru program prochází všechny subjekty a jejich predikáty a případně rekurzivně prochází všechny objekty (opět jako subjekty a všechny jejich predikáty) do hloubky 2. Tady je komplexita $O(n^3)$, n = počet trojic, měnit se bude podle hloubky zanoření.

Při přejmenování prochází program opět všechny subjekty a jejich predikáty, případně až do hloubky 2. Zároveň zde probíhá porovnávání se seznamem predikátů zvolených k přejmenování. Komplexita je zde $O(n^3 \times m)$, n = počet trojic, m = počet zvolených predikátů. V nejhorším případě (v každé trojici jiný predikát, všechny predikáty zvolené k přejmenování $\rightarrow$ m = n) může být komplexita až $O(n^4)$. Opět míra komplexity se bude měnit dle hloubky zanoření.

4 Uživatelská příručka

Po spuštění aplikace je uživatel vyžádán k vložení souboru. Aplikace přijímá soubory ve formátech `.ttl`, `.nt` a `.xml`. Po vložení souboru lze přistoupit k výběru vlastností (predikátů), které budou použity u všech subjektů daného typu.

V tomto kroku je uživateli zobrazena tabulka s jednotlivými typy subjektů. U každého typu se nachází seznam vlastností, které daný subjekt má. U každé vlastnosti lze zaškrtnout, zda ji použít v novém názvu subjektu. Aplikace dále nabízí

možnost na konec nového názvu zahrnout původní ID subjektu pro jednoznačnost. Aplikace interaktivně zobrazuje, jak bude přibližně nový název vypadat.

Dále uživatel uvidí tabulku se seznamem změn, které při přejmenování proběhnout. Pokud souhlasí, může přejít na stažení souboru. Pokud ne, může se vrátit na předchozí formulář a provést další změny. Stránka zároveň umožňuje stáhnout přehled změn v *.csv* formátu.

Uživateli je na konci nabídnut ke stažení soubor s novými názvy subjektů ve stejném formátu, v jakém byl původní soubor.

5 Závěr

Semestrální práce kompletně splňuje zadání číslo 4(a). Aplikace umožní uživateli nahrát RDF soubor v nejčastěji používaných formátech. Aplikace zjistí vlastnosti (včetně zanořených do stupně 2) k jednotlivým třídám a nabídne uživateli možnosti pro přejmenování subjektů. Po schválení změn uživatelem a přejmenování vrátí nový soubor s novými názvy subjektů.

Práce obsahuje detailní informace o implementaci a uživatelskou příručku.

Při tvorbě práce jsem si prohloubil znalosti RDF a naučil se základní používání knihovny Apache Jena.

Na práci bych se díval spíše jako na funkční prototyp, než hotové dílo. Spousta částí programu je dle mého implementována zbytečně složitě, ale lepší řešení jsem nenalezl. Přesto myslím, že prototyp úspěšně dokazuje, že přejmenování uzlů na základě svých vlastností je možné, nicméně s určitými omezeními.