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

24. listopadu 2019

Welcome to RDF Node renaming program

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

Type	Properties					
_links	☐ Include original ID	☐ self	☐ type			
character_link	☐ Include original ID	☐ battle_tag	☐ id	☐ key	☐ type	
clan_link	☐ Include original ID	☐ clan_name	☐ clan_tag	☐ decal_url	☐ icon_url	☐ id
key	☐ Include original ID	☐ href	☐ type			
ladder	☐ Include original ID	☐ _links	☐ dateCrawled	☐ id	☐ ladderId	☐ league
league	☐ Include original ID	☐ key	☐ league_key	☐ type		
league_key	☐ Include original ID	☐ league_id	☐ queue_id	☐ season_id	☐ team_type	☐ type
legacy_link	☐ Include original ID	☐ id	☐ name	☐ path	☐ realm	☐ type
member	☐ Include original ID	☐ character_link	☐ legacy_link	☐ played_race_count	☐ type	
played_race_count	☐ Include original ID	☐ count	☐ race	☐ type		
race	☐ Include original ID	☐ en_US	☐ type			
self	☐ Include original ID	☐ href	☐ type			
team	☐ Include original ID	☐ current_rank	☐ current_win_streak	☐ highest_rank	☐ id	☐ join_time_s

Submit

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

2.3 Předpoklady

Při tvorbě semestrální práce vycházím z předpokladu, že každý RDF subjekt má informaci o svém typu. Tato informace je uložena jako jeden z výrazů, kde je predikátem následující URI `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`.

Druhým předpokladem je, že všechny subjekty stejného typu mají stejné predikáty. Nemůže se například stát, že jeden subjekt typu auto obsahuje informaci o objemu motoru a druhý ne.

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`.

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.

Pokud subjekt obsahuje více výrazů stejného typu, pouze s jinými objekty, jsou tyto výrazy sloučeny do jednoho predikátu s informací o počtu těchto predikátů a původní predikáty odstraněny. (Zde se jedná pouze o úpravu pro uživatelskou volbu k přejmenování, původní model je ponechán beze změn.)

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. (Zde je každý predikát reprezentován svým ID) Aplikace také nabízí možnost zaškrtnout volbu pro přidání původního ID subjektu na konec nového názvu. U několika subjektů se mi při testování stalo, že i při zaškrtnutí všech možných predikátů (včetně ID!) stejně nebyl finální název jednoznačný.

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 2 pole. První pole obsahuje ID predikátů, které byly zvoleny pro přejmenování. Druhé pole obsahuje názvy typů, u kterých bude na konec nového názvu připojeno původní ID.

Tyto pole 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`. Zároveň se u `RdfPredicate` změní hodnota `selected` na `true` (toto slouží pro případ, že dojde k reloadu stránky z důvodu nejednoznačnosti názvů, uživatel uvidí, které položky už označil).

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 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. 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 metody 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 sloučeny do jednoho (všechny typy aut mohou být označeny jako "auto", když při přejmenování nevyužijeme predikát "značka"nebo "SPZ").

Proto po přejmenování probíhá kontrola nového modelu. Pokud nový model má stejný počet subjektů, jako původní model a počet trojic je u nového modelu stejný jako u původního, přejmenování je považované za úspěšné. Pokud ne, uživateli se načte znovu formulář pro přejmenování s chybovou hláškou informující o nejednoznačnosti názvů.

3.4 Možnosti pro zlepšení a další úpravy

Jak bylo řečeno, funkčnost programu je zajištěna dvěma předpoklady. Každý subjekt musí mít výraz s informací o svém typu a u všech subjektů stejného typu se předpokládají stejné predikáty. Prvním krokem k vylepšení je tedy zbavit program požadování těchto předpokladů.

Při testování na daných souborech jsem došel do stavu, kdy jsem použil všechny predikáty pro přejmenování subjektu a nový název stejně nebyl jednoznačný. Proto jsem musel zavést možnost připojit původní ID. Další vylepšení by mohlo být zbavit se nutnosti používat původní ID. Teoreticky by šlo pro nový název používat i několikanásobně zanořené vlastnosti.

Upravit by šla i kontrola jednoznačnosti názvů v novém modelu. Mé řešení je sice funkční, ale hodně „kostrbaté“.

Frontendová část byla napsána jen pro ověření funkčnosti, zde lze program hodně vizuálně vylepšit (lze například uživateli zobrazovat, jak bude nový název vypadat na základě zaškrtnutých predikátů).

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.

V případě, že přejmenování na základě navolených vlastností proběhne v pořádku, uživateli je nabídnut ke stažení soubor s novými názvy subjektů ve stejném formátu, v jakém byl původní soubor.

Pokud dojde během přejmenování ke konfliktům (názvy jsou nejednoznačné), uživateli se znovu načte tabulka s chybovou hláškou, která informuje o nejednoznačnosti nových názvů.

5 Závěr

Semestrální práce kompletně splňuje zadání číslo 4(a). Práce umožní uživateli nahrát RDF soubor v nejčastěji používaných formátech a nabídnout možnosti pro přejmenování subjektů. Po přejmenování uživateli vrátí nový soubor s novými názvy subjektů.

Práce obsahuje detailní informace o implementaci a uživatelskou příručku. V práci jsou zmíněny předpoklady pro to, aby program fungoval, jak má. Práce také zmiňuje možnosti dalších úprav a vylepšení.

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. Velkou částí práce byla pouze investigace a spousta pokusů, které vedly do slepých uliček. 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.