Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Tomáš Hladík

Tvorba datové sady o publikacích Univerzity Karlovy dle principů Linked Data

Katedra softwarového inženýrství

Vedoucí bakalářské práce:	RNDr. Jakub Klímek
Studijní program:	Informatika
Studijní obor:	Obecná informatika

Praha 2013

Poděkování.

Chtěl bych poděkovat RNDr. Jakubu Klímkovi za příkladné vedení bakalářské práce, za konstruktivní připomínky nejen při psaní této práce, za trpělivost, ale také za další rady a tipy spojené s touto prací. Zároveň bych chtěl poděkovat rodičům, kteří mi umožnili a podporují mě ve vysokoškolském studiu, Mgr. Janu Popelkovi za poskytnutí základní verze dat o publikacích Univerzity Karlovy a Michalu Vávrovi za pomoc se stylistickými úpravami práce.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Praze dne 20. května 2013 Podpis autora

Název práce: Tvorba datové sady o publikacích Univerzity Karlovy dle principů Linked Data

Autor: Tomáš Hladík

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Jakub Klímek

Abstrakt: Cílem této práce bylo vytvořit datovou sadu o publikacích Univerzity Karlovy, která bude přívětivější ke zpracování počítačem a následné vizualizaci. K dispozici máme existující oficiální aplikaci obsahující veškeré publikace vydané na Univerzitě Karlově, která ale nad velkým množstvím dat nefunguje ideálně. Informace jsme získávali přes export v XML, dále jsme data zpracovali za pomoci programovacího jazyka Java do požadovaného formátu, nad kterým se můžeme následně dotazovat mnohem rychleji, než jak tomu je v oficiální aplikaci. Díky této práci můžeme zapojit Univerzitu Karlovu do projektu Linked Universities, který sdružuje univerzity poskytující data v počítačově příjemnějším formátu.

Klíčová slova: RDF, Linked Data, sémantický web, publikace

Title: Creating a dataset about publications of Charles University according to the principles Linked Data

Author: Tomáš Hladík

Department: Department of Software Engineering

Supervisor: RNDr. Jakub Klímek

Abstract:

The aim of this thesis was to create a dataset about publications of Charles University, that will be friendlier for computer processing and subsequent visualization. There is already existing official application, which includes every publication issued by Charles University, nevertheless the application does not work perfectly because of the large amount of data. Information has been gained through exports in XML, further the data has been processed into the desired format, over which we can query much faster than it is in the official application. The process has been made by using programming language Java. We engage Charles University into the Linked Universities project, which brings together universities providing data in a computer-friendlier format.

Keywords: RDF, Linked Data, semantic web, publications

Obsah

Úvod **2**

1 Prvotní náhled **3**
1.1 Motivace a cíl . . . 3
1.2 Základní použité technologie . . . 4
1.2.1 Značkovací jazyk XML . . . 4
1.2.2 RDF datový model . . . 6
1.3 Využití sémantického webu . . . 11
1.3.1 Linked Data . . . 11
1.3.2 Linked Universities . . . 13
1.3.3 Datová sada DBpedia . . . 14

2 Datová sada **15**
2.1 Zdroj dat - aplikace OBD UK . . . 15
2.2 Struktura XML dat . . . 16
2.3 Návrh datové struktury . . . 17
2.4 Použité slovníky . . . 18

3 Převod dat **21**
3.1 Autoři a editoři . . . 21
3.2 Číselník stavů a formátů . . . 22
3.3 Fakulty . . . 23
3.4 Knihy . . . 24
3.5 Konference . . . 25
3.6 Pracoviště . . . 25
3.7 Přiřazení pracovišť k autorům publikací . . . 26
3.8 Vydavatelé . . . 27
3.9 Záznamy . . . 28

4 Uložení a vizualizace dat **29**
4.1 Dotazovací jazyk SPARQL . . . 29
4.1.1 Základní syntaxe . . . 29
4.1.2 Matematické funkce . . . 31
4.1.3 Další klíčová slova . . . 31
4.2 Grafy . . . 33
4.3 Server Virtuoso a SPARQL endpoint . . . 34
4.4 Vizualizace dat v programu Payola . . . 35
4.4.1 Vizualizace SPARQL dotazu . . . 36
4.4.2 Propojení dat . . . 39
4.4.3 Propojení SPARQL dotazu s DBpedií . . . 42

5 Uživatelská dokumentace **46**
5.1 Konfigurační soubory . . . 46
5.2 Spuštění aplikace . . . 48
5.3 Minimální požadavky aplikace . . . 50
5.4 Automatické aktualizace . . . 50

5.5 Používané třídy 51
5.6 Netypické chování aplikace 51
5.7 Kompletní vytvoření celého grafu 52

6 Programátorská dokumentace 54
6.1 Třída Download 54
6.2 Třída Update 55
6.3 Třída ZpracovaniXML 56
6.4 Třída XML 56
6.4.1 Třída Autor 59
6.4.2 Třída Konference 59
6.4.3 Třída Pracoviste 60
6.4.4 Třída Kniha 60
6.4.5 Třída Titul 60
6.4.6 Třída Vydavatel 61
6.4.7 Třída Zaznam 61
6.5 Další třídy 62
6.5.1 Třída Jazyky 62
6.5.2 Třída Města 62
6.5.3 Třída MojeFunkce 63
6.6 Nahrání souboru na server Virtuosa 64
6.6.1 Třída SparqlAuth 64
6.6.2 Třída UpluoadVirtuoso 64
6.6.3 Třída UpVirtuoso 64
6.7 Možnosti rozšíření 65

7 Obsah CD 66

Závěr 67

Seznam použité literatury 68

Seznam tabulek 72

Seznam použitých zkratek 73

Úvod

Cílem této práce je zpracování dat poskytovaných Univerzitou Karlovou o vydaných publikacích. Univerzita Karlova monitoruje veškeré publikace (s výjimkou tří lékařských fakult) v centrálním systému pro evidenci publikační činnosti pracovníků a studentů UK. Tato aplikace OBD UK je dostupná na webových stránkách s přihlašovacími údaji z Centrální autentizační služby UK pouze uživatelům, kterým je přiděleno oprávnění pro přístup. Aplikace nabízí svým uživatelům funkce pro přidávání oprávnění k přístupu dalším uživatelům, přidávání nových záznamů, prohlížení detailů publikací v různých jazycích, filtrování publikací podle zvolených kritérií. Podle zadaných filtrovacích kritérií můžeme výsledná data z aplikace exportovat.

V této aplikaci je monitorováno přes 80.000 záznamů, odezva aplikace na většinu filtrovacích příkazů je v řádu desítek vteřin a export dat všech uložených záznamů pro vypršení časového limitu operace není možný. K záznamům jsou přiřazeny pouze předem určené parametry, nemůžeme tedy přidávat vlastní informace, které u publikace považujeme za podstatné.

Popsané problémy budeme řešit pomocí standardů Linked Data, které popisují metodu publikování informací jako síť vzájemně propojených dat, což nabízí více možností pro použití vhodně strukturovaných informací, ke kterým můžeme současně přiřazovat další údaje z již existujících dostupných datových sad.

V této práci se zaměříme na vytvoření datové sady o publikacích Univerzity Karlovy při použití základních principů Linked Data, představíme si možnosti pro publikování vzniklé datové sady, přidávání dalších informací, které odkazují na jiné, již existující, datové sady. Poukážeme také na způsoby, jak vytvářet vlastní filtrovací kritéria, jak získávat a vizualizovat tato data a jak data publikovat, aby byla veřejně dostupná. V závěru této práce si blíže představíme používání přiloženého programu pro aktualizaci datové sady.

1. Prvotní náhled

V první kapitole této práce se zaměříme na krátké představení technologií a standardů používaných v průběhu celé práce. Představíme si značkovací jazyk XML [1], ve kterém jsme získávali data o publikacích. Následně objasníme použití standardů RDF [2] se dvěma možnostmi serializace RDF/XML a Turtle. V závěru této kapitoly se podrobněji zaměříme na standardy Linked Data a projekt Linked Universities, kvůli kterému tato práce především vznikla. Nakonec si představíme datovou sadu DBpedie, která je nedílnou součástí Linked Data. V následujících kapitolách si podrobněji popíšeme postup této práce, stejně jako detailní návrh datové struktury a možné prezentování dat uživatelům. V závěru této práce se zaměříme na dokumentaci pro uživatele ohledně používání přiloženého programu a také na programátorskou dokumentaci.

1.1 Motivace a cíl

V této práci jsme vytvořili soubor dat, který obsahuje informace o publikacích a jejích autorech, které byly vydány v rámci Univerzity Karlovy. K dispozici máme oficiální internetovou aplikaci, která poskytuje svým uživatelům tyto informace, které zpracujeme do počítačově přívětivějšího formátu.

Motivací pro vytvoření této práce byla samotná aplikace OBD UK poskytující dostatek informací, ale způsob pracování s daty není ideální především z hlediska výkonu a času, proto jsme vytvořili efektivnější způsob, který s daty pracuje podstatně rychleji. Využili jsme standardů RDF a Linked Data. Na následujících stránkách si podrobně představíme, co tyto standardy obsahují, jak postupoval vývoj této práce a jak vypadá výsledná datová sada.

Touto prací se Univerzita Karlova zapojí do projektu Linked Universities, je-

[1] XML - eXtensible Markup Language
[2] RDF - Resource Description Framework

jímž cílem je spojovat univerzity, které nabízejí strukturovaná data lépe přístupná počítačům, se kterými můžeme pracovat snadněji než klasickým přístupem do databáze. Na následujících stránkách nahlédneme na tento princip, představíme si základní technologie potřebné k vývoji datové sady a nakonec také popis struktury dat, která charakterizují veškeré uložené publikace Univerzity Karlovy.

V dalších kapitolách nahlédneme na způsob, jak s těmito daty pracovat nejen při počítačovém zpracování, ale také si popíšeme princip pro jejich zobrazení uživatelům přímo na obrazovce. Popíšeme si také způsob, jak propojovat více existujících datových sad, abychom takto mohli získat více dat než ta, která jsme získali prostřednictvím aplikace OBD UK.

Vytvořená datová sada bude volně veřejně přístupná na stránkách OpenData.cz [3] a Data Hub [4]

1.2 Základní použité technologie

V následující části si představíme použité technologie v rámci této práce. Se značkovacím jazykem XML se během bakalářského studia seznámil téměř každý student, ale ne všichni používají standardy RDF, jehož syntaxe serializace RDF/XML využívá datového modelu XML. Popíšeme si základní standardy, příklady a nahlédneme na formát exportu, který jsme měli pro tuto práci k dispozici.

1.2.1 Značkovací jazyk XML

Jedná se o jeden ze základních formátů pro serializaci a následný přenos dat, se kterým jsme se setkali ve fázi získávání informací o publikacích. Internetová aplikace OBD UK, kterou si blíže představíme v kapitole 2.1, nám nabízí export uložených dat v XML. Základem jsou informace o dané publikaci společně s auto-

[3] `http://linked.opendata.cz`
[4] `http://datahub.io`

ry. Aplikace OBD UK nabízí také exporty ve jiných formátech pro lepší zobrazení uživatelům - Microsoft Word, Excel a ve formátu HTML [5], ale pro tuto práci je XML nejvýhodnější kvůli dalšímu zpracování.

Struktura XML dokumentu

Všechny validní XML dokumenty musí splňovat následující podmínky [10]:

- Každý XML dokument obsahuje právě jeden kořenový element `<nazevElementu>`, kterým dokument začíná, a uzavírací element `</nazevElementu>`, kterým celý dokument končí. Na velikosti písmen záleží, nelze tedy použít např. `</názevelementu>`.
- Každý element musí být určen svým názvem a zároveň může obsahovat atributy `<element atribut="hodnota atributu">`. Hodnoty atributů jsou uvedeny v uvozovkách "text", nebo 'text', ale musí splňovat podmínku stejných uvozovek na začátku i na konci hodnoty atributu.
- Pokud je element prázdný, neobsahuje tedy žádné další vnitřní informace, je značen `<element />`.
- Elementy mohou mít následující vnitřní obsah:
  - Textový obsah
  - Seznam elementů, které se ale nesmí křížit
  - Smíšený obsah textu a elementů

Je nutno říci, že nad XML dokumenty existuje několik dotazovacích jazyků (ku příkladu XPath, XQuery), XML dokument můžeme také testovat na validitu (zda se nacházejí elementy na správném místě, zda mají správný obsah, zda se jedná o jediný výskyt, či celou sekvenci, apod.) vůči XML schématu nebo DTD [6].

[5] HTML - HyperText Markup Language
[6] DTD - Document Type Definition

Pro tuto práci ale XML slouží pouze k přenosu dat, proto se těmito standardy zabývat nebudeme a pokročíme k další možnosti pro serializaci dat, ve které budeme výsledná data reprezentovat.

1.2.2 RDF datový model

Myšlenkou pro vznik datového modelu RDF byl sémantický web [4]. Problémem webových stránek je zejména to, že nezřídka kdy zobrazují důležité informace, data v nich poskytnutá nejsou příliš dobře strukturovaná a jsou tím pádem hůře dostupná z ostatních stránek. Pokud se běžný uživatel pokusí najít nějaké informace, zcela jistě použije jeden z vyhledávačů. Bohužel webové stránky dost často nerozlišují důležité informace od běžného textu, což pro počítačové vyhledávání je podstatným problémem. Z tohoto důvodu vznikl sémantický web, který má za cíl přesněji popisovat sémantiku poskytovaných dat, aby bylo zřejmé, která data mají určitý význam a jsou v kontextu nepostradatelná. To také znamená, že z různých webových stránek můžeme použít data, o kterých víme, že rozšiřují námi poskytované informace.

Model RDF vznikl právě kvůli popisování subjektů a vytváření souborů informací o subjektech. Jedná se tedy vždy o trojice ve tvaru

„subjekt - predikát - objekt“

Zjednodušeně řečeno chceme popsat danou entitu co nejvíce vlastnostmi, které se nám podařilo zjistit. Dané trojice by názorně mohly vypadat takto:

Já	měřím	183 cm
Já	studuji	ve třetím ročníku
Já	umím mluvit	česky
Já	umím mluvit	anglicky
Alois Hladík	vzdělání	vysokoškolské
Alois Hladík	je otcem	Já

Tabulka 1.1: Představa RDF trojic

V tabulce 1.1 se v prvním sloupci nachází subjekt (entita, kterou popisujeme), ve druhém sloupci je predikát (vlastnost, kterou chceme specifikovat) a ve třetím sloupci je objekt (hodnota vlastnosti, kterou může být pouhý text, nebo opět subjekt, který je uveden jako příklad na posledním řádku).

Základní princip RDF tedy známe, následně je důležité zmínit tři důležité body pro tvorbu RDF dokumentů [7]:

1. reprezentace subjektů a objektů, abychom mohli zároveň rozlišit, který objekt je pouhým textem a o kterém objektu víme více bližších informací
2. používání slovníků, již existujících standardů, které napomáhají zpracování počítačem
3. uložení dat

Základním stavebním kamenem pro tvorbu RDF dokumentů jsou správné URI [7], které reprezentují subjekty. Musíme tedy dbát zvýšené opatrnosti při tvorbě odkazů, musíme zachovávat jednoznačnost a zároveň se snažit o to, aby mezi odkazy byla jistá logika z důvodu přehlednosti, abychom mohli předem odhadnout, co se za daným odkazem může skrývat.

Druhým bodem, který nás bude zajímat, je používání vlastností, které jsou

[7] URI - Uniform Resource Identifier

popsané takzvanými slovníky. Množina vlastností obsažených ve slovníku je typicky vztažena k používání u podobných událostí, např. použijeme jiný slovník k popisu informací o lidech a jiný slovník pro popis publikací, nicméně počet slovníků pro používání v rámci jednoho dokumentu není nijak omezen. Slovníky zpravidla nabízejí dva typy údajů, které můžeme použít - vlastnosti a třídy. První vlastností, kterou můžeme vždy použít, je označení typu subjektu a právě pro určení hodnoty této vlastnosti použijeme třídu. Použití vlastností je poté již intuitivní. Každý slovník poskytuje na své webové stránce specifikace, kde můžeme získat veškeré potřebné informace, tedy třídy, vlastnosti a způsob použití.

V následujícím příkladu si názorně ukážeme na příkladu člověka, jak by správně měly vypadat poskytované informace. Budeme popisovat osoby, použijeme tedy slovník FOAF - Friend Of A Friend (předpona foaf), který je specifikován na webových stránkách: `http://xmlns.com/foaf/0.1/` [13]. Všichni popisovaní lidé budou mít nastaven typ foaf:Person.

Subjekt	**Predikát**	**Objekt**
<*[8]/Tomáš-Hladík>	rdf:type	foaf:Person
<*/Tomáš-Hladík>	foaf:age	”22”^^xsd:positiveInteger
<*/Tomáš-Hladík>	foaf:name	”Tomáš Hladík”@cs
<*/Alois-Hladík>	rdf:type	foaf:Person
<*/Alois-Hladík>	foaf:title	”RNDr.”
<*/Tomáš-Hladík>	foaf:knows	<*/Alois-Hladík>

Tabulka 1.2: Příklad RDF trojic

V tabulce 1.2 si můžeme všimnout použití typů u objektů. Pokud se jedná o souvislý text, můžeme specifikovat jazyk textu. Pokud se jedná např. o číslo, potom můžeme připojit typ (celočíselný, kladný, ...), stejně jako to umožňuje XML Schema [6].

[8] * značí nahrazení URI - může být reprezentována např. takto http://www.example.com/Person

Posledním použitým slovníkem je slovník RDF Syntax s předponou rdf, který slouží jako základní slovník pro veškeré RDF dokumenty a ostatní slovníky, které od něj mohou odvozovat své údaje [12].

Slovník RDF Syntax slouží jako základ i pro většinu ostatních slovníků. Je možné si definovat vlastní třídy a vlastnosti na základě již existujících tak, že omezíme, rozšíříme nebo specifikujeme již existující informaci.

Nakonec nám zbývá poukázat na to, jak má být soubor správně uložen. Zde existují dvě nejčastěji využívané metody - RDF/XML a Turtle, na které se zaměříme v následujících dvou kapitolách.

RDF/XML

Syntaxe, která je bližší počítačovému zpracování, vychází z klasického XML modelu. Kořenový element obsahuje předpony a definice použitých slovníků. Každý element bude obsahovat dané trojice, a pokud o jakékoli entitě používáme více informací, všechny uvedeme v rámci podelementu dané entity.

V příkladu si popíšeme Katedru softwarového inženýrství (KSI) za použití slovníků Academic Institution Internal Structure Ontology (předpona aiiso) [25] a RDF Schema, které definuje kořenový element. Vnitřní element aiiso:Department obsahuje atribut s hodnotou URI zvolené entity a podelementy značí vlastnosti vztahující se k této katedře. Smíšený vnitřní obsah elementů je v této fázi značně nevyhovující.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:aiiso="http://purl.org/vocab/aiiso/schema#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
     <aiiso:Department
         rdf:about="http://linked.opendata.cz/resource/
```

```
    publications/department/cz/cuni/1294">
        <aiiso:code>KSI</aiiso:code>
        <aiiso:description>Katedra softwarového inženýrství
        </aiiso:description>
    </aiiso:Department>
</rdf:RDF>
```

Turtle/N3

Druhá možnost serializace RDF dokumentu je přívětivější k vývojářům i uživatelům a obsahuje právě dané trojice [9], které jsou přímo vidět a nejsou skryté za žádným XML kódem. Samotnou syntaxi jsme si již téměř popsali v dřívějších tabulkách. Nejdříve definujeme používané slovníky, následně sepisujeme informace o entitách. Převod mezi oběma druhy formátů je triviální, proto většina aplikací pracujících s RDF daty umí pracovat s oběma formáty stejně dobře. Pro uložení souboru ve formátu Turtle/N3 používáme koncovku .ttl [10]. Předchozí příklad by správně syntakticky vypadal následovně:

```
@prefix aiiso: <http://purl.org/vocab/aiiso/schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
<http://linked.opendata.cz/resource/publications/
department/cz/cuni/1294>
    a                   aiiso:Department ;
    aiiso:code          "KSI" ;
    aiiso:description   "Katedra softwarového inženýrství" .
```

Z příkladu je zřejmé, že místo klasického zápisu trojic můžeme zapisovat pouze predikát a objekt, které jsou společné dané entitě. Vlastnosti jsou oddělené střed-

[9] N3 - notace trojic
[10] TTL - Terse RDF Triple Language

níkem, celkový souhrn ukončíme tečkou. Predikát „a“ je náhradou za predikát „rdf:type“.

1.3 Využití sémantického webu

V závěru této kapitoly si představíme standardy Linked Data a projekt Linked Universities, který byl hlavní motivací pro vznik této práce.

1.3.1 Linked Data

Již na začátku práce jsme se zmínili o tom, že dnešní web je směřován spíše pro běžného uživatele, který dokáže webovými stránkami projít tak, že díky své intuici zvládne vyfiltrovat důležité a méně podstatné informace. Umělé inteligenci ale toto domýšlení chybí. Linked Data se tedy snaží co nejvíce rozšířit webové stránky poskytující strukturovaná data, informace a vědomosti z URI a připojených RDF data. Velký důraz je kladen právě na správnou volbu URI. Rozhodně bychom se měli snažit o co nejvyšší unikátnost, abychom minimalizovali možnost duplicitních URI jak v rámci používané datové sady, tak i v rámci Internetu obecně. Jeden příklad HTTP [11] URI je možné najít v předchozí kapitole, obecně URI pro tuto práci budou ve formátu

`http://linked.opendata.cz/resource/publications/cz/cuni/ID`,

kde ID je jednoznačný identifikátor zvolené entity. Všechny uložené záznamy mají URI přesně v tomto tvaru, ostatní modelované entity mají pozměněný tvar, ku příkladu autoři a editoři záznamů mají v URI navíc uvedený řetězec „people“ pro odlišení záznamů a osob.

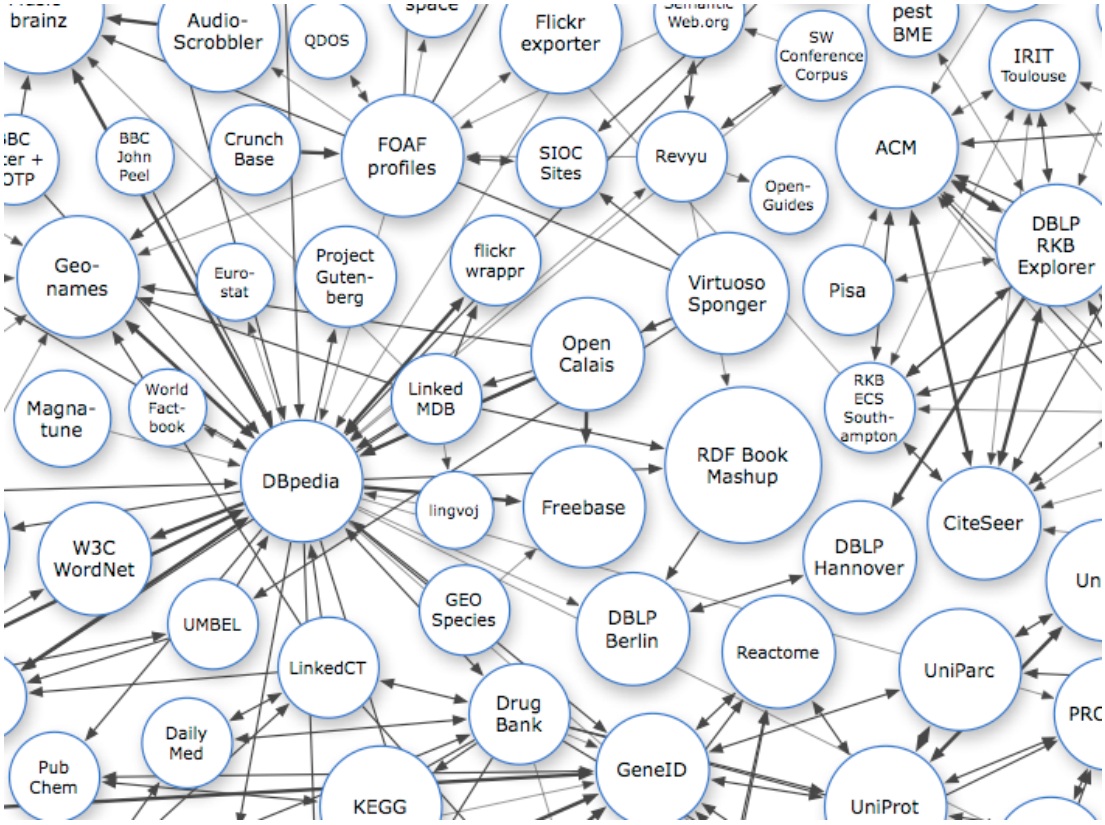
[11] HTTP - HyperText Transfer Protocol

Dalším důležitým principem Linked Data [5], který je nutno zmínit, je zobrazení dat. Cílem pro nás je propojení dat z různých datových sad, ale způsob, kterým se zobrazí data na obrazovce uživateli, není v tuto chvíli tak důležitý. Důležitější pro tuto práci bylo vytvořit datovou sadu, umístit ji na veřejné místo, aby byla viditelná i pro ostatní uživatele, a provázat entity s datovými sadami poskytovanými dalšími uživateli. Příklad můžeme uvést u publikace, o které víme, že byla publikovaná v Praze. O Praze toho datová sada neříká mnoho, pouze dokážeme zjistit, které publikace byly v Praze vytvořeny a publikovány, ale už nedokážeme ani zjistit, ve kterém státu se Praha nachází. Některé další datové sady se naopak zaměřují na sbírání informací o městech, které nám o Praze mohou vypsat více dostupných znalostí. K samotné vizualizaci dat se dostaneme v kapitole 4.

Každým rokem se tento projekt rozšiřuje o nové datové sady, aktuální stav je dostupný na oficiálních stránkách projektu [12]. LOD Cloud [13] větších datových sad si můžeme představit na obrázku 1.1.

[12] `http://linkeddata.org`

[13] LOD Cloud - The Linking Open Data Cloud, soubor datových sad splňujících standardy Linked Data



Obrázek 1.1: Propojení existujících datových sad

Samozřejmostí je poskytování informací spojených s URI. Již jsme si popsali, jak značit entity, posledním krokem je tedy ke každé entitě přiřadit data, což se rovná vytvoření validního RDF dokumentu popsaného v kapitole 1.2.2, kde URI bude jednoznačně určovat popisované subjekty.

1.3.2 Linked Universities

V následující kapitole si představíme další projekt, kvůli kterému tato práce vznikla. Vývojáři vidí velký potenciál v poskytování informací v rámci pouhých dat, zvláště pak na akademické půdě. Do projektu Linked Data se zapojilo mnoho uživatelů, ale v době napsání této práce se mezi propojené univerzity Linked Universities [16] řadilo pouze 8 univerzit. Jedná se o čistě evropský projekt, do kterého jsou zapojeny prozatím 3 anglické, 1 finská, řecká, německá, švédská a francouzská univerzita, a proto bychom chtěli do tohoto projektu zapojit také

Univerzitu Karlovu, která bude poskytovat takto strukturovaná data o svých publikacích. Samozřejmostí je možnost v budoucnu tuto datovou sadu rozšířit o další (libovolné) užitečné informace.

1.3.3 Datová sada DBpedia

Zároveň také nesmí chybět zmínka o projektu DBpedia [2] [14], jedná se o projekt, který byl předběžně spuštěn 23. ledna 2007, ale oficiální verze se datuje k 6. srpnu 2012 [26]. Za DBpedií stojí dvě univerzity z Lipska a Berlína ve spolupráci se společností OpenLink Software [15]. Cíl DBpedie je zřejmý, již jsme si popsali, že Linked Data poskytují pouze dostupné informace a nijak se nestarají o zobrazování. Proto vznikl projekt DBpedie jako součást projektu Wikipedie [16] (největší internetová encyklopedie, na jejíž tvorbě se může podílet každý člověk, a právě proto nabízí tolik dat). DBpedie si kladla za cíl posbírat tato data a vytvořit vhodné trojice pro RDF formát. Wikipedie obsahuje spoustu textu, ale zároveň také velmi často informační tabulku v pravé části stránky (s údaji typu počet obyvatel, hlavní město státu, rozloha, vlajka, ...), a to jsou přesně ta data, která jsou pro RDF užitečná. Subjekt je určen URI, predikát a objekt je poté již určen tabulkou (např. počet obyvatel je 10.000.000), problém je tedy jen se správným určením typů a vlastností z daných slovníků. Stejně jako Wikipedie, tak i DBpedie existuje v různých jazykových verzích, a proto můžeme z jedné stránky získat více dat z několika světových jazyků. Pokud v naší datové sadě tedy máme zmínku o městě, budeme se snažit použít odkaz přímo na DBpedii, protože můžeme získat mnohem více informací, které by nás mohly zajímat.

Pro srovnání velikostí datových sad použijeme počet uložených trojic. Námi vytvořená datová sada obsahuje 3.160.712 trojic, datová sada DBpedie 354.419.073 trojic [17].

[14] `http://dbpedia.org/About`
[15] `http://www.openlinksw.com/`
[16] `http://www.wikipedia.org/`
[17] údaje jsou platné k 7.5.2013

2. Datová sada

V předchozí kapitole jsme se zaměřili na to, jak obecně vytvářet propojená data, v této kapitole se dostaneme ke konkrétnímu příkladu, který byl námětem pro tuto práci. Budou nás zajímat veškeré publikace publikované v rámci Univerzity Karlovy, které jsou uložené v centrální databázi.

2.1 Zdroj dat - aplikace OBD UK

V současné době se již dokončuje zavedení centrálního systému, který eviduje veškerou publikační činnost této univerzity. Tento projekt byl zahájen v roce 2008 a pro přístup do aplikace stačí přihlašovací údaje z Centrální autentizační služby Univerzity Karlovy [1], ale také přihlašovací práva. Každá fakulta (včetně několika ústavů) Univerzity Karlovy má svého správce, výjimkou jsou pouze tři lékařské fakulty, které momentálně čekají na napojení do systému. Aplikace OBD UK je dostupná přes webové prohlížeče na webových stránkách [2]. V tuto chvíli databáze obsahuje více než 80.000 publikací. Při práci s aplikací se setkáváme se dvěma hlavními problémy. Prvním je časová odezva samotné aplikace. Pokud chceme data o publikacích dále zpracovávat, případně si je samostatně ukládat a vytvářet databázi jen pro zvolenou katedru, narazíme na problém, kdy exporty pracují až příliš pomalu a často náš požadavek ukončí až vypršení časového limitu pro danou operaci. Z tohoto důvodu potřebujeme najít jiný způsob, jak s daty pracovat. Tento způsob jsme si názorně popsali v minulé kapitole. Druhým problém je to, že Univerzita Karlova je velkou organizací - během zpracování projektu jsme se dostali častokrát do fáze, kdy jsme potřebovali mírně upravit XML výstup. Naše přání nebyla realizována, jelikož firma spravující OBD UK má několik požadavků na opravu aplikace a ten náš nebyl až tak důležitý. Proto jsme museli hledat náhradní řešení, která si podrobněji popíšeme v části dokumentace.

[1] CAS UK: `https://ldap1.cuni.cz/`

[2] `http://verso.is.cuni.cz`

V následujících kapitolách si nejdříve uvedeme, s jakými daty jsme pracovali, v jakém formátu jsme měli tato k dispozici a do jakého formátu jsme je chtěli převést.

2.2 Struktura XML dat

Co se týká publikací, aplikace OBD UK nabízí klasické procházení, filtrování a zobrazení detailů jednotlivých publikací. V této práci jsme využili možnost exportu více záznamů. Pro účel této práce nám správce aplikace pro MFF [3] poskytnul všechna uložená data ve formátu XML do 22. listopadu 2012, což výrazně zrychlilo export uložených dat. Dalšími typy exportů, vedle námi požadovaného XML formátu, jsou formáty `doc` (textový dokument pro zobrazení v programu Microsoft Word), `xls` (tabulkový dokument pro aplikaci Microsoft Excel) a `html` (webová stránka). Velký XML export nám poskytoval data až do poloviny listopadu, proto stáhnout a pravidelně aktualizovat data již není problém.

Jelikož potřebujeme data dále zpracovávat, je zřejmé, že XML formát je pro nás nejlepší variantou a stručně si popíšeme data, která se nám podařilo získat. Export v XML formátu je řešen jako seznam záznamů v jednom kořenovém elementu. O každém záznamů máme k dispozici seznam autorů, název a abstrakt v alespoň jednom jazyce, velmi často konferenci, kde byla publikace prezentována, také ISSN [4], ISBN [5], počet stránek a mnoho dalších podobných informací. Dále ještě o autorech dokážeme zjistit jméno, příjmení, tituly, identifikační UK číslo a pracoviště (fakulta, katedra, ...). Musíme ale dodat, že ne všechny informace jsou vyplněny korektně, některé podstatné informace často chybí, proto jsme se na tyto poznatky snažili klást co největší důraz pro odstranění, případně editaci na místech, kde byla chyba evidentní. XML export vypadá přibližně následovně:

[3] MFF - Matematicko-fyzikální fakulta Univerzity Karlovy v Praze
[4] International Standard Serial Number
[5] International Standard Book Number

```
<zaznamy><zaznam id="75854">
   <autor_list><autor>
      <jmeno>Jakub</jmeno>
      <prijmeni>Klímek</prijmeni>
      <titul_pred>RNDr.</titul_pred>
      <titul_za />
      <kodautora>16667789</kodautora>
      <autor_prac>
         <kodprac>1294</kodprac><zkratka>KSI</zkratka>
         <fakulta_zkratka>MFF</fakulta_zkratka>
         <popis_prac>Katedra softwarového inženýrství
         </popis_prac><popis_fakulta>
         Matematicko-fyzikální fakulta</popis_fakulta>
      </autor_prac>
   </autor></autor_list>
   <titul_list><titul>
      elementy <nazev>, <jazyk>, <abstrakt>, <klicova_slova>
   </titul></titul_list>
   elementy <odkaz><url></odkaz>, <isbn>, <issn>,
   <konference_nazev>, <konani_misto>, <dat_kon_zacatek>,
   <dat_kon_konec>, <vydavatel_mesto>, <vydavatel_nazev>,
   <strany>, <rok>, <literarni_forma>, <stav>
   a několik dalších elemtů
</zaznam></zaznamy>
```

2.3 Návrh datové struktury

Datová strukturu byla navržena dle existujících slovníků tak, aby se logika struktury výrazně nelišila od dřívěji popsaného XML modelu, který nám nabízí export aplikace, ale také jsme vytvářeli z jednoho záznamu co nejvíce entit. To

znamená, že místo jednoho záznamu zajistíme vytvoření co nejvíce dalších entit vztahujících se k danému záznamu. Výhodu tohoto postupu si můžeme popsat na příkladu autorů. V aplikaci OBD UK je problém s autory a jejich tituly. Pokud autor dosáhne vyššího vzdělání, v aplikaci OBD UK bychom museli u všech záznamů změnit titul tohoto autora. V používané datové sadě máme pro každého autora pouze jednu přiřazenou entitu, stačí tedy provést jednu změnu, která se následně projeví u všech uložených záznamů.

Modelované entity během postupu práce jsou následující:

- Základní informace o záznamu (název, abstrakt a klíčová slova v několika jazycích, ISBN, ISSN, počet stran, id, ...)
- Autoři (jméno, příjmení, tituly, osobní číslo)
- Pracoviště (katedry) autorů u jednotlivých publikací
- Konference, na které byl záznam publikován
- Sborníky, případně jiná díla, ve kterých je publikace založena
- Fakulta, na které byl záznam vytvořen
- Stav a formát publikace, správce aplikace OBD UK povolil přidávání publikací pouze ve stavu „přijatém“, ale pro testování jsme mohli využít více stavů
- Informace o vydateli publikace

2.4 Použité slovníky

V poslední části této kapitoly se zaměříme na používané slovníky. Existujících slovníků je dostupných velmi mnoho, proto může být obtížné najít tu metodu a třídu, kterou přesně potřebujeme, často bývá výhodnější si napsat vlastní potřebnou metodu. Slovníky jsou tematicky řazené, proto je silně doporučováno použít

existující metodu z důvodu případného budoucího hledání podobných dat. V práci jsme použili následujících 14 slovníků (s krátkým popiskem funkcionality využité v této práci, v závorce je uvedena používaná předpona, v poznámce HTTP URI specifikace):

- Academic Institution Internal Structure Ontology (`aiiso`) [25] [6] - popsání oddělení ve společnosti, v tomto případě fakult na univerzitě
- Bibliographic Ontology (`bibo`) [14] [7] - základní slovník sloužící k popisu literatury a většiny publikací
- DBpedia Ontology (`dbpedia-owl`)[8] - napojení na datovou sadu DBpedie
- Dublin Core (`dcterms`) [8] [9] - často používaný slovník k popsání základních údajů o entitách
- Event Ontology (`event`) [21] [10] - popis událostí, v tomto případě nezbytné pro konference
- FOAF - Friend of a Friend (`foaf`) [13] [11] - základní slovník určený pro práci s osobami
- RDF Schema (`rdfs`) [11] [12] - přidání orientačních popisků u jednotlivých entit
- RDF Syntax (`rdf`) [12] [13] - tvorba vlastních metod a tříd
- Simple Knowledge Organization System (`skos`) [18] [14] - nezbytné pro tvorbu číselníků pro stavy a formáty

[6] `http://purl.org/vocab/aiiso/schema#`
[7] `http://purl.org/ontology/bibo/`
[8] `http://dbpedia.org/ontology/`
[9] `http://purl.org/dc/terms/`
[10] `http://purl.org/NET/c4dm/event.owl#`
[11] `http://xmlns.com/foaf/spec/`
[12] `http://www.w3.org/2000/01/rdf-schema#`
[13] `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
[14] `http://www.w3.org/2004/02/skos/core#`

- Time (`time`) [15] [15] a Timeline (`timeline`) [22] [16] - pomůže při bližším určení času
- Vivo Ontology (`vivo`) [17] - slovník pro popisování typů publikací
- Vocabulary of Interlinked Datasets (`void`) [1] [18] - popsání informací ohledně vytvořené datové sady
- XML Schema (`xsd`) [6] [19] - slovník konsorcia W3C [20], který definuje většinu potřebných typů

[15] `http://www.w3.org/2006/time#`
[16] `http://purl.org/NET/c4dm/timeline.owl#`
[17] `http://vivoweb.org/ontology/core#`
[18] `http://rdfs.org/ns/void#`
[19] `http://www.w3.org/2001/XMLSchema#`
[20] W3C - World Wide Web Consortium - `http://www.w3.org`

3. Převod dat

V této kapitole si detailně popíšeme, jak vypadají data na vstupu z XML dokumentu (základní struktura dat je popsána v kapitole 2.2) a jak je převedeme do RDF (viz 2.3). Data jsou při zpracování převedena do správného tvaru pro validní RDF dokument, který serializujeme syntaxí Turtle (viz kapitola 1.2.2). Používáme předpony slovníků popsaných v kapitole 2.4.

V následujících částech si vždy představíme tabulkou XML elementy, které pocházejí z aplikace OBD UK. Druhý sloupec typicky značí použitý predikát, použitý slovník poznáme díky předponě z předchozí kapitoly. Ve třetím sloupci se poté vždy nachází stručné vysvětlení použité vlastnosti.

Detailní postup převádění elementů, návrh datových struktur a používané funkce a knihovny si detailněji představíme v části dokumentace určené programátorům (kapitola 6).

3.1 Autoři a editoři

Tabulka 3.1 znázorňuje, jak převedeme XML elementy týkající se autorů do trojic za použití standardů RDF. V dalších kapitolách budeme postupovat stejně jako u následující tabulky.

XML element	RDF predikát	vysvětlení
`<jmeno>`	`foaf:firstName`	Použitá vlastnost se vždy vztahuje k určení křestního jména osob
`<prijmeni`	`foaf:lastName`	Vlastnost používaná pro určení příjmení osob
	`foaf:name`	Celé jméno člověka
	`dcterms:title`	Vlastnost pro určení názvu entity
`<kodautora>`	`foaf:account`	Identifikační číslo používané pro přístup do účtů spravovaných CAS UK
`<kodprac>`	`foaf:workInfoHomepage`	Základ pro URI u pracoviště autora
`<titul_pred>`	`foaf:title`	Titul člověka
`<titul_za>`	`foaf:title`	Titul člověka

Tabulka 3.1: Převod dat autorů

Pokud je vyplněn element `<kodautora>`, potom pro tvorbu URI použije tento osmimístný kód, v ostatních případech používáme kombinaci jména a příjmení, což nemusí být vždy unikátní kombinace, ale z aplikace OBD UK bližší informace zjistit nedokážeme. URI dále určuje řetězec „people“, výsledná data podle tabulky 3.1 jsou dostupná v přiloženém souboru „output_autori.txt“.

3.2 Číselník stavů a formátů

Pro tvorbu číselníků jsme použili základní stavy a formáty. A ostatní entity jsou odvozené z těchto stavů.

```
<http://linked.opendata.cz/resource/publications/cz/cuni/status>
a skos:ConceptScheme ;
skos:prefLabel "Seznam stavů"@cs ;
skos:altLabel "Status list"@en .
<http://linked.opendata.cz/resource/publications/cz/cuni/format>
a skos:ConceptScheme ;
skos:prefLabel "Seznam formátů"@cs ;
skos:altLabel "List of formats"@en .
```

XML element	RDF predikát	vysvětlení
	`skos:prefLabel`	Popisek entity
	`dcterms:title`	Název entity
	`skos:definition`	Definice entity
	`skos:inScheme`	Hlavní entita (stav/formát)

Tabulka 3.2: Převod dat stavů a formátů

Všechny vlastnosti (s výjimkou poslední) v tabulce 3.2 byly odvozeny z XML elementu `<stav>`, resp. `<literarni_forma>`. K tvorbě URI byl použit název stavu (případně formátu) společně s řetězci „format“, resp. „status“. Výsledná data jsou dostupná v přiloženém souboru „output_stavy.txt“.

3.3 Fakulty

XML element	RDF predikát	vysvětlení
`<fakulta_zkratka>`	`dcterms:title`	Titulek entity za pomocí zkratky fakulty
`<fakulta_popis>`	`dcterms:description`	Detailnější popis, plný název fakulty

Tabulka 3.3: Převod dat fakult

K tvorbě URI byla použita zkratka fakulty společně s řetězcem „faculty“. Výsledná data podle tabulky 3.3 jsou dostupná v přiloženém souboru „output_fakulty.txt“.

3.4 Knihy

Kniha obecně značí publikaci, ve které může být daný záznam uložen, např. sborník a jiné.

XML element	RDF predikát	vysvětlení
`<zdroj_nazev>`	`dcterms:title`	Název entity (knihy)
`<rok>`	`dcterms:date`	Přiřazené datum (v tomto případě rok)

Tabulka 3.4: Převod dat knih

Jelikož je kniha, ve které může být daný záznam publikován, jednoznačně přiřazena k tomuto záznamu, k tvorbě URI bylo použito ID záznamu společně s řetězcem „proceedings“. Výsledná data podle tabulky 3.4 jsou dostupná v přiloženém souboru „output_knihy.txt“.

3.5 Konference

XML element	RDF predikát	vysvětlení
`<konference_nazev>`	`dcterms:title`	Název entity, v tomto případě i konference
`<konani_misto`	`event:place`	Místo konání události
	`event:time`	Čas konání události
	`time:intervalEquals`	Časový interval
`<dat_kon_zacatek>`	`timeline:start`	Začátek intervalu (konference)
`<dat_kon_konec>`	`timeline:end`	Konec intervalu (konference)

Tabulka 3.5: Převod dat konferencí

Jelikož je konference přímo přiřazena k záznamu, k tvorbě URI bylo použito ID záznamu společně s řetězcem „conferences“. Výsledná data podle tabulky 3.5 jsou dostupná v přiloženém souboru „output_konference.txt“.

3.6 Pracoviště

XML element	RDF predikát	vysvětlení
`<zkratka>`	`aiiso:code`	Identifikační kód pracoviště
`<popis_prac>`	`aiiso:description`	Popis pracoviště (celý název)
`<popis_prac>`	`dcterms:title`	Určení názvu entity
`<fakulta_zkratka>`	`aiiso:part_of`	Určení fakulty, jejíž je pracoviště součástí

Tabulka 3.6: Převod dat pracovišť

K tvorbě URI byla použita zkratka pracoviště společně s řetězcem „department“. Výsledná data podle tabulky 3.6 jsou dostupná v přiloženém souboru „output_pracoviste.txt“.

3.7 Přiřazení pracovišť k autorům publikací

Jelikož u každého záznamu mohou být autoři přiřazeni u jiných pracovišť, vytvořili jsme vlastní typ „Affiliation“, který k danému záznamu a autorovi přiřazuje pracoviště. Samotné přepsání elementů znázorňuje tabulka 3.7.

```
<http://linked.opendata.cz/resource/publications/property/
affiliation> a rdf:Property;
rdfs:domain bibo:Article ;
rdfs:label "Affiliation"@cs ;
rdfs:label "Affiliation"@en ;
rdfs:range
<http://linked.opendata.cz/resource/publications/class/
Affiliation> .
<http://linked.opendata.cz/resource/publications/class/
Affiliation> a rdfs:Class ;
rdfs:label "Affiliation"@cs ;
rdfs:label "Affiliation"@en .
```

XML element	RDF predikát	vysvětlení
`<kodautora>`	`foaf:maker`	Vlastnost přiřazuje člověka, který je autorem dané publikace
	`foaf:topic`	Předmět entity, v tomto případě se jedná o záznam
	`dcterms:title`	Vlastnost pro určení názvu entity
`<kodprac>`	`foaf:workInfoHomepage`	Pracoviště autora

Tabulka 3.7: Přiřazení pracovišť k autorům

U přiřazení pracovišť k autorům u daných publikací byla v URI použita kombinace ID záznamu a kódu autora společně s řetězcem „affiliation“. Výsledná data podle tabulky 3.7 jsou dostupná v přiloženém souboru „output_affiliace.txt“.

3.8 Vydavatelé

XML element	RDF predikát	vysvětlení
`<vydavatel_nazev>`	`dcterms:title`	Název entity
`<vydavatel_mesto>`	`dbpedia-owl:hometown`	Přiřazené město u vydavatele

Tabulka 3.8: Převod dat vydavatelů

K tvorbě URI jsme použili název vydavatelství společně s řetězcem „publisher“. Výsledná data podle tabulky 3.8 jsou dostupná v přiloženém souboru „output_vydavatele.txt“.

3.9 Záznamy

XML element	RDF predikát	vysvětlení
`<kodautora>`	`dcterms:creator`	Autor publikace
`<kodautora>`	`dcterms:editor`	Editor publikace
	`affiliation`	Vlastní typ pro přidání pracovišť
`<zaznam>`	`dcterms:identifier`	Identifikátor záznamu
`<isbn>`	`bibo:isbn`	ISBN záznamu
`<issn>`	`bibo:issn`	ISSN záznamu
`<strany>`	`bibo:pageStart`	První stránka záznamu autora v dané publikaci
`<strany>`	`bibo:pageEnd`	Poslední stránka záznamu autora v dané publikaci
`<prace_rozsah>`	`bibo:numPages`	Počet stránek
`<url>`	`rdfs:seeAlso`	Doporučení pro shlédnutí
`<literarni_forma>`	`dcterms:type`	Forma záznamu
`<stav>`	`bibo:status`	Stav záznamu
`<nazev>`	`dcterms:title`	Titulek záznamu
`<abstrakt>`	`dcterms:description`	Abstrakt (popis) záznamu
`<klicova_slova>`	`dcterms:subject`	Důležité informace ohledně záznamu (klíčová slova)
`<vydavatel_nazev>`	`dcterms:publisher`	Vydavatel záznamu
	`bibo:authorList`	Seznam autorů publikace
	`bibo:presentedAt`	Vlastnost, která uvádí, kde byl záznam prezentován
	`bibo:reproducedIn`	Vlasnost uvádějící, kde je záznam zařazen

Tabulka 3.9: Převod dat záznamů

K tvorbě URI jsme použili identifikační číslo záznamu. Výsledná data podle tabulky 3.9 jsou dostupná v přiloženém souboru „output_zaznamy.txt“.

4. Uložení a vizualizace dat

V předchozích kapitolách jsme si popsali, jak správně vytvářet RDF dokumenty a ukládat data. Nyní se dostáváme do části, kdy se zaměříme na problém, jak data představit běžnému uživateli a jak s nimi pracovat. Představíme si také jeden z projektů, který vznikl na této fakultě a slouží k lepší vizualizaci uložených dat. Popíšeme si dotazovací jazyk speciálně určený pro tento typ dat a zjistíme, jak veřejně zpřístupnit výsledná data. Vše si doplníme o příklady, které jsme vytvořili v rámci této práce.

4.1 Dotazovací jazyk SPARQL

Pro práci s RDF dokumenty je možné využít dotazovací jazyk SPARQL [19] [1], který je svojí syntaxí velmi blízký dotazovacím jazykům databází. Při práci s databázemi zjišťuje to, že i samotnou databázi si můžeme představit jako trojice určené pro RDF dokumenty. Subjektem by mohla být vhodná kombinace názvu tabulky s primárním klíčem sloupce, predikátem název sloupce a objektem samotná hodnota sloupce.

4.1.1 Základní syntaxe

Každý SPARQL výraz [20] (stejně jako RDF dokument) začíná definicemi předpon použitých slovníků následujícím způsobem:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

Samotný výraz musí začínat jedním ze čtyř klíčových slov SELECT, CONSTRUCT, ASK, DESCRIBE, po kterém může následovat klíčové slovo WHERE, které zužu-

[1] SPARQL - SPARQL Protocol and RDF Query Language

je vypsaná data splňující vepsanou podmínku. Pro SPARQL výrazy je nezbytné používání proměnných zapisovaných ve tvaru `$promenna` nebo `?promenna`, přičemž platí, že `$promenna` i `?promenna` se vždy ztotožní se stejnými hodnotami a značí i stejnou proměnnou. V našich příkladech budeme značit proměnné otazníkem.

Již jsme se zmínili o existenci pětice klíčových slov, jejich vysvětlení začneme od posledního - WHERE. Klíčové slovo WHERE je určeno pro práci s RDF trojicemi. Vybereme vždy takové trojice, které splňují WHERE část výrazu. Základním příkladem může být `?clovek a foaf:Person.`, což nám zajistí vypsání všech osob ze souboru, nad kterým se dotazujeme. Z příkladu je již zřejmé, že výraz je zapisován jako množina trojic, které jsme si představili v kapitole 1.2.2, a v podobném stylu můžeme psát více omezovacích podmínek.

```
?clovek a foaf:Person;
  foaf:name "Tomáš";
 foaf:knows ?clovek2.

?clovek2 a foaf:Person;
  foaf:name "Alois".
```

Tímto dotazem jsme si omezili podmínky na všechny Tomáše, kteří znají alespoň jednoho Aloise. Stejně jako u databázových SQL [2] dotazů, tak i SPARQL dotazy nabízejí možnost pro vnitřní příkazy, tedy lze použít další další SELECT ve WHERE části dotazu.

Základní klíčová slova mají následující význam:

- `SELECT (?a AS ?vyslednaPromenna)` zajistí vypsání proměnných na obrazovku do tabulky, kde sloupce značí proměnné, které se nám podařilo namatchovat v rámci WHERE podmínky, řádky poté znázorňují všechny možné výsledky, které splňují podmínku.

[2] SQL - Structured Query Language

- `CONSTRUCT` výraz se používá, pokud budeme chtít vytvořit novou entitu z podmínek ve `WHERE` části výrazu. Například okopírování a vytvoření celého souboru je možné jednoduchým použitím `CONSTRUCT {?a ?b ?c} WHERE {?a ?b ?c}`, což znamená, že proměnné se namatchují postupně na všechny existující trojice, které poté klíčové slovo `CONSTRUCT` vloží do výsledného souboru. Podobně si můžeme omezit všechny informace o osobách změnou `WHERE {?a a foaf:Person; ?b ?c.}`.
- `ASK` můžeme využít v případu, kdy chceme zjistit, zda lze namatchovat proměnné splňující podmínky, které běžně píšeme do WHERE části.
- `DESCRIBE ?a` používáme pro extrahování informací o dané entitě. Získáme tedy všechny trojice o subjektu, které se namatchují pro danou proměnnou.

4.1.2 Matematické funkce

Základní syntaxi jsme si popsali, musíme ještě připomenout, že můžeme používat základní matematické funkce, které nám dotazovací jazyk přibližují SQL příkazům. Zápis je naprosto stejný jako v SQL jazyce, tedy `SELECT (COUNT(?pocet) AS ?celkovyPocet)`. Dostupné matematické funkce jsou COUNT pro počet odpovídajících proměnných, SUM pro součet hodnot, AVG, MAX a MIN pro průměrnou, nejvyšší a nejnižší hodnotu použité proměnné.

Podobně můžeme využít zápisu `SELECT DISTINCT ?a` pro zakázání výpisu duplicitních hodnot.

4.1.3 Další klíčová slova

Stejně jako v SQL příkazech můžeme i zde použít klíčová slova `LIMIT` a `ORDER BY`:

`ORDER BY ?a`, příp. `ORDER BY DESC(?a)`

`LIMIT 10`

Ve výsledku tedy obdržíme prvních 10 seřazených hodnot. Pokud bychom chtěli přeskočit prvních n výskytů, poté musíme použít `OFFSET n`.

Předposledním klíčovým slovem, na které se zaměřím, je `GROUP BY`, které slouží pro rozdělení výsledku do skupin.

```
SELECT ?b (count(*) as ?vysledek)
WHERE {?a a ?b}
GROUP BY ?b
HAVING count(*) > 9
ORDER BY DESC(?vysledek)
```

Předchozí příklad nám vrátí použité typy v souboru, nad kterým se dotazujeme, které se vyskytují alespoň desetkrát. Ve výsledné datové sadě se nejčastěji vyskytuje vlastní typ affiliace, kterých je přes 150 tisíc, následuje 65 tisíc prezentací a 48 tisíc článků.

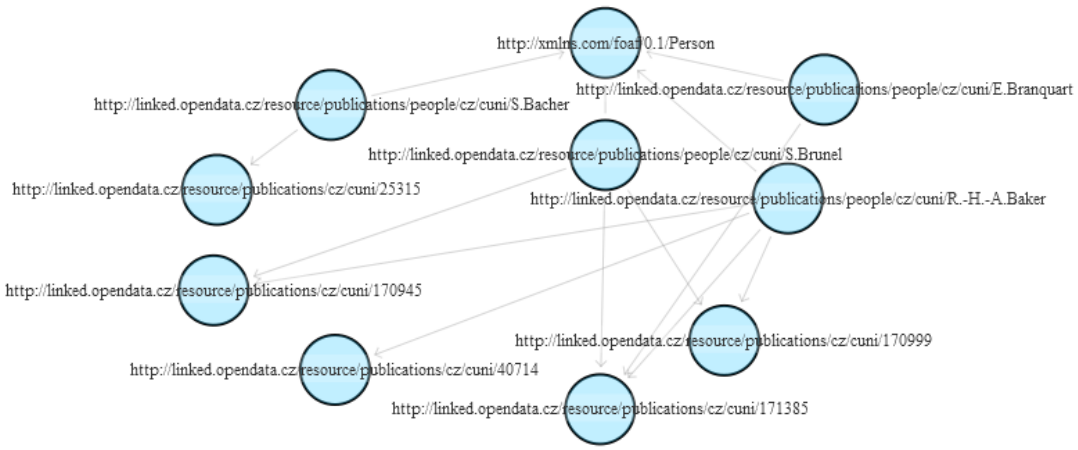
Posledním potřebných klíčovým slovem je FILTER. Oproti klasickému SQL příkazu nám zde chybí možnost omezit proměnné. Přesně to nám zastupuje toto klíčové slovo, které se využívá v části příkazu WHERE. Opět si názorně ukážeme příklad, který vypíše všechny články delší než 10 stránek.

```
   SELECT ?a
WHERE {?a a bibo:Article;
bibo:pageStart ?start;
bibo:pageEnd ?end.
FILTER(?end - ?start >= 10)}
```

Těchto publikací se v datové sadě nachází v tuto chvíli přes 10.000. Nerovnost je v příkazu použita, jelikož publikace začínající na stránce 20 a končící na stránce 30 obsahuje 11 stránek, ale rozdíl vrací číslo 10.

4.2 Grafy

RDF dokument si lze představit jako orientovaný multigraf (množina vrcholů a hran, kde každá hrana spojuje 2 vrcholy), vrcholy budou v tomto případě vždy subjekty a objekty zadaných trojic a hranu reprezentuje predikát, který propojuje dva vrcholy, resp. objekty. Pro každý veřejně přístupný graf musíme zároveň určit IRI [3]. Na následujícím obrázku 4.1 vidíme část grafu obsahujícího 4 osoby (typu foaf:Person), které jsou autory pěti publikací (propojení je realizováno přes predikát foaf:made).



Obrázek 4.1: Ukázka RDF grafu

Již v počátku této práce jsme si popsali princip propojených údajů přístupných na Internetu, proto pojem „graf“ je velmi přesný, i když prozatím jsme nahlédli pouze na graf v rámci datové sady. Stejně tak bychom si multigraf mohli představit jako síť propojených (vzájemně na sebe odkazujících) datových sad. Ve všech grafech je implicitně zakázáno zobrazení duplicitních trojic, což je velkou výhodu, protože nemusíme kontrolovat, zda vložená trojice nevytvoří duplicitní hodnoty. Graficky to znamená, že mezi dvěma vrcholy grafu nemohou existovat dvě stejně pojmenované hrany.

[3] IRI - Internationalized Resource Identifier

4.3 Server Virtuoso a SPARQL endpoint

V předchozích kapitolách jsme se dopouštěli drobných nepřesností tím, že jsme mluvili o dotazování nad souborem, ale ve skutečnosti probíhá dotazování přes SPARQL endpoint nad grafy. SPARQL endpoint se nachází na veřejně dostupném (nebo privátním, pokud prozatím nechceme graf publikovat) serveru a poskytuje přístup k databázi, ve které jsou grafy uloženy. Pro dotazování typicky potřebujeme znát HTTP URI SPARQL endpointu a následně čtveřici parametrů `?graf`, `?subjekt`, `?predikát`, `?objekt`. Proměnná `?graf` značí identifikátor grafu a jednoznačně určuje trojice, které jsou ke grafu připojené.

Již v první kapitole jsme se zmínili o tom, že společnost OpenLink Software podporovala projekt DBpedie, a proto bylo potřebné zajistit server a webové stránky podporující jednoduchou práci s RDF dokuemtny. To především znamená nahrání dat na server, změny a výpis dat. Pro tuto práci jsme měli přístupný Virtuoso Universal Server [24] [4] se SPARQL endpointem [5], resp. SPARQL endpointem s rozšířenými možnostmi pro editaci grafu [6]. Klasický SPARQL endpoint nabízí pouze možnosti pro získání dat, resp. zobrazení v několika formátech - klasické HTML, XML, JSON [7], prostý text, RDF/XML, Turtle a další. Tento endpoint je dostupný všem uživatelům, zabezpečený SPARQL endpoint je přístupný po předchozí autentifikaci, protože nabízí navíc také možnosti přidávání a mazání dat, což rozhodně nepřihlášený uživatel nemůže provádět.

Prozatím jsme si SPARQL Update dotaz [23] pro editaci dat neuvedli, použití následujících příkazů je opět velmi jednoduché:

- `CLEAR GRAPH <graf>` - smaže kompletně celý graf
- `DELETE DATA FROM <graf> {trojice}` - smaže zadané trojice

[4] Virtuoso Universal server: `http://linked.opendata.cz:8890`

[5] SPARQL endpoint: `http://linked.opendata.cz:8890/sparql`

[6] SPARQL endpoint s rozšířenými možnostmi: `http://linked.opendata.cz:8890/sparql-auth`

[7] JSON - JavaScript Object Notation

- `INSERT DATA INTO <graf> {trojice}` - vloží trojice do grafu

V příkazech můžeme vypsat přímo trojice pro smazání nebo vložení, ale také lze použít proměnné stejně jako u klasického SPARQL dotazu, tedy ... `{?a ?b ?c} WHERE {?a a foaf:Person; ?b ?c; ... }`.

Pro velká data ale není tento způsob vkládání praktický, proto nabízí Virtuoso Server i Conductor, který je přístupný přes webové prohlížeče, s několika základními funkcemi, především nahrání validního RDF dokumentu. Dále také nabízí možnost nahrání na server SQL příkazem, který na rozdíl od běžného uploadu vypisuje i chybové hlášky o místech, kde není dokument validní. Nevýhodou SQL příkazu je, že nahrávaný soubor může být pouze ve složkách, které mají přiřazené právo na upload do Virtuosa, proto SQL příkaz na externím serveru nelze využít.

SQL příkaz pro nahrání dat při povolené složce c:/temp vypadá následovně:

```
DB.DBA.TTLP_MT(file_to_string_output('c:/temp/output.ttl'), '',
'http://localhost:8890/graph');
```

Pro tuto práci je dostupný SPARQL endpoint Virtuoso Universal Serveru [8], po dokončení práce bude graf publikován na stránkách OpenData.cz [9] se svým SPARQL endpointem [10] a také na Data Hub [11].

4.4 Vizualizace dat v programu Payola

Nyní se dostáváme do fáze, kdy jsme zjistili, jak mají být data strukturovaná, kam je možné data uložit, jak je vypisovat. V této kapitole se zaměříme na projekt Payola [17] studentů Matematicko-fyzikální fakulty, který dostává Linked Data do nové podoby. Již víme, co to znamená graf, uvnitř kterého mohou být entity vzájemně propojené. Také víme, že grafy mohou odkazovat na jiné grafy.

[8] `http://linked.opendata.cz:8890/sparql`
[9] `http://opendata.cz`
[10] `http://linked.opendata.cz/sparql`
[11] `http://datahub.io`

Již zmíněný projekt ale dodává grafu ten správný význam. Všechno, co k práci s Payolou potřebujeme, je SPARQL endpoint a IRI grafu. Payola nevyžaduje znalost SPARQL jazyka, SPARQL dotaz lze nahradit použitím vestavěných funkcí pro výpis dat za pomoci určených vlastností, přidání vhodných filtrů a dalších podmínek, které simulují ekvivalentní SPARQL dotaz. Základním pojmem Payoly je analýza, která primárně zvládne přesně to, co každý SPARQL dotaz, ale navíc poskytuje funkci JOIN, která zobrazí výsledky ze dvou SPARQL endpointů a přes zadaný predikát je spojí do jednoho výsledku. Poté můžeme využít i objekty v datové sadě představující odkaz na DBpedii a můžeme tak získat mnohem více dat.

4.4.1 Vizualizace SPARQL dotazu

V prvním příkladu se zaměříme na vizualizaci pouhého SPARQL dotazu. Ve všech následujících příkladech musíme nastavit pro graf [12] SPARQL endpoint [13].

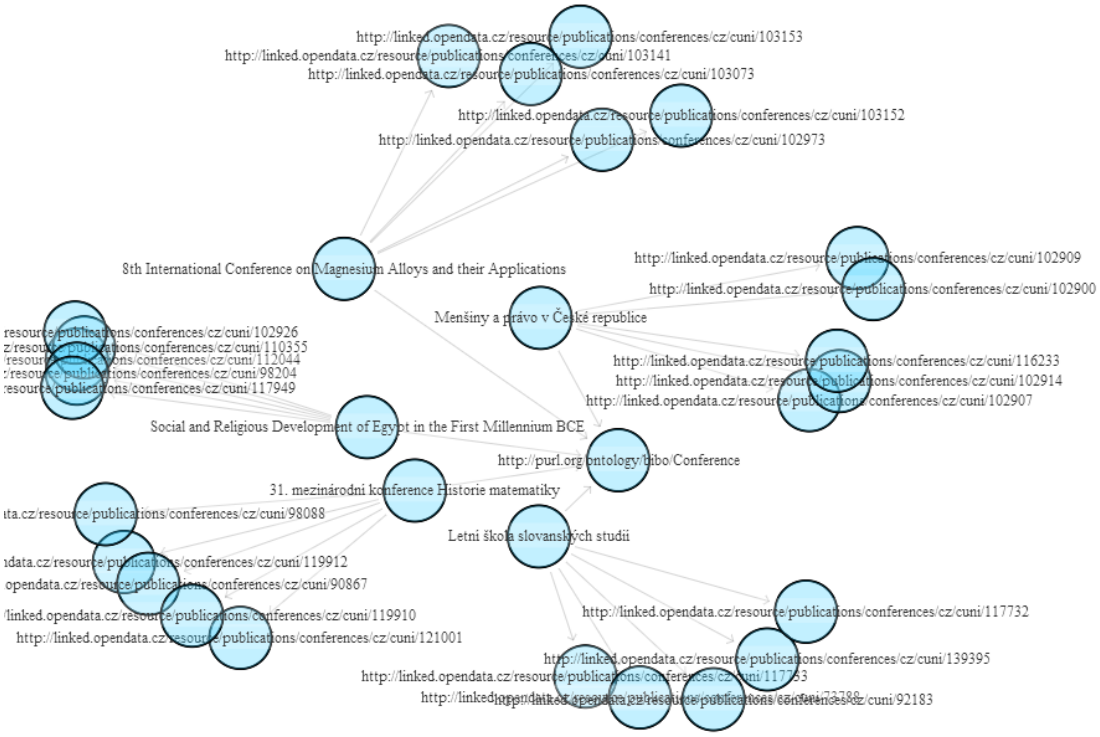
Ke každému záznamu může být uložena nejvýše jedna konference, ke které se záznam vztahuje. O konferenci máme informace pouze o názvu a době konání. Cílem následujícího SPARQL dotazu je sjednotit konference podle názvu tak, abychom dokázali zjistit, které záznamy se prezentovaly na dané konferenci.

```
prefix bibo: <http://purl.org/ontology/bibo/>
prefix dcterms: <http://purl.org/dc/terms/>
prefix owl: <http://www.w3.org/2002/07/owl#>
CONSTRUCT
{
        ?noveURI a bibo:Conference;
        dcterms:title ?konference;
```

[12] http://linked.opendata.cz/resource/dataset/bibliography/cz/cuni
[13] http://linked.opendata.cz:8890/sparql

```
        owl:sameAs ?uri;
        <http://linked.opendata.cz/property/pocetKonferenci> ?pocet.
}
WHERE
{
        {
              SELECT (IRI(encode_for_uri(?konference)) AS ?noveURI)
             (count(?konference) as ?pocet) ?konference
              WHERE {
                      ?uri a bibo:Conference;
                      dcterms:title ?konference.
              }
              GROUP BY ?konference
              HAVING (count(?konference) > 4)
              ORDER BY ?pocet
        }
        ?uri a bibo:Conference; dcterms:title ?konference.
}
ORDER BY desc(?pocet)
LIMIT 500
```

Rozbor dotazu začneme od vnitřního příkazu SELECT. Vytvoříme skupiny konferencí podle názvu a do výpisu zařadíme pouze ty, které mají alespoň 5 publikací, u kterých je konference s daným názvem přiřazena. Vnitřní SELECT vrátí název konference, počet záznamů a zjednodušený identifikátor vzniklý pouze z názvu konference. V následujícím vnějším dotazu připojíme všechny konference se stejným názvem v rámci vytvořené skupiny. Výsledek vypadá následovně:



Obrázek 4.2: Vizualizace klasického SPARQL dotazu

Pro znázornění v obrázku 4.2 byl dodatečně použit „`LIMIT 5`“ ve vnitřní části `SELECT` příkazu. Uprostřed diagramu vidíme typ `http://purl.org/ontology/bibo/Conference`, což je třída značící konference. K této entitě je přiřazeno 5 konferencí, které se dostaly na výstup.

- 31. mezinárodní konference Historie matematiky
- 8th International Conference on Magnesium Alloys and their Application
- Letní škola slovanských studií
- Menšiny a právo v České republice
- Social and Religious Development of Egypt in the First Millennium BCE

K vybraným konferencím jsou přiřazeny další konference se stejným názvem. Z URI přidružených konferencí, které jsou ve tvaru

`http://linked.opendata.cz/resource/publications/conferences/cz/cuni/ID`, zjistíme ID záznamu, který byl na této konferenci prezentován. Mírnou modifikací SPARQL dotazu bychom mohli také zjistit název publikací a další připojené informace.

Diagram znázorňuje pouze několik výsledků, tabulka 4.1 představuje 5 konferencí, na kterých bylo prezentováno nejvíce publikací Univerzity Karlovy:

Název konference	Počet prez. publ.
Week of doctoral students 2011	43
Week of Doctoral Students 2010	42
200 let Všeobecného občanského zákoníku	29
InterCorp 2009	21
Mathematical Methods in Economics 2010	15

Tabulka 4.1: Seznam konferencí a počtu prezentovaných publikací

Analýza je veřejně přístupná na stránkách projektu Payola pod název „Konference“ [14].

4.4.2 Propojení dat

V následujícím příkladu si ukážeme, jak můžeme propojit data ze dvou SPARQL endpointů za pomoci programu Payola. Nejprve vizualizujeme jednoduchý dotaz, v další kapitole dotaz ještě rozšíříme. Základním principem je propojení výsledných entit ze dvou SPARQL endpointů na základě stejných URI. Tento postup lze následně opakovat a propojit tak více datových sad (vždy dvojici výsledků spojíme do jednoho výsledku, který opět můžeme spojit s dalším). U každé publikace známe vydavatele, případně i město, kde byla publikace vydána. Ale nedokážeme určit zemi, ve které toto město leží. K tomu nám dopomůže datová

[14] `http://live.payola.cz/analysis/032be8f7-9c1f-4e57-ba0b-149a67ea2797` (20.5.2013)

sada DBpedie, která dokáže přiřazovat města do správných států. Následujícím SPARQL dotazem vybereme všechny vydavatele s městy, ke kterým je vydavatel připojen:

```
prefix dbpedia-owl: <http://dbpedia.org/ontology/>
prefix bibo: <http://purl.org/ontology/bibo/>
prefix foaf: <http://xmlns.com/foaf/0.1/>
prefix dcterms: <http://purl.org/dc/terms/>
CONSTRUCT {
       ?vydavatel a foaf:Organization;
       dbpedia-owl:hometown ?mesto.
}
WHERE {
       ?vydavatel a foaf:Organization;
       dbpedia-owl:hometown ?mesto.
}
limit 1000
```

Podobně ze SPARQL endpointu DBpedie [15] z daného grafu [16] vybereme města se státy. Toto můžeme vyřešit jednoduchým SPARQL dotazem, případně využít funkcionalitu Payoly.

`Typed: http://dbpedia.org/ontology/City` (vybereme všechna města)

`Property Selection: http://dbpedia.org/ontology/country` (přidáme predikát státu, ke kterému město patří)

`Join Property URI: http://dbpedia.org/ontology/hometown` (propojíme přes objekt města v prvním dotazu, který napojíme na entitu města)

K námi získaným datům můžeme takto navíc přidat informaci o státu, ve kterém se daná města nacházejí. Z obrázku 4.4 můžeme vyčíst údaje, že Vilnius je

[15] `http://dbpedia.org/sparql`
[16] `http://dbpedia.org`

litevské město a Budapešť je v Maďarsku. Výpis obsahuje ale všechny informace, které očekáváme, tedy i o Brnu, San Diegu a dalších městech, které na obrázku nejsou viditelné.

http://linked.opendata.cz/resource/publications/publisher/cz/cuni/Liga-lidských-práv	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/Brno
http://dbpedia.org/resource/Vilnius	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://dbpedia.org/ontology/City
	http://dbpedia.org/ontology/country	http://dbpedia.org/resource/Lithuania
http://linked.opendata.cz/resource/publications/publisher/cz/cuni/Salk-Institute,-La-Jolla,-San-Diego,-USA	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/San_Diego
http://linked.opendata.cz/resource/publications/publisher/cz/cuni/Mendelova-univerzita-Brno	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/Brno
http://linked.opendata.cz/resource/publications/publisher/cz/cuni/ESRI-Environmental-Sytems-Research-Institute	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/San_Diego
http://linked.opendata.cz/resource/publications/publisher/cz/cuni/Česká-vědecká-společnost-pro-mykologii	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/Brno
http://linked.opendata.cz/resource/publications/publisher/cz/cuni/ÚBO-AV-ČR	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/Brno
http://linked.opendata.cz/resource/publications/publisher/cz/cuni/Transport-Research-Centre	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/Brno
http://linked.opendata.cz/resource/publications/publisher/cz/cuni/Nadace-Universitas,-CERM,-NAUMA	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/Brno
http://linked.opendata.cz/resource/publications/publisher/cz/cuni/Ústav-Geoniky-AV-ČR---Katedra-geografie-PF-MU	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/Brno
http://linked.opendata.cz/resource/publications/publisher/cz/cuni/Urban	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/Brno
http://dbpedia.org/resource/Budapest	http://dbpedia.org/ontology/country	http://dbpedia.org/resource/Hungary
	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://dbpedia.org/ontology/City
http://linked.opendata.cz/resource/publications/publisher/cz/cuni/Technical-university-Brno,-Faculty-of-architecture.	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/Brno

Obrázek 4.3: Strukturovaný výpis vydavatelů

Entita vydavatele obsahuje ve vytvořené datové sadě pouze informaci o městě, ve kterém vydavatel publikoval alespoň jeden záznam, nyní dostáváme dodatečné informace o poloze města a není problém určit, že Brno je české město. Tento údaj pouze na základě informací obsažené v námi vytvořené datové sadě zjistit nelze.

Analýza je veřejně přístupná na stránkách projektu Payola pod název „Propojení datových sad“ [17].

[17] `http://live.payola.cz/analysis/2f4001ed-4e98-42b9-afc2-ab3caa5fa58b` (20.5.2013)

4.4.3 Propojení SPARQL dotazu s DBpedií

V posledním příkladu se vrátíme opět k vydavatelům. Nyní se zaměříme na 10 nejčastějších vydavatelů, u kterých si popíšeme i několik chyb. Nejdříve vydavatele rozřadíme do skupin podle jejich identifikátoru, následně zjistíme počet záznamů, které publikovali. A nakonec vypíšeme města, ve kterých vydali alespoň jednu publikaci. V druhém dotazu opět zopakujeme stejný postup jako u předešlé analýzy.

```
prefix foaf: <http://xmlns.com/foaf/0.1/>
prefix dbpedia-owl: <http://dbpedia.org/ontology/>
prefix dcterms: <http://purl.org/dc/terms/>
CONSTRUCT
{
       ?vydavatel a foaf:Organization;
       <http://linked.opendata.cz/property/pocet> ?pocet;
       dbpedia-owl:hometown ?mesto.
}
WHERE
{
       {
             SELECT ?vydavatel (count(?vydavatel) as ?pocet)
             WHERE
                     {
                     ?vydavatel a foaf:Organization.
                     ?zaznam dcterms:publisher ?vydavatel.
             }
             GROUP BY ?vydavatel
             ORDER BY DESC(count(?vydavatel))
             limit 10
```

```
        }
        ?vydavatel dbpedia-owl:hometown ?mesto
}
```

Druhý dotaz je naprosto stejný jako v předešlém případě, tedy

`Typed: http://dbpedia.org/ontology/City`

`Property Selection: http://dbpedia.org/ontology/country`

`Join Property URI: http://dbpedia.org/ontology/hometown`

Po spuštění analýzy bychom měli získat výsledky [18], které znázorňuje tabulka 4.2:

Vydavatel	Počet publikací
Karolinum	1120
Matfyzpress	472
Univerzita Karlova v Praze	431
Academia	393
Přírodovědecká fakulta Univerzity Karlovy v Praze	360
C.H.Beck	342
Springer	330
Linde	295
C. H. Beck	268
Nakladatelství Lidové novinky	199

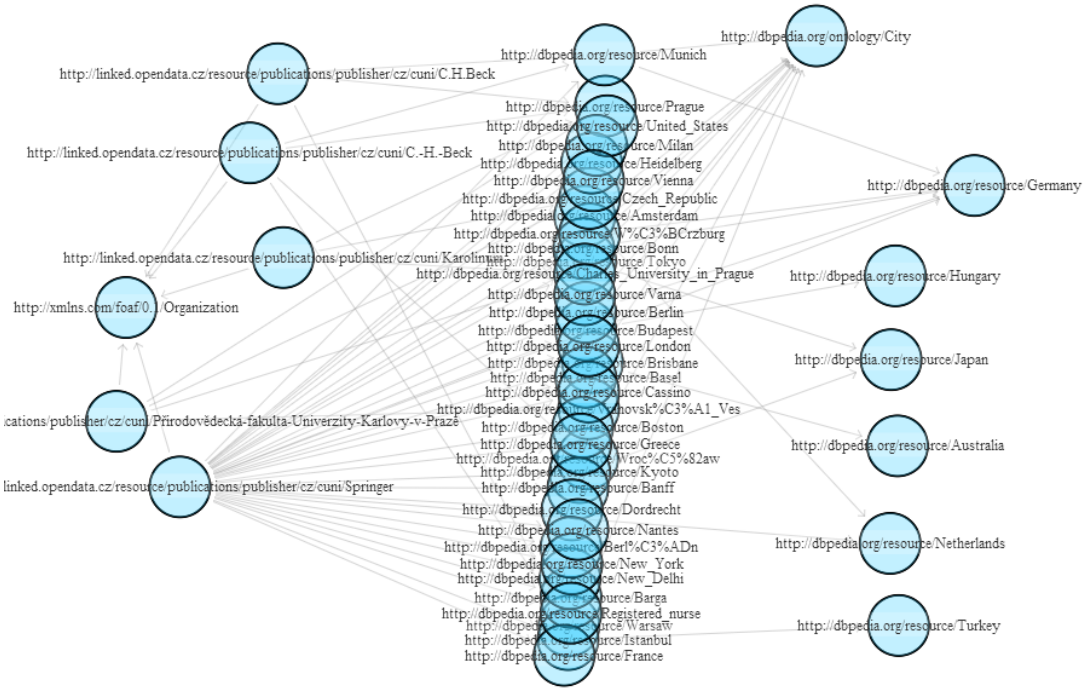
Tabulka 4.2: Nejčastější vydavatelé

Navíc si můžeme výsledek vizualizovat a ihned vidíme, u kterých států je větší počet měst, ve kterých se nachází alespoň jeden vydavatel s minimálně jednou publikací z Univerzity Karlovy. Výsledky vypadají následovně:

[18] Výsledky získané ze SPARQL endpointu `http://linked.opendata.cz:8890/sparql`

	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/Wroc%C5%82aw http://dbpedia.org/resource/Prague http://dbpedia.org/resource/Warsaw http://dbpedia.org/resource/Munich
	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://xmlns.com/foaf/0.1/Organization
http://dbpedia.org/resource/Istanbul	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://dbpedia.org/ontology/City
	http://dbpedia.org/ontology/country	http://dbpedia.org/resource/Turkey
http://dbpedia.org/resource/Amsterdam	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://dbpedia.org/ontology/City
	http://dbpedia.org/ontology/country	http://dbpedia.org/resource/Netherlands
http://linked.opendata.cz/resource/publications/publisher/cz/cuni/Přírodovědecká-fakulta-Univerzity-Karlovy-v-Praze	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/Prague http://dbpedia.org/resource/Charles_University_in_Prague http://dbpedia.org/resource/Vranovsk%C3%A1_Ves http://dbpedia.org/resource/Czech_Republic
	http://linked.opendata.cz/property/pocet	360
	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://xmlns.com/foaf/0.1/Organization
http://dbpedia.org/resource/Munich	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://dbpedia.org/ontology/City
	http://dbpedia.org/ontology/country	http://dbpedia.org/resource/Germany
http://linked.opendata.cz/resource/publications/publisher/cz/cuni/Karolinum	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/Charles_University_in_Prague http://dbpedia.org/resource/Prague http://dbpedia.org/resource/Czech_Republic
	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://xmlns.com/foaf/0.1/Organization
	http://linked.opendata.cz/property/pocet	1120
http://linked.opendata.cz/resource/publications/publisher/cz/cuni/C.H.Beck	http://linked.opendata.cz/property/pocet	342
	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://xmlns.com/foaf/0.1/Organization
	http://dbpedia.org/ontology/hometown	http://dbpedia.org/resource/Prague http://dbpedia.org/resource/Munich http://dbpedia.org/resource/Warsaw
http://dbpedia.org/resource/Kyoto	http://dbpedia.org/ontology/country	http://dbpedia.org/resource/Japan
	http://www.w3.org/1999/02/22-rdf-	http://dbpedia.org/ontology/City

Obrázek 4.4: Strukturovaný výpis vydavatelů



Obrázek 4.5: Grafické znázornění vydavatelů

Na obrázku 4.4 je viditelný výpis tabulkou se dvěma typy entit, města (`http://dbpedia.org/ontology/City`) a vydavatelé (`http://xmlns.com/foaf/0.1/Organization`). K vydavatelům jsou napojeny dvě vlastnosti, počet vydaných publikací a města, ve kterých byly publikace vydány.

Obrázek 4.5 graficky znázorňuje získané výsledky, na levé straně je znázorněno několik nakladatelství, která jsou napojena na města. Z DBpedie jsme poté získali další informace o státech, ve kterých tato města leží. Při srovnání předchozích obrázků s tabulkou 4.2 narazíme ale na několik lehce viditelných problémů.

- Aplikace OBD UK často posktyuje nevalidní data - městem vydavatele je často označena jiná entita, v příkladu jsou patrné „Charles_Univerzity_in_Prague“, „United_States“, „Greece“ a další.
- Data v aplikaci OBD UK se pro stejné údaje často mírně liší, viz vydavatel „C.H.Beck“ a „C. H. Beck“
- Aplikace Payola nerespektuje příkaz `limit 10`, když vrátila pouze 5 výsledků, které ani nejsou těmi nejčastějšími výskyty v dotazu

Analýza je veřejně přístupná na stránkách projektu Payola pod název „Vydavatelé ve státech“ [19].

Touto cestou si můžeme vytvářet podrobnější analýzy, můžeme zjistit, jestli zadaná data, která nám poskytla aplikace OBD UK, jsou validní. K podobným účelům využijeme vnější a vnitřní spojení a jednoznačně dokážeme určit, zda např. zadaná města jsou skutečně typu `http://dbpedia.org/ontology/City`.

[19] `http://live.payola.cz/analysis/38c8fed0-c23f-4c9d-8a70-5e2c56fbd084` (20.5.2013)

5. Uživatelská dokumentace

V následující kapitole se zaměříme na činnost přiloženého programu z hlediska uživatele. Nedílnou součástí uživatelské dokumentace by také mělo být, jak s vytvořenou datovou sadou pracovat, ale tento krok jsme si detailně popsali v předchozích kapitolách. Samotné ovládání je plně automatizované. Očekává se automatické spouštění aplikace tak, aby uživatel do jeho chodu zasahovat nemusel, případně výjimečně zkontroloval, zda program funguje správně a zda nedošlo k chybám na straně externích aplikací. Proto program pro plné fungování potřebuje přístup k Internetu. V této kapitole se zaměříme na to, jak může uživatel za použití přiložených zdrojových souborů datovou sadu kompletně vytvořit, aktualizovat a nahrávat na server Virtuoso.

5.1 Konfigurační soubory

Nejdůležitější části pro uživatele jsou konfigurační soubory a soubory z exportu. Všechny konfigurační soubory se nacházejí ve složce „conf“ hlavního adresáře. Program podporuje následujících pět konfiguračních souborů:

- ciselnikFormatu.conf - Tento soubor definuje, jak se budou přepisovat typy publikací na správné typy určené slovníky. Například formát software lze přepsat za pomoci slovníku Vivo Ontology a třídy Software. Pokud není nějaký typ publikace definován, potom se implicitně použije „bibo:Article“. Všechny typy, které se v datové sadě nacházejí, si můžeme projít v souboru output_stavy.txt, kde se kromě formátů také nacházejí stavy dokumentů. Toto nastavení bude použito pouze na nově zpracované záznamy.
- format.conf - Soubor popisující, které formáty chceme do datové sady zahrnout. Vztahuje se pouze na následující zpracování, existující datovou sadu případná změna neovlivní. Pokud je soubor prázdný, žádné omezení na for-

máty není. Na velikosti písmen v tomto případu záleží, vybírat můžeme z těchto formátů a každý formát, který chceme zahrnout, napíšeme na nový řádek:

- ABSTRAKT V ČASOPISU
- ABSTRAKT V ELEKTRONICKÉ PODOBĚ
- ABSTRAKT V KONFERENČNÍM SBORNÍKU
- AKADEMICKÁ PRÁCE
- AUDIOVIZUÁLNÍ TVORBA
- CERTIFIKOVANÁ METODIKA
- ČLÁNEK V ČASOPISU
- [FSV] Diskusní text
- [FSV] Prague Social Science Studies
- [FSV] Pražské sociálně vědní studie
- [FSV] Working paper
- JINÝ VÝSLEDEK
- KAPITOLA V KNIZE
- KNIHA
- Kniha pouze s editory (editor=autor)
- KONFERENCE, WORKSHOP
- LÉČEBNÝ nebo PAMÁTKOVÝ POSTUP
- Nezařazeno
- ODRŮDA, PLEMENO
- PATENT, UŽITNÝ VZOR, PRŮMYSLOVÝ VZOR
- POLOPROVOZ
- PROTOTYP, FUNKČNÍ VZOREK, OVĚŘENÁ TECHNOLOGIE
- PŘEDNÁŠKA NEBO POSTER
- PŘÍSPĚVEK V KONFERENČNÍM SBORNÍKU (abstrakty ne!)
- SOFTWARE
- SPECIALIZOVANÁ MAPA S ODBORNÝM OBSAHEM
- STAŤ VE SBORNÍKU PRACÍ
- VÝSLEDKY REALIZOVANÉ POSKYTOVATELEM
- VÝSTAVA

- osobniCisla.txt - Tento soubor existuje kvůli problému, kdy byla v aplikaci OBD UK změněná struktura exportu, která nám přidala k autorům osobní identifikační čísla na úkor informací ohledně pracovišť autorů. Proto jsme si uložili tato data ze starých souborů a v případě potřeby je lze využít.

- parametry.txt - V následující kapitole popíšeme, které třídy lze použít a s jakými parametry. Tento soubor nahradí použití parametrů u hlavní třídy, a pokud při spuštění program nedostane žádné parametry, použije parametry načtené ze souboru. Každý parametr je nutné psát na nový řádek.

  - Přihlašovací jméno do aplikace OBD UK
  - Heslo do aplikace OBD UK

  - Datum, od kdy chceme data stáhnout. Toto datum se dynamicky mění tak, abychom nestahovali duplicitní data. Program dále potřebuje datum, do kdy chceme data stáhnout, ale v případě načtení parametrů ze souboru se použije půlnoc předchozího dne. Datum je ve formátu den.měsíc.rok bez počátečních nul
  - Odkaz na SPARQL endpoint
  - Autentifikační údaje jméno
  - Heslo pro zabezpečený SPARQL endpoint
  - HTTP IRI grafu

- stav.conf - Zde platí stejné podmínky jako u povolených formátů, rozdíl je pouze v tom, kdy nám správce aplikace OBD UK pro MFF povolil do výsledné datové sady zahrnout pouze PŘIJATÉ publikace, pro interní použití ale lze použít jakékoli stavy z následujících:
  - Rozpracovaný
  - K autorizaci
  - Vrácený
  - Uložený
  - Ke kontrole
  - Kontrolovaný
  - PŘIJATÝ
  - Schválený pracovištěm
  - Schválený sekcí
  - PŘIJATÝ (nepublikovat)

5.2 Spuštění aplikace

Pokud chceme stáhnout data z aplikace OBD UK a aktualizovat s nimi graf, můžeme přímo využít dvou základních možností. Nejjednodušší možností je spuštění přiloženého souboru prace.exe [1], který nejdříve stáhne potřebná data (vzhledem k parametrům načteným z konfiguračního souboru), aktualizuje RDF dokument a nakonec aktualizuje publikovaný graf. Tento spustitelný soubor musí mít právo na zápis do souborů output_..., které se nacházejí ve stejné složce. Konfigurační soubory se nacházejí ve složce „conf“.

[1] Přiložené exe soubory lze spustit pouze na Windows (98/Me/NT/2K/XP/Vista/7/8)

Druhou možností je spuštění JAR [2] souboru z příkazového řádku, kde platí stejné poznámky jako u spustitelného souboru. Z příkazového řádku můžeme využít [3]

```
java -jar prace.jar
```

Poslední možností je zkompilování dané třídy, kterou budeme potřebovat, případně využít vývojové prostředí NetBeans [4].

Pro správnou funkčnost aplikace je nutné myslet na kódování konfiguračních souborů, které aplikace používá.

Konfigurační soubory jsou připravené tak, aby aplikace byla možná spustit z vývojového prostředí NetBeans za pomoci přiložených zdrojových kódů a knihoven v operačním systému Windows 7 [5]. Pro tento případ je použité kódování UTF-8 (bez BOM) [6]. Pro případ spouštění z příkazového řádku Windows 7 musí být konfigurační soubory překonvertovány do kódování ANSI [7]. Zároveň je také nutné myslet na samotné používání kódování aplikace pro správnou práci s diakritikou.

Na CD jsou přiloženy spustitelné soubory „prace.exe“ (stažení dat z aplikace OBD UK, modifikace RDF dokumentu a publikovaného grafu) a „zpracovaniXML.exe“ (převod souboru „xml.xml“ do RDF dokumentu). Současně jsou připojeny i soubory prace.jar a zpracovaniXML.jar, který s parametrem cesty k souboru zpracuje daný soubor.

[2] JAR - Java Archive File

[3] `http://www.oracle.com/technetwork/java/javase/downloads/index.html`

[4] `https://netbeans.org/`

[5] `http://windows.microsoft.com/cs-cz/windows7/products/home`

[6] UTF-8 - UCS (Universal Character Set) Transformation Format 8 bit - bez BOM - Byte Order Mark, což je sekvence bytů pro správnou detekci kódování

[7] ANSI - American National Standards Institute

5.3 Minimální požadavky aplikace

Program stáhne data z aplikace OBD UK, aktualizuje data v RDF dokumentu i veřejně přístupném grafu, je tedy nutný přístup k Internetu. Jelikož zpracovává pouze interní příkazy, neposkytuje GUI [8] a uživatel běh programu neovlivní. Aplikace si v průběhu zpracování potřebuje načíst soubory s uloženými daty, proto je vyžadována minimální paměť 100 MB. Pokud nemůžeme tuto paměť zajistit, je nutné tyto soubory odstranit, nebudeme si ale udržovat kompletní RDF dokument v jednom souboru, na výsledný publikovaný graf tento krok nebude mít žádný vliv.

5.4 Automatické aktualizace

Následující část je popisována pro operační systém Windows 7, pokud budeme chtít naplánovat automatické aktualizace veřejného grafu.

1. Klikneme na tlačítko Start.
2. Vybereme položku Ovládací panely.
3. Pokračujeme tlačítkem Systém a zabezpečení.
4. Klikneme na položku Nástroje pro správu.
5. Nakonec použijeme nástroj Plánovač úloh.

Případně lze nástroj Plánovač úloh otevřít z příkazového řádku příkazem „Taskschd.msc“. Postup pro vytvoření automatického spuštění aplikace je následující.

- V menu zvolíme vytvoření nové úlohy

[8] Graphical User Interface

- Zvolíme novou aktivační událost pro začátek úlohy „Po přihlášení“
- Nastavíme novou akci pro úlohu „Spustit program“ s parametrem „prace.exe“

Následně se bude graf aktualizovat vždy po přihlášení uživatele.

5.5 Používané třídy

Program byl navržen tak, aby mohl být používán bez lidského přístupu kvůli automatickým aktualizacím veřejného grafu. Přesto ale mohou nastat případy, kdy člověk bude chtít tuto aplikaci spravovat. Aplikace nejdříve stáhne data z aplikace OBD UK, následně data převede do validního RDF formátu a nahraje na server Virtuoso. Toto zajistí třída „Download“, která vyžaduje 8 parametrů (přihlašovací jméno a heslo do aplikace OBD UK, datum od a do kdy chceme data stáhnout, URI SPARQL endpointu s autentifikačními údaji jménem a heslem a HTTP IRI grafu). Použití těchto parametrů není povinné. Pokud nejsou parametry zadány, použijí se implicitní parametry ze souboru conf/parametry.txt.

5.6 Netypické chování aplikace

Aplikace ve většině případů nepotřebuje lidský zásah a funguje plně automaticky, problémy mohou nastat tehdy, když tato aplikace nezíská validní data z aplikace OBD UK v prvním kroku a druhý krok ohlásí chybu. Exporty bohužel neposkytují vždy plně validní výsledky a místo toho nám poskytují chybovou hlášku. Toto náš program ještě dokáže zachytit a stáhne si data z méně dnů. Pokud ale stahujeme pouze jeden den a export není validní, následně ani přiložená aplikace není schopná pokračovat. Data z OBD jsme stahovali přes 5 měsíců a tento problém se vyskytl u pěti dnů, byl nahlášen přímo autorům aplikace

OBD UK, bohužel zatím nebyl opraven. Veškeré chybové hlášky jsou dostupné v souboru „conf/log.txt“.

Jediným řešením je manuální stažení souboru, jelikož víme, který den nejde stáhnout automaticky. Z tohoto důvodu je nutné data stáhnout manuálně a také omezit podmínky pro exportované záznamy (v tomto případě zřejmě postačí změnit stav pouze na „PŘIJATÝ“, resp. v souladu s konfiguračními soubory). Následně je nutné spustit třídu ZpracovaniXML s parametrem značícím relativní (případně absolutní) cestu k souboru, který jsme stáhli a který chceme následně zpracovat.

Poté máme přístup k vygenerovaným datům v souboru output_2.ttl, což je soubor, který obsahuje především nově získaná data v datové sadě. Tento soubor ještě potřebujeme nahrát do Virtuosa, což lze provést buď manuálně přes prohlížeč, případně použít třídu UpVirtuoso se čtveřicí parametrů - URI na zabezpečený SPARQL endpoint, autentifikační údaje jméno a heslo a HTTP IRI grafu.

5.7 Kompletní vytvoření celého grafu

V závěru této kapitoly se zaměříme na problém, jak postavit celý graf z XML souborů, které máme k dispozici, což je potřebné v případě větších změn ve struktuře dat. Potřebujeme tři typy souborů [9]. Prvním a největším souborem je export poskytnutý správcem aplikace OBD UK (Mgr. Janem Popelkou), který obsahuje export všech záznamů do 22.11.2012 (včetně). Následně potřebujeme další dva druhy souborů. Soubory, které jsme získali před změnou části exportu, kdy exporty obsahovaly informace o pracovištích. Posledním typem souborů jsou současné exporty, které tyto informace nemají, ale obsahují naopak identifikační číslo oso-

[9] Zmíněné soubory nejsou přiložené na CD z důvodu používání neveřejných dat (záznamů, které nejsou publikovány). Data lze získat přímo z aplikace OBD UK, případně od autora nebo vedoucího práce.

by v rámci Univerzity Karlovy. Proto potřebujeme využít existující konfigurační soubor s identifikačními čísly souborů. Pokud ale tento soubor neexistuje, lze ho vytvořit za pomoci třídy ZpracovaniStarsichSouboru, která ze všech novějších souborů posbírá informace o ID záznamu, jménu a příjmení autora a identifikačním číslu.

K vytvoření validního souboru využijeme již popsanou třídu ZpracovaniXML, kterou spustíme na největší soubor s XML daty. Poté potřebujeme udělat menší změnu ve zdrojovém kódu třídy XML, kde odstraníme z poznámky třídu KodyAutoru a v konstruktoru třídy XML odkomentujeme načtení dat společně s náhradou osmimístného identifikačního čísla UK v části ukončení elementu (funkce endElement -> case "kodautora"). Poté třídu ZpracovaniXML použijeme u všech dostupných XML souborů, kde ze starších souborů takto získáme navíc údaje ohledně identifikačních čísel autorů a jejich pracovišť (za předpokladu existujících záznamů ve starších i novějších exportech, resp. jejich výskytu v konfiguračním souboru). Tato část je uvedena v poznámce z důvodu možného použití, není však součástí kódu z důvodu nízkého využití a zpomalení aplikace.

6. Programátorská dokumentace

V poslední kapitole naší práce se zaměříme na to, které části jsou důležité pro programátora, jaké všechny třídy jsou potřebné pro tuto aplikaci a jaké knihovny jsme použili. Popis začneme od třídy, která je používána pro kompletní aktualizaci grafu, budeme postupovat přes třídy, které jsou volány a ukončíme třídami, které nejsou pro aplikaci potřebné, ale během postupné práce na projektu jsme je potřebovali, proto jsou zachovány i v konečné verzi.

6.1 Třída Download

Tato třída obstará všechny tři podstatné kroky. Nejdříve se pokusí stáhnout data z internetové aplikace OBD UK, následně je převede a vytvoří RDF dokument output.ttl a output_2.ttl obsahující nová data, který v posledním kroku nahraje na server Virtuosa.

Tato třída obsahuje dvě funkce:

- **public static void main(String[] args)** počítá s osmi parametry, které mohou být zadané přímo na vstupu, nebo uložené v konfiguračním souboru, přesnou podobu parametrů jsme si popsali v části 5.1. Tato funkce pouze rozdělí jednotlivá data tak, aby mezi nimi byl maximální rozdíl dvou týdnů, protože pro větší rozdíl často aplikace OBD UK překročí časový limit pro vygenerování exportu (který je 1 510 sekund) a s danými daty zavolá následující funkci.
- **private static void download(String jmeno, String heslo, Calendar c1, Calendar c2, String sparql, String jmeno2, String heslo2, String graf)** požaduje parametry přihlašovací jméno a heslo do aplikace OBD UK, datum od a do kdy chceme data exportovat, URI SPARQL endpointu s autentifikačními údaji a HTTP IRI grafu. Funkce využije třídu

Update, které předá dvě konkrétní data pro stáhnutí a ostatní parametry. Pokud je vygenerování exportu neúspěšné (v tomto případě stažený soubor neexistuje, nebo obsahuje pouze „`<zaznamy></zaznamy>`“), potom interval pro stažení exportu rozdělíme na poloviny, je-li to možné. Následně předá požadavky třídě Update.

6.2 Třída Update

- **public static void main(String args[])** přijímá naprosto stejné parametry jako funkce main předchozí třídy, stáhne zadaná data z aplikace OBD UK a předá soubor k dalšímu zpracování, pokud existuje a obsahuje více dat než pouhé `<zaznamy></zaznamy>`. Ke stažení dat používáme prohlížeč HtmlUnit[9], jedná se o volně šiřitelný prohlížeč určený pro programovací jazyk Java bez žádné podpory GUI. Java příkazy napodobují chování uživatele v klasickém prohlížeči, to znamená, že vybereme např. pátý odkaz na stránce, na který chceme kliknout. Samotná syntaxe je velmi podobná prostředí DOM [1] pro zpracování XML dokumentů. Vytvoření kompletního kódu pro Javu je opět velmi snadné, lze využít doplněk pro prohlížeč Mozilla Firefox [2] HtmlUnit Scripter [3] (opět volně šiřitelný), který v prohlížeči aktivujeme, projdeme webové stránky tak, jak chceme, aby to dělala i tato funkce, a následně vygenerujeme výsledný kód. Tento výsledný kód ale ještě nemusí být zcela funkční, protože si tento doplněk neumí poradit s rámci, které aplikace OBD UK používá, a proto tento krok musíme manuálně nastavit. Jinak ale nastavení přihlašovacích údajů a filtrovacích kritérií provádí tak, jak očekáváme. Druhým problémem bylo samotné vygenerování výsledků, protože po kliknutí na tlačítko „Hledat“ musí aplikace počkat několik (většinou desítek) vteřin než se data vygenerují a zobrazí uživateli, na což prohlížeč HtmlUnit nereagoval. Tento problém jde vyřešit klasic-

[1] DOM - Document Object Model
[2] `http://firefox.mozilla.cz/`
[3] `http://code.google.com/p/htmlunitscripter/`

kým uspáním aplikace na tři minuty, poté jsou již data jistě připravena ke stažení.

HtmlUnit Scripter následně vygeneruje kód, kde bychom měli zmínit především jeden parametr „savePagesLocally“, který při hodnotě `true` ukládá náhled webových stránek do určené složky a můžeme zjistit, na jaké stránce a v jakém stavu se aktuálně prohlížeč nachází. Pokud se podařilo soubor stáhnout a obsahuje nějaká data, poté se pokračuje ve zpracování zavoláním hlavních funkcí tříd ZpracovaniXML a SparqlAuth.

6.3 Třída ZpracovaniXML

- **public static void main(String[] args)**, metoda vyžaduje jeden parametr určující soubor, který bude zpracovávat. Pokud není parametr zadán, implicitně zpracuje soubor xml.xml [4]. Následné zpracování provede třída „XML“, která představuje rozhraní SAX [5] pro zvolený XML dokument.

6.4 Třída XML

Třída XML implementuje rozhraní SAX, což znamená, že je vždy vyvolána událost při otevření a ukončení elementu, počátku i konci dokumentu, další existující události pro zpracování nejsou až tak důležité.

Mezi nejdůležitější atributy třídy patří seznamy autorů, záznamů, konferencí, knih, vydavatelů, povolených stavů a formátů, dále dvourozměrné seznamy fakult (s klíčem představující zkratku fakulty), pracovišť (s klíčem ID pracoviště). Ve všech seznamech si udržujeme veškerá načtená data, která dostaneme buď z předchozích zpracování, nebo ze zpracování aktuálního XML dokumentu.

Základní metody třídy XML jsou následující:

[4] Není součástí CD, jedná se o XML data poskytnutá správcem aplikace OBD UK
[5] SAX - Simple API for XML, API - Application Programming Interface

- **public XML()**, konstruktor třídy, ve kterém se do paměti načtou údaje o povolených formátech a stavech. Současně se načte i číselník pro převod formátu do správné třídy pro RDF dokument. V poznámce je zároveň možnost povolení načtení již existujících kódů autorů, pokud bychom chtěli zpracovat i starší dokumenty.

- **public void startDocument()**, na začátku dokumentu si program načte již existující data ze souborů začínajících output_... .txt, které obsahují data z dříve zpracovaných XML dokumentů. Tento krok není bezpodmínečně nutný, ale rozhodli jsme se, že i v těchto souborech se budeme snažit vyhýbat duplicitním datům, proto je tento krok součástí funkce. Načtená data představují informace o autorech, konferencích, fakultách, pracovištích, knihách, stavech, vydavatelích, záznamech a přiřazení pracovišť k osobám u jednotlivých záznamů.

- **public void endDocument()**, na konci dokumentu potřebujeme veškerá načtená data opět zapsat do stejných souborů, ze kterých jsme je načetli. Samotný proces převodu dat je popsaný v kapitole 3. Po ukončení převodu vytvoříme soubor output.ttl, který vznikne sloučením dříve vyjmenovaných souborů. Liší se pouze tím, že na začátku doplníme prefixy slovníků a také nahradíme některé znaky typu > znakem > z důvodu toho, že aplikace OBD UK podobné znaky v XML exportu používat nemůže. V TTL dokumentech se tyto znaky na správných místech vyskytovat mohou. Tento krok opakujeme ještě jednou a do souborů lišících se v názvu koncovkou „_2“ zapíšeme převážně nově načtené entity a entity, ve kterých došlo k editaci dat. Soubory podobně sloučíme do jednoho většího souboru, který je určen pro nahrání na server Virtuoso.

- **public void startElement(String uri, String localName, String qName, Attributes atts)**, metoda pro zpracování začátku elementu. Funkčnost je rozdělena podle názvu elementu a typicky proběhne např. při elementu <začátku elementu autor přiřazení nového objektu do atributu autor třídy XML, veškeré důležité zpracování probíhá až při ukončení elementu.

- **public void endElement(String uri, String localName, String qName)**, metoda reagující na ukončení elementu. Opět je funkčnost rozdělena podle názvu elementu, základním krokem je ověření toho, zda daný element odpovídá datům, která chceme zařadit na výstup a zároveň, jestli se jedná o novou, či již existující entitu. Příkladem uvedeme ukončení elementu autor. Ve skutečnosti autoři jsou děleni na více typů, ať už samotný autor, editor, ale také vedoucí práce a další. V práci jsme zpracovávali pouze autory a editory, ostatní typy do výpisu nezařadíme. Následně zkontrolujeme, jestli načtený kód autora odpovídá dříve existujícímu autorovi, pokud ne, přidáme autora do seznamu autorů, pokud ano, potom u autorů můžeme ještě editovat titul a jeho pracoviště a podle toho se rozhodneme, jestli autora přeskočíme, nebo ho budeme chtít editovat a následně uložit na server Virtuoso. Obecně by program fungoval takto, ale při načtení autora ještě nevíme, zda bude záznam odpovídat správnému stavu a formátu, proto autory nekontrolujeme okamžitě, ale až při zjištění dostatečného množství údajů, podle kterých lze jednoznačně rozhodnout, zda autory budeme ve výpisu potřebovat, nebo ne. Podobně musíme zkontrolovat i ostatní elementy a také element typu záznam, zda je správného formátu a stavu vůči nastavení z konfiguračních souborů.

- **public void characters(char[] ch, int start, int length)**, metoda, která ve většině programů bývá implementována podobně. Problémem je, že veškerý textový obsah nemusí být doručen v rámci jednoho volání, proto potřebujeme postupně načítat obsah do globální proměnné, ze které získáme data až při ukončení elementu. Při začátku elementu vždy tuto proměnnou potřebujeme vyprázdnit.

Třída XML obsahuje často seznamy dalších tříd (autoři, ...), které si následně představíme. Všechny následující třídy obsahují true/ false atribut, který určuje, zda budeme chtít entitu nahrát na server, nebo ne. Typicky je při načtení ze starších dat nastaven na false, při načtení z aktuálního XML dokumentu na true (případně při editaci již existující entity je změněn z false na true).

6.4.1 Třída Autor

Seznam atributů:

- Jméno autora
- Příjmení autora
- Typ autora, v řešení této práce ukládáme pouze autory typu „Autor“ a „Editor“
- Osobní osmimístné číslo CAS UK, pokud export toto číslo nenabízí, poté jako identifikátor použijeme jméno a příjmení autora
- Seznam titulů autora
- Seznam pracovišť, u kterých je autor veden jako autor/editor jakékoli publikace

6.4.2 Třída Konference

Seznam atributů:

- Název konference
- Datum začátku konference
- Datum konce konference
- Seznam míst, na kterých se konference konala, typicky se jedná o jedno místo, ale kvůli napojení na DBpedii můžeme získat pro jedno místo více odkazů, např. na mezinárodní a na českou verzi

6.4.3 Třída Pracoviste

Seznam atributů:

- Zkratka pracoviště
- Zkratka fakulty, na které se pracoviště nachází
- Celý název pracoviště
- Celý název fakulty

6.4.4 Třída Kniha

Tento název není naprosto vystihující. Ke každému záznamu z aplikace Verso UK může být připojen údaj, který značí, zda byl daný záznam zahrnut jako součást větších děl, např. v knihách, sbornících a dalších typech publikací.

Seznam atributů:

- ID prezentace, které odpovídá ID záznamu, ke kterému prezentace patří
- Titul prezentace
- Rok, kdy byla prezentace uskutečněna

6.4.5 Třída Titul

Každý záznam obsahuje stejné informace, které mohou být přeloženy do několika různých jazyků.

Seznam atributů:

- Název publikace

- Popis (abstrakt) publikace
- Klíčová slova přiřazená k publikaci
- Jazyk, ve kterých jsou tato data napsána

6.4.6 Třída Vydavatel

Seznam atributů:

- ID vydavatele
- Název vydavatelství
- Seznam měst, ve kterých byly publikace tohoto vydavatele publikovány

6.4.7 Třída Zaznam

Hlavní třída, která obsahuje veškeré informaci o dané publikaci.

Seznam atributů:

- Seznam autorů, ID, typ autora (zda se jedná o autora, nebo editora) a seznam pracovišť, ve kterých v době vydání publikace pracuje [6]
- Seznam titulů v alespoň jednom jazyce
- ID prezentace a konference
- ISBN, ISSN, URL publikace
- Počáteční, koncová strana a rozsah
- Vydavatel
- Literární forma a stav

[6] pracoviště nejsou poskytována v novějších typech exportů

6.5 Další třídy

Poslední použité třídy se vztahují k jazykům, městům (které jsme museli z XML exportu mírně upravit) a dodatečným funkcím. Také jsme během implementace projektu použili několik tříd, které ve finální verzi nejsou potřebné, ale není vyloučeno jejich pozdější použití, proto zůstávají nedílnou součástí konečné verze projektu.

6.5.1 Třída Jazyky

Tato třída obsahuje pouze jediný důležitý atribut jazyky, což je číselník určující přepsání jazyku z „cze“ na „cs“, což je validní zkratka České republiky dle normy ISO 639-1. Tento číselník je při vyvolání konstruktoru načten z konfiguračního souboru seznam_jazyku.txt. Pokud by přesto byl načten prozatím neexistující jazyk, bude uložen jako neznámý a je třeba přiřadit zkratku. Tento krok bohužel nelze automatizovat, protože data z exportů nebyla zcela validní, mezi daty lze najít ku příkladu jazyk „mul“, což ve skutečnosti ale byla angličtina.

6.5.2 Třída Města

Třída Města představuje také číselník pro jejich přepis na existující stránky DBpedie. Číselník je opět načten z externího souboru seznam_mest.txt, oproti minulému číselníku se liší hlavně tím, že v číselníku pro jednu hodnotu můžeme přiřadit více dat, např. odkazy na českou a mezinárodní verzi DBpedie. Při načtení nového města je tento krok plně automatizován. Pokud program načte město „Prague“, zjistí, zda existuje stránka `http://dbpedia.org/resource/Prague`. V tomto případě uspěje. Uspěl by i v případu „Praha“, pro který bude stránka přesměrována na již zmíněnou stránku. Ale u příkladu „Prha“ je zřejmé, o kterém městě se zmiňujeme, ale tento problém již musí být opraven manuálně.

6.5.3 Třída MojeFunkce

Třída obsahující pouze statistické funkce potřebné v průběhu zpracování souboru

Existující metody:

- **static public boolean validniURI(String uri)** zjistí, zda se jedná o validní URI. V tomto kroku byla použita třída URI a také UrlValidator veřejně dostupné knihovny Apache [7]. Dva kroky byly použité z důvodu větší jistoty, protože URI nesmí obsahovat mezery a znaky typu „[“, což ale obě třídy nezvládaly stoprocentně, jejich průnik již ale vrací dostatečně spolehlivé výsledky.
- **static public boolean tryParseInt(String value)** zjistí, zda lze vstupní řetězec konvertovat do číselného typu.
- **static public String zdrojovyKod(String url)** funkce, která vrátí zdrojový kód zadané HTTP URI stránky.
- **static public boolean existujeWebStranka(String URLName)** zjistí, zda existuje zadaná webová stránka. V této funkci je zakázáno přesměrování na jinou stránku.
- **static public String existujeWebStranka2(String URLName)** funkce, ve které je oproti minulé naopak přesměrování povoleno a je vrácena URI stránky po případném přesměrování.
- **static public String prevodData(String datum)** funkce, která převede datum z formátu DD.MM.RRRR do formátu RRRR.MM.DD (počáteční nula není vyžadována ani na vstupu).

[7] `http://commons.apache.org/`

6.6 Nahrání souboru na server Virtuosa

V následující části si představíme poslední tři třídy, které nám zajišťují upload do Virtuosa. Všechny třídy mají jednu statickou „main“ funkci, která samotný upload zajistí, ale liší se ve způsobu provedení.

6.6.1 Třída SparqlAuth

Tato třída řeší nahrání prostřednictvím zabezpečeného SPARQL endpointu. Samotné nahrání imituje prohlížeč použitím externí knihovny HtmlUnit, proto potřebuje 4 parametry na vstupu, URI endpointu, autentifikační údaje (jméno a heslo) a HTTP IRI grafu. Na server Virtuosa jsou nahraná pouze nově vzniklá data ze souboru output_2.ttl. V tuto chvíli je tato třída nepoužívaná.

6.6.2 Třída UpluoadVirtuoso

Tato třída přijímá stejné parametry jako předchozí, je implementována za pomoci externí knihovny HtmlUnit a simuluje přihlášení přes Conductor a nahrání souboru output_2.ttl přes „Quad Store Upload“. Tato třída je momentálně nepoužívaná.

6.6.3 Třída UpVirtuoso

Poslední způsob je používán v současné verzi programu. Využívá volně dostupnou externí knihovnu Jena [8], která zajistí upload přímo přes SPARQL endpoint a neimituje prohlížeč jako předchozí příklady dvou tříd. Hlavní metoda přijímá opět 4 stejné atributy popsané v prvním případě. Pro upload se využije soubor output_2.ttl, který obsahuje pouze data určená pro upload.

[8] `http://jena.apache.org/`

6.7 Možnosti rozšíření

Jelikož funkčnost aplikace silně závisí na aplikaci OBD UK, je velmi pravděpodobné, že několikrát dojde ke změně exportu vypisovaných dat, proto bude nutné provést drobné zásahy do zdrojového kódu pro stažení dat. V době dokončování práce také čekáme na realizaci několika požadavků v aplikaci OBD UK, v exportu například chybí informace o pracovištích autora, které zde bývaly, proto bude následně nutné stáhnout všechna data znovu a celý graf aktualizovat s těmito informacemi.

Touto prací jsme vytvořili datovou sadu, která je zaměřená pouze na publikace, proto by bylo možné rozšířit datovou sadu o další informace vztahující se k Univerzitě Karlově - organizační strukturu univerzity, rozvrhy, zaměstnance, studenty a ostatní podobné informace.

Také nebylo cílem této práce navrhnout aplikaci nejefektivnějším způsobem, snažili jsme se o co největší přehlednost v uložených datech, proto především načítání již uložených dat by bylo možné (z hlediska času) vylepšit. Pokud ale potřebujeme získaná data pouze udržovat v grafu, je možné načítané soubory odstranit a tím výrazně urychlit fungování programu.

Datová sada byla navržena jako alternativní možnost pro zobrazování informací o záznamech vzhledem k oficiální aplikaci OBD UK. Datová sada je volně přístupná na stránkách projektů OpenData.cz a Data Hub, dále by bylo možné práci rozšířit do podoby webového projektu s podobnými filtrovacími kritérii, které nabízí OBD UK.

7. Obsah CD

Na přiloženém CD jsou dostupné

- Zdrojové kódy (NetBeans 7.2.1 project)
- Použité externí knihovny
- Elektronická podoba této práce
- Aplikace pro převod dat
- Výsledná datová sada

Závěr

V této práci jsme si představili základní principy pro naprosto odlišný způsob práce s daty oproti klasickým HTML stránkám, seznámili jsme se se základními principy pro tvorbu RDF dokumentů s použitím principů Linked Data. Zaměřili jsme se na vytvoření vlastní datové sady o vydaných publikacích a její publikování pro neomezený přístup, představili jsme si možnosti pro dotazování nad vzniklou datovou sadou i s možností propojení s existujícími datovými sadami, především DBPedií.

Současně předpokládáme, že exporty aplikace OBD UK se mohou v budoucnu měnit, proto bude potřebné provést menší změny i do přiložené aplikace. Předpokládáme, že aplikace bude spuštěna každý den pro aktualizaci publikované datové sady a případné vzniklé problémy budou odstraněny, protože fungování přiloženého programu silně závisí na externích aplikacích.

Touto prací jsme zapojili Univerzitu Karlovu do projektu Linked Universities, která sdružuje evropské univerzity poskytující data ve formě propojených dokumentů. Publikovaná datová sada je přístupná na stránkách projektů OpenData.cz a Data Hub.

Zároveň jsme si popsali, jak pracovat s datovou sadou mnohem efektivněji a rychleji než v oficiální aplikaci OBD UK. Filtrovací kritéria, která v aplikaci OBD UK vyžadují několik minut čekání, dokážeme z naší datové sady získat mnohem rychleji a nejsme omezeni nadefinovanými funkcemi, můžeme vytvářet vlastní kritéria a s pomocí aplikace Payola také vlastní grafické analýzy.

Seznam použité literatury

[1] ALEXANDER, Keith, et al. *Describing linked datasets with the void vocabulary.* W3C Interest Group Note, W3C, 2011. [cit. 2013-05-18]. Dostupné z WWW: `http://www.w3.org/TR/void`

[2] AUER, Sören, et al. *Dbpedia: A nucleus for a web of open data. In: The semantic web.* Springer Berlin Heidelberg, 2007. p. 722-735. ISBN 978-3-540-76297-3.

[3] BECKETT, Dave; MCBRIDE, Brian. *RDF/XML syntax specification (revised).* W3C recommendation, 2004. [cit. 2013-05-18]. Dostupné z WWW: `http://www.w3.org/TR/REC-rdf-syntax`

[4] BERNERS-LEE, Tim, et al. *The semantic web. Scientific american.* 2001, 24.5: 28-37. [cit. 2013-05-18]. Dostupné z WWW: `http://isel2918929391.googlecode.com/svn-history/r347/trunk/RPC/Slides/p01_theSemanticWeb.pdf`

[5] BERNERS-LEE, Tim, et al. *Linked data-the story so far. International Journal on Semantic Web and Information Systems.* 2009, 5.3: 1-22. [cit. 2013-05-18]. Dostupné z WWW: `http://eolo.cps.unizar.es/Docencia/doctorado/Articulos/LinkedData/bizer-heath-berners-lee-ijswis-linked-data.pdf`

[6] BIRON, Paul, et al. *XML schema part 2: Datatypes. World Wide Web Consortium Recommendation REC-xmlschema-2-20041028.* 2004. [cit. 2013-05-18]. Dostupné z WWW: `http://www.w3.org/TR/2000/CR-xmlschema-2-20001024`

[7] BIZER, Chris; CYGANIAK, Richard; HEATH, Tom. *How to publish linked data on the web.* 2008. [cit. 2013-05-18]. Dostupné z WWW: `http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/LinkedDataTutorial`

[8] BOARD, DCMI Usage. *DCMI metadata terms*. Retrieved March, 2008, 10: 2008. [cit. 2013-05-18]. Dostupné z WWW: http://dublincore.org/documents/dcmi-terms

[9] BOWLER, Mike. *HtmlUnit*. 2002. [cit. 2013-05-18]. Dostupné z WWW: http://htmlunit.sourceforge.net/

[10] BRAY, Tim, et al. *Extensible markup language (XML)*. World Wide Web Journal, 1997, 2.4: 27-66. [cit. 2013-05-18]. Dostupné z WWW: http://www.w3pdf.com/W3cSpec/XML/2/REC-xml11-20060816.pdf

[11] BRICKLEY, Dan; GUHA, Ramanathan V. *Resource Description Framework (RDF) Schema Specification 1.0: W3C Candidate Recommendation*. 27 March 2000. [cit. 2013-05-18]. Dostupné z WWW: http://www.w3.org/TR/2000/CR-rdf-schema-20000327

[12] BRICKLEY, Dan; GUHA, Ramanathan V. *RDF Vocabulary Description Language 1.0: RDF Schema*. 2004. [cit. 2013-05-18]. Dostupné z WWW: http://www.w3.org/TR/rdf-schema/

[13] BRICKLEY, Dan; MILLER, Libby. *FOAF vocabulary specification 0.98*. Namespace Document, 2010, 9. [cit. 2013-05-18]. Dostupné z WWW: http://ontogenealogy.com/documents/2012/08/foaf-vocabulary-specification-0-98-20100809.pdf

[14] D'ARCUS, B.; GIASSON, F. *Bibliographic Ontology Specification*. Specification Document, 4 November 2009. Retrieved August, 2009, 10: 2011. [cit. 2013-05-18]. Dostupné z WWW: http://bibliontology.com/specification

[15] HOBBS, Jerry R.; PAN, Feng. *Time ontology in OWL*. W3C working draft, 2006, 27. [cit. 2013-05-18]. Dostupné z WWW: http://www.w3.org/TR/owl-time/

[16] KEßLER, Carsten; D'AQUIN, Mathieu; DIETZE, Stefan. *Linked Data for science and education*. Semantic Web, 2013, 4.1: 1-2. [cit. 2013-

05-18]. Dostupné z WWW: http://iospress.metapress.com/content/b24426587514113x/fulltext.html

[17] KLÍMEK, Jakub; HELMICH, Jiří, NEČASKÝ, Martin. *Payola: Collaborative Linked Data Analysis and Visualization Framework*. In 10th Extended Semantic Web Conference (ESWC 2013), Montpellier, France, May 2013.

[18] MILES, Alistair; BECHHOFER, Sean. *SKOS Simple Knowledge Organization System Namespace Document-HTML Variant*. World Wide Web Consortium (W3C) Recommendation, 2009. [cit. 2013-05-18]. Dostupné z WWW: http://www.w3.org/TR/skos-reference/

[19] PRUD'HOMMEAUX, Eric, et al. *SPARQL query language for RDF*. W3C recommendation, 2008, 15. [cit. 2013-05-18]. Dostupné z WWW: http://www.w3.org/TR/rdf-sparql-query/

[20] QUILITZ, Bastian; LESER, Ulf. *Querying distributed RDF data sources with SPARQL*. In: The Semantic Web: Research and Applications. Springer Berlin Heidelberg, 2008. p. 524-538. [cit. 2013-05-18]. Dostupné z WWW: https://www.ki.informatik.hu-berlin.de/wbi/research/publications/2008/DARQ-FINAL.pdf

[21] RAIMOND, Yves; ABDALLAH, Samer. *The event ontology*. Technical report, 2007. [cit. 2013-05-18]. Dostupné z WWW: http://motools.sourceforge.net/event/event.html

[22] RAIMOND, Yves; ABDALLAH, Samer. *The Timeline Ontology*, 2007. [cit. 2013-05-18]. Dostupné z WWW: http://motools.sourceforge.net/timeline/timeline.html

[23] SEABORNE, Andy, et al. *Sparql update, a language for updating rdf graphs*. 2008. [cit. 2013-05-18]. Dostupné z WWW: http://www.hpl.hp.com/techreports/2007/HPL-2007-102.pdf

[24] SERVER, Virtuoso Universal. *OpenLink Software*. 2010. [cit. 2013-05-18]. Dostupné z WWW: http://virtuoso.openlinksw.com/

[25] STYLES, Rob; SHABIR, Nadeem. *Academic Institution Internal Structure Ontology (AIISO)*. 2008. [cit. 2013-05-18]. Dostupné z WWW: http://vocab.org/aiiso/schema

[26] WIKIPEDIA. *DBpedia - Wikipedia, The Free Encyclopedia*. 2013. [cit. 2013-05-18]. Dostupné z WWW: http://en.wikipedia.org/w/index.php?title=DBpedia&oldid=545874300

Seznam tabulek

Představa RDF trojic 1.1

Příklad RDF trojic 1.2

Převod dat autorů 3.1

Převod dat stavů a formátů 3.2

Převod dat fakult 3.3

Převod dat knih 3.4

Převod dat konferencí 3.5

Převod dat pracovišť 3.6

Přiřazení pracovišť k autorům 3.7

Převod dat vydavatelů 3.8

Převod dat záznamů 3.9

Seznam konferencí a počtu prezentovaných publikací 4.1

Nejčastější vydavatelé 4.2

Seznam použitých zkratek

ANSI - American National Standards Institute, národní kódování, např. CP-1250 pro český jazyk

BOM - Byte Order Mark, sekvence bytů pro správnou detekci kódování

CAS UK - Centrální autentizační služba Univerzity Karlovy

DOM - Document Object Model, objektově orientovaná reprezentace XML

DTD - Document Type Definition, jazyk pro popis XML dokumentu

FOAF - Friend Of A Friend, slovník určený pro popis osob

GUI - Graphical User Interface, grafické uživatelské rozhraní umožňující ovládat aplikaci

HTML - HyperText Markup Language, značkovací jazyk pro publikaci dokumentů na Internetu

HTTP - HyperText Transfer Protocol, internetový protokol pro výměnu hypertextových dokumentů ve formátu HTML

ID - IDentifikační číslo

IRI - Internationalized Resource Identifier, mezinárodní identifikátor v rámci Internetu v kódování UCS

ISSN - International Standard Serial Number, mezinárodní standardní číslo sériové publikace

ISBN - International Standard Book Number, mezinárodní standardní číslo knihy

JAR - Java ARchive, archivní soubor pro sjednocení více tříd programovacího jazyka Java

JSON - JavaScript Object Notation, JavaScriptový objektový zápis určený pro přenos dat

KSI - Katedra Softwarového inženýrství

LOD - Linking Open Data, komunita zabývající se publikováním datových sad

MFF - Matematicko - fyzikální fakulta

N3 - Notace trojic pro RDF formát

OBD UK - Evidence a zpracování publikační činnosti pracovníků a studentů Univerzity Karlovy

RDF - Resource Description Framework, standardy sémantického webu

SAX - Simple API for XML (Application Programming Interface), zpracování XML soubory založené na vyvolaných událostech

SPARQL - Sparql Protocol And RDF Query Language, dotazovací jazyk pro RDF dokumenty

SQL - Structured Query Language, dotazovací jazyk pro relační databáze

TTL - Terse RDF Triple Language, formát zápisu RDF dat

UCS - Universal Character Set, standardní množina znaků

UK - Univerzita Karlova

URI - Uniform Resource Identifier, řetězec znaků pro identifikaci zdroje informací

UTF-8 - UCS Transformation Format 8 bit, způsob kódování znaků s proměnnou délkou znaku

W3C - World Wide Web Consortium, mezinárodní konsorcium spravující webové standardy

XML - eXtensible Markup Language, značkovací jazyk určený pro přenos dat