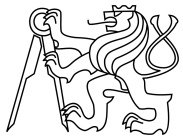
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Efektivní dotazování nad databází PubChem
Student:	Bc. Jan Valášek
Vedoucí:	RNDr. Jakub Galgonek, Ph.D.
Studijní program:	Informatika
Studijní obor:	Znalostní inženýrství
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Cílem práce je prozkoumat efektivitu vybraných přístupů a technologií umožňujících vyhodnocování dotazů v jazyce SPARQL na RDF datech pocházejících z projektu PubChemRDF.

1) Vytvořte reprezentativní sadu dotazů pro testování, která bude pokrývat typické dotazy pro databázi malých molekul.
2) Uložte data v nativní RDF formě do hybridní (relační i RDF) databáze Virtuoso, proveďte analýzu různých nastavení a vhodně zvolte dodatečné indexy pro dosažení co nejlepšího výkonu.
3) Převeďte původní RDF data do vhodné relační formy a vytvořte mapování této formy na původní RDF formu. Dotazování má probíhat v jazyce SPARQL, který je pomocí mapování automaticky převáděn na dotaz v jazyce SQL, který je následně nad relačními daty vyhodnocen.
4) Porovnejte efektivitu dotazování v SQL přímo nad relačními daty, ve SPARQL nad nativními daty a ve SPARQL nad mapovanými daty.
5) Srovnejte efektivitu se zvolenou relační databází (například PostgreSQL) umožňující mapovat relační data do RDF formy.

Seznam odborné literatury

https://pubchem.ncbi.nlm.nih.gov/rdf/https://pubchem.ncbi.nlm.nih.gov
http://docs.openlinksw.com/virtuoso/
https://www.w3.org/RDF/
https://www.w3.org/TR/sparql11-query/
https://www.elixir-czech.cz

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 7. února 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Diplomová práce

Efektivní dotazování nad databází PubChem

Bc. Jan Valášek

Vedoucí práce: RNDr. Jakub Galgonek, Ph.D.

9. ledna 2018

Poděkování

Děkuji RNDr. Jakubu Galgonkovi, Ph.D. že věnoval svůj čas a znalosti vedení mé práce. Také děkuji rodině za podporu, a organizacím CERIT-SC a ELIXIR CZ za poskytnuté prostředky pro realizaci celé práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 9. ledna 2018 .....

Odkaz na tuto práci

Valášek, Jan. *Efektivní dotazování nad databází PubChem.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato práce se zabývá způsoby a technologiemi vyhodnocování dotazů v jazyce SPARQL na rozsáhlých RDF datech pocházejících z projektu PubChemRDF. Součástí publikace je popis jednotlivých nástrojů, použitých optimalizací, měření výkonu a celkové zhodnocení.

Klíčová slova databáze, SPARQL, RDF, OWL, r2rml mapování, grafová databáze, postgreSQL, měření výkonu

Abstract

This thesis examines different ways and technologies for evaluating SPARQL queries upon extensive RDF data provided by PubChemRDF porject. Decription of tested tools, optimizations, performance and overall evaluation are also a part of this publication.

Keywords database, SPARQL, RDF, OWL, r2rml mapping, graph database, postgreSQL, performance

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Správa a využívání dat nejrůznějšího charakteru se stává velmi klíčovou potřebou pro stále více uživatelů, organizací a firem. Uložiště dat, a nástroje pro práci s nimi, je potřeba zvolit, navrhnout a dimenzovat dle konkrétních potřeb daného subjektu. Od fyzických, papírových záznamů ukládaných do šanonů, přes klasické relační databáze až po rozsáhlé datové sklady. O tom, který ze způsobů zvolit rozhodují faktory jako například

- množství ukládaných dat - jedná se o desítky záznamů či stovky milionů?
- frekvence - archivují se data pouze jednou měsíčně jako report či jsou generována několikrát za sekundu?
- důvod ukládání dat - Pro vlastní potřebu, pro splnění právních norem, business analýzy...
- operace, které nad daty potřebujeme vykonávat - historizace, vyhledávání, agregace, MDM, data quality, sdílení, archivace..
- finanční rozpočet, technické a personální zázemí subjektu
- očekávaný přínos - marketing, efektivita procesů, podpora rozhodování, splnění právních norem

Tato práce se věnuje problematice efektivního ukládání a využívání dat z oblasti chemie (databáze PubChem) v Ústavu organické chemie a biochemie Akademie Věd ČR (ÚOCHB). V textu se zabývám zejména prací s relačními (tabulkovými) databázemi a grafově orientovanými databázemi, respektive způsoby zpracování „Linked Dat“[6] (kde jsou data přímo vzájemně propojena pomocí vazeb).

Každý typ má své výhody i nevýhody, a je vhodný pro jiný druh dat a jinou kategorii dotazů.

Relační a grafové databáze

Tradiční relační databáze se skládají ze (vzájemně propojených) databázových tabulek. Řádky v tabulce představují jednotlivé záznamy, sloupce pak atributy těchto záznamů (například „jméno“, „věk“).

Klasické relační databáze jsou efektivní především pokud předem známe strukturu zpracovávaných dat, a lze ji dobře popsat tabulkami s řádky a sloupci[19]. Dotazy, které vyžadují často vyhodnocovat vazby mezi více tabulkami (join operace), mohou být výpočetně velmi náročné a někdy je ani nelze provádět v reálném čase, ale pouze dávkově (s výsledky dotazu v řádu minut nebo hodin).

Na druhou stranu grafové databáze mají data uložena v podobě grafu, tedy uzlů a vztahů mezi nimi (například formát RDF[1]) [5]. Díky této struktuře je snažší provádět operace využívajících vazeb mezi daty. Nejsou již potřeba náročné join operace nad celými tabulkami (pokud jsou data uložena ve vhodné struktuře), ale stačí pouze „následovat hranu“ mezi dvěma záznamy.

Grafové databáze jsou vhodné především tam, kde dopředu neznáme přesně strukturu dat, nebo tam, kde nás zajímají různé vztahy mezi daty. V grafových databázích jsou informace o vztazích přímo součástí hlavních entit (hran grafu), a lze tedy přidávat nové uzly i vztahy bez nutnosti upravovat stávající strukturu databáze (přidávají se nové uzly a hrany do stávajícího grafu). Tento druh uložení dat je vhodný především pokud potřebujeme procházet data v závislosti na jejich vazbách, objevovat nová spojení a celkově využívat jejich grafovou strukturu[13]. Jedna z hlavních výhod se skrývá také v lepší interoperabilitě mezi více datovými zdroji. Jednotlivé entity jsou označovány identifikátorem IRI (Internationalized Resource Identifier), aby bylo možné sledovat vazby i do dalších databází a jiných zdrojů dat. Pokud jsou v obou systémech použity stejné IRI pro danou entitu, či je lze snadno vzájemně převést, usnadní to propojení dat z těchto systémů.

Bohužel jsou grafové databáze stále poměrně nové, a tak nemusí nutně zaručovat standardy, na které jsme zvyklí z relačních databází. Jako například transakčnost, ACID principy, obnovu po havárii[13]... Navíc jsou některé grafové databáze implementovány na backendu jako několik relačních tabulek, takže ve výsledku stejně pro procházení grafu probíhají klasické relační join operace.

Možnost mapování dat

Další zkoumanou možností, jak s daty pracovat, je originální grafová data (například ve formátu RDF[5]) převést do relační podoby. Tedy z grafové podoby navrhnout (nebo nechat automaticky vygenerovat) databázovou strukturu tvořenou tabulkami a rozdělit zdrojová data do jednotlivých tabulek,

[1] RDF = Resource Description Framework

které jsou vzájemně propojeny pomocí cizích klíčů. Například z grafu extrahovat data o lidech, a uložit je do jedné tabulky, údaje o smlouvách do druhé atp. Pro tato relační data je pak možné vytvořit takzvané mapování[10], umožňující k nim přistupovat i přes grafové SPARQL dotazy. Data jsou tak uložena v tabulkách v relační podobě (lze využívat například indexace tabulek), ale máme možnost se na ně dotazovat jako kdyby byla stále v grafové podobě SPARQL dotazy. Mapování umožní tyto dotazy přeložit do SQL podoby, a vykonat je nad relační databází.

To se může hodit například v situacích, kdy jsou zdrojová data v RDF formátu, a je potřeba k nim zachovat přístup přes rozhraní pro SPARQL dotazy, ale efektivnější je uchovávat tyto data v relační databázi. Nebo pokud máme zdrojová data v relační databázi, ale potřebujeme s uloženými daty pracovat jako s grafovými.

Ukázka tohoto přístupu je vidět na obrázcích 1.2, 0.1 a 0.2.

O projektu PubChem

Projekt PubChem[15] shromažďuje a poté veřejně poskytuje podrobné informace o chemických sloučeninách (Compounds), chemických látkách (Substances) a experimentech na biologických vzorcích (Bioassays). Navíc poskytuje další informace o vztazích mezi těmito základními složkami, a o vazbách na externí systémy dalších vědeckých institucí. Data z interních databází systému PubChem jsou dostupná dvěma různými způsoby. Jeden způsob je s těmito daty pracovat online, přes webové rozhraní a využívat vestavěné vyhledávací nástroje[15], zároveň je však možné díky projektu PubChemRDF[16] stáhnout zdrojová data v RDF formátu.

Cíl práce

Tato práce navazuje na prototyp vytvořený v rámci ÚOCHB a cílem je prozkoumat, jak lze efektivně uložit data poskytovaná prostřednictvím projektu PubChemRDF, a jak s nimi nejlépe pracovat. Ideálně s možností vyhodnocovat dotazy v jazyce SPARQL[9] na těchto rozsáhlých datech.

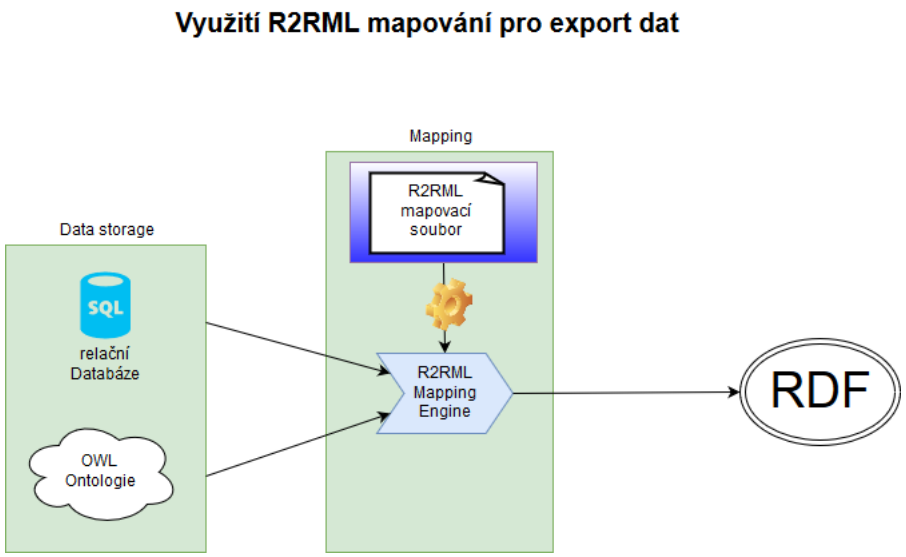
Pro měření výkonnostních rozdílů je třeba vytvořit reprezentativní sadu testovacích dotazů. Tato sada bude pokrývat typické dotazy pro databázi malých molekul.

Jeden ze zkoumaných přístupů spočívá v uložení původních RDF dat přímo do hybridní databáze Virtuoso[20] (v grafové podobě). U této varianty je potřeba provést analýzu různých nastavení a vhodně zvolit dodatečné indexy pro dosažení co nejlepšího výkonu.

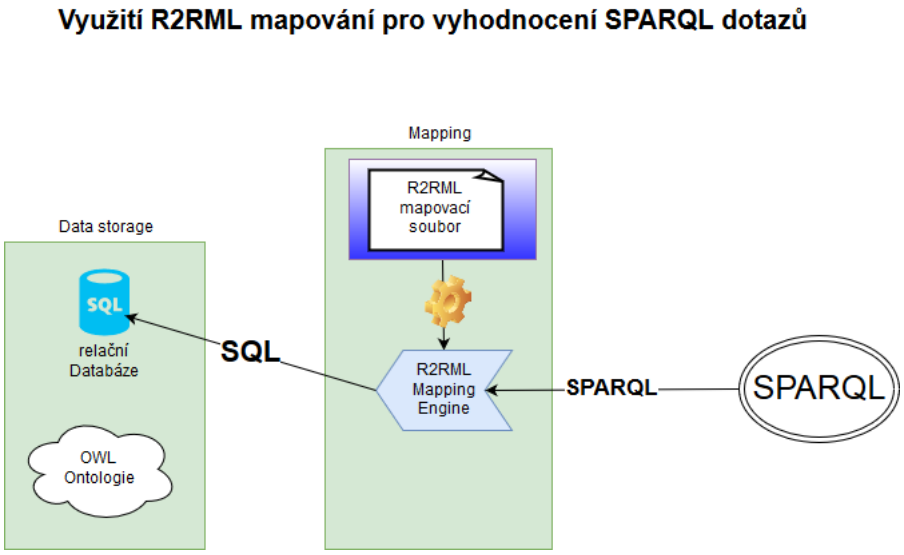
Další přístup spočívá v převedení dat z originální RDF podoby do vhodné relační formy (tedy tabulky, funkce) a poté namapování této relační formy zpět na původní RDF formu. Tedy mít data uložena v relační podobě, ale s možností dotazovat se na ně i v jazyce SPARQL0.2 (pro grafové databáze).

SPARQL je pomocí mapování automaticky převáděn na dotaz v jazyce SQL (pro relační databáze), který je následně nad relačními daty vyhodnocen.

Výsledkem práce bude srovnání efektivity dotazování v SQL přímo nad relačními daty, ve SPARQL nad nativními daty a ve SPARQL nad mapovanými daty. A na závěr srovnání efektivity s další relační databází (například postgreSQL) umožňující mapovat relační data do RDF formy.



Obrázek 0.1: Schéma průběhu mapování relačních dat do RDF podoby během exportu



Obrázek 0.2: Využití R2RML mapování při online překladu dotazu ze SPARQL jazyka do SQL

Kapitola 1

Analýza a návrh

1.1 Forma dat, a možnosti práce s nimi

1.1.1 Data jako nositel informace

Důležité informace a poznatky z výzkumu je užitečné a potřebné uchovávat, archivovat a aktualizovat. O tom, jak nakládat s informacemi získanými během výzkumu a experimentů, je potřeba rozhodnout ideálně již na začátku, a zvolit patřičné datové uložiště a nástroje pro práci s ním. Rozhodnutí by mělo reflektovat zejména množství získaných informací, zdroje informací (odkud, v jakém formátu), a další využitelnost těchto informací (kdo a jak často je bude potřebovat, jakým způsobem s nimi bude pracováno, zda budou aktualizována). Pokud chceme získané údaje pouze archivovat, stačí nám uložit informace v nějakém jednoduchém formátu (například textové soubory, multimédia, adresáře...). Když budou data ale dále využívána, je potřeba tomu přizpůsobit především jejich strukturu. Zejména pokud budeme chtít v datech například efektivně vyhledávat informace, propojovat jednotlivé záznamy, hledat souvislosti. Dále uvádím několik způsobů jak s daty pracovat, a které v závěrečné práci zkoumám.

1.1.2 Relační podoba dat

Relační databáze je model, který **ukládá data do** jednotlivých, vzájemně provázaných, **tabulek**. Dnes je naprostá většina používaných databází relační[14], často se tedy pojem „databáze“ používá jako synonymum k „relační databázi“.

Každá tabulka v relační databázi se skládá ze sloupců (atributů) a řádků (jednotlivých záznamů) viz příklad tabulky 1.1.

Tabulka 1.1: ukázka relační tabulky

ID	Title	description
653499	LGALS7B	galectin 7B
44409	Cbp80	cap binding protein 80
729979	LOC729979	hypothetical LOC729979
...	...	...
...	...	...
...	...	...

V ukázkové tabulce je znázorněna sada údajů o třech genech. Každý gen je jednoznačně identifikován pomocí svého ID (první sloupec - ID). Dále je v tabulce uložena ke každému genu informace o jeho názvu (druhý sloupec - title), a popisu daného genu (třetí sloupec - description). Každý sloupec je vždy nějakého konkrétního typu, a podle toho se s ním pak zachází. Sloupec může podle typu uchovávat textové řetězce, číselné hodnoty, datum, ale i například celý soubor.

V relačních databázích se často využívá také takzvaných **primárních a cizích klíčů**. Primární klíč (tzv. Primary key, PK) jednoznačně identifikuje konkrétní instanci/záznam z databázové tabulky (například rodné číslo v tabulce osob). Primární klíč musí být unikátní v rámci celé tabulky, a nesmí mít NULL (nevyplněnou) hodnotu. Primární klíč obvykle bývá unikátní číselná hodnota, ale může to být například i textový řetězec, či dokonce složený primární klíč z několika sloupečků dohromady (například adresa + jméno). A pokud se chceme z jedné tabulky „odkázat“ na nějakou instanci v jiné tabulce, použijeme k tomu cizí klíč (tzv. Foreign Key, FK). Cizí klíč v jedné tabulce je v podstatě hodnota primárního klíče v druhé tabulce. Pomocí cizího klíče vytvoříme tedy „spojení“ z první tabulky do druhé tabulky.

Primární a cizí klíče slouží pro efektivní práci s daty. Například pro rychlejší vyhledávání v tabulce či k tomu, aby nebylo nutné duplikovat data ve více tabulkách, ale stačilo se přes cizí klíč odkázat na jednu tabulku, ve které se potřebná data nachází.

Standardní způsob práce s daty v relační databázi probíhá pomocí **SQL dotazů** (Structured Query Language). Se SQL dotazy lze vytvářet, upravovat a mazat tabulky, vkládat do tabulek data, načítat je i mazat. Základní dotaz pro výběr dat z tabulky je ve formátu:

```
SELECT sloupec1 sloupec2 ....
FROM tabulka1 ...
WHERE sloupec1 > 5
```

V dotazu je příkaz SELECT, popisující ze kterých sloupečků chceme čist data, lze zadat i „*“ pokud chceme hodnoty ze všech sloupců. Příkaz FROM říká,

ze kterých tabulek chceme požadovaná data. WHERE je podmínka či filtr, popisující, které záznamy chceme. Pokud daný záznam podmínce vyhovuje, je vrácen, pokud nevyhovuje, je ignorován. V tomto příkladu například dostaneme pouze ty záznamy, kde je hodnota ve sloupci s názvem „sloupec1“ větší než 5.

Pomocí SQL příkazů lze tabulky i vytvářet či s nimi jinak pracovat a měnit jejich strukturu. Následující jednoduchý příkaz vytvoří tabulku s názvem „tabulka3“, skládající se ze tří sloupců. Každý sloupec musí být nějakého typu (například číselná hodnota).

```
CREATE TABLE tabulka3
(sloupec1 typSloupce1,
sloupec2 typSloupce2,
sloupec3 typSloupce3);
```

Pokud máme ve více tabulkách sloupce se stejnými názvy, rozlišíme je přidáním názvu tabulky. Například **zamestnanci**.jmeno a **zakaznici**.jmeno.

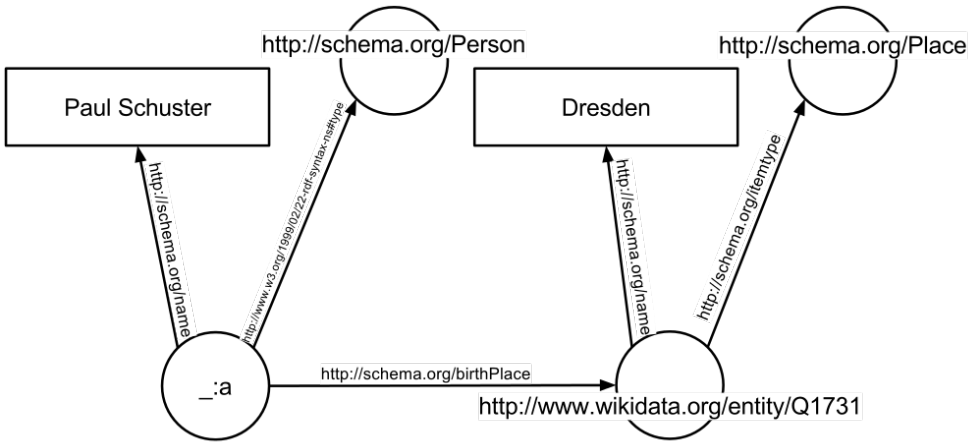
1.1.3 Sémantický web

Organizace W3C definuje sémantický web jako „web pro dat“, na rozdíl od klasického „webu pro dokumenty“[6]. Cílem tohoto přístupu je především dosáhnout možnosti efektivnějšího počítačového zpracování dat na webu a jejich interaktivního propojování. Sémantický web prolinkovává (propojuje) vzájemně data pomocí IRI a umožňuje lidem vytvářet datové struktury, slovníky a pravidla pro využívání těchto dat. Jelikož jsou v praxi IRI často velmi dlouhé, pro zkrácení zápisu IRI je možné definovat si takzvaný „prefix“, kterým můžeme nahrazovat společnou část IRI.

Linkování dat slouží především k zefektivnění strojové práce s daty, zachycení různých závislostí mezi nimi, a celkové integrace dat s jejich kontextem[7]. Například v jedné databázi jsou uložená data o chemických sloučeninách a malých molekulách. V další databázi (pod správou jiné instituce) jsou zase informace o patentech, normách, vědeckých článcích atp. Pokud je některá sloučenina patentována, lze u této sloučeniny mít pouze IRI patřičného patentu, a po připojení databáze patentů propojit snadno na data o patentech. Rovněž pokud se o některé sloučenině psalo v článku či na konferenci, stačí nám přístup k těmto datům a použití stejného slovníku, aby šlo data správně napojit.

Hlavní důvod, proč toto na klasickém webu jednoduše nejde provést, spočívá v tom, že web je tvořen především aplikacemi, nikoliv samotnými daty. Tyto aplikace potřebná data sice využívají a pracují s nimi, ale „nechávají si je pro sebe“[7]. Tudíž nelze příliš snadno propojovat informace z různých systémů, a navzájem je kombinovat.

Klíčovým prvkem sémantického webu jsou dva základní předpoklady[7]. Za prvé společný formát pro integraci a kombinaci dat z různých datových



Obrázek 1.1: Příklad propojení RDF dat na sémantickém webu. Autor: Denny Vrandečić

zdrojů. Na rozdíl od klasického webu, kde výměna dat probíhá obvykle na úrovni celých dokumentů. A druhým předpokladem je společný jazyk pro popis závislostí mezi daty a reálným světem. Tak lze začít zpracovávat data v jedné databázi, a pokračovat napříč různými dalšími prolinkovanými databázemi díky jejich propojení.

Například v případě rozbitého mobilního telefonu má tento telefon svůj typ. Typ telefonu je propojen s výrobcem telefonu, výrobce telefonu je propojen s autorizovanými servisními místy, servisní místo má svou adresu a otvírací dobu. Tímto způsobem lze na základě dat projít několika systémy (prodejce telefonu, výrobce, autorizovaného servisu) a automaticky zjistit kam může zanést zákazník mobilní telefon do servisu.

Pokud se použije například standardizovaný slovník FOAF[23], kde se entita osoby označuje instancí z domény <foaf:Person>, jméno osoby <foaf:name>, email jako <foaf:mbox> a podobně, je možné zajistit interoperabilitu mezi systémy. Díky těmto standardům můžeme snáze rozpoznat, že osoba v jednom systému je shodná s osobou ve druhém systému. V případě sémantického webu umožní seznam takovýchto pravidel automatické zpracování a vyhledávání informací napříč systémy.

Naproti tomu u klasického webu je pro toto potřeba, aby nám dané informace a linky zpřístupnil ve své aplikaci každý poskytovatel obsahu (jednotlivý článek v řetězu). Navíc pokud už potřebné informace zveřejní, dělá to nyní každý jiným způsobem, obvykle nevhodným pro strojové zpracování. Některé firmy mají kontakt na hlavní straně webu, některé v záložce „kontakty“, některé v záložce „o nás“ atp. Priorita u této formy poskytování informací je totiž aby byly čitelné a snadno dohledatelné člověkem, nikoliv strojovým zpracováním. A přístup přímo k samotným datům v relační databázi by nám

v tomto případě také nepomohl, protože nevíme, v které tabulce a přes které vazby potřebná data najít. Ani jednotlivé databáze nejsou navzájem nijak propojeny, takže nemáme jak pokračovat z jedné databáze do druhé (V databázi prodejce najdeme podle typu například výrobce telefonu, ale už nemáme informaci, kde se nachází databáze výrobce, a kde v ní jsou potřebné tabulky).

Základním stavebním prvkem sémantického webu je Resource Description Framework, **RDF**, standard pro popis a výměnu dat na webu. Jeho základní součástí jsou trojice subjekt-predikát-objekt, tzv. triplety, popisující v podstatě vazbu zdroj-vazba-cíl[5]. Nad daty v této podobě se potom můžeme dotazovat v jazyce SPARQL[9], což je obdoba SQL dotazovacího jazyka používaného v prostředí relačních databází. Jazyk SPARQL ale využívá konceptu grafu a vztahů mezi daty. Výsledkem dotazu mohou být další RDF grafy či přímo tabulka výsledků.

Jazykem standardně používaným pro popis věcí, skupin a jejich vzájemných vazeb v prostředí sémantického webu je tzv. Web Ontology Language (**OWL**) vyvíjený W3C[8].

1.1.4 RDF podoba dat

Resource Description Framework (RDF[5]) je standard pro ukládání a výměnu sémantických dat v prostředí internetu, Využívá linkové struktury podobné webovým adresám (URL), takzvané **IRI** (Internationalized Resource Identifier).

RDF model má vlastnosti, které mimo jiné usnadňují slučování a propojování dat z různých zdrojů (nevyžaduje shodné schéma uložení dat v obou zdrojích)[5]. Navíc velmi dobře podporuje zejména postupný vývoj schémat, bez nutnosti jakýchkoliv úprav na straně žadatele o data.

Základním stavebním prvkem jsou takzvané triplety, neboli trojice: „**subjekt - predikát - objekt**“. Díky této struktuře je možné data reprezentovat i v grafové podobě, kde má predikát funkci orientované hrany od uzlu „subjekt“ do uzlu „objekt“. Jednoduchý triplet představující vztah mezi dvěma entitami může vypadat například takto: **„person:Petr“ - „relationship:friendship“ - „person:Karel“**. Což znázorňuje, že se Petr přátelí s Karlem, neboli osoba:Petr má vztah:přátelství s osoba:Karel.

Tento základní model umožňuje práci jak se strukturovanými, tak pouze částečně strukturovanými daty. Data lze také snáze kombinovat, vystavovat a sdílet mezi více aplikacemi. Struktury složené z jednotlivých uzlů a orientovaných hran vytváří orientovaný a popsaný graf (uzly a hrany jsou popsané IRI). **Hrany v tomto grafu potom reprezentují nějaký druh spojení mezi dvěma uzly**[5].

Pokud chceme například osobě kromě jména a příjmení zaznamenat i místo trvalého pobytu, tak jednoduše přidáme jeden triplet s touto informací.

Dále je také možné poměrně jednoduše propojovat různé zdroje dat na základě společných či obecně známých slovníků. Pro entitu z jednoho zdroje dat

(databáze A) dokážeme nalézt odpovídající entitu v dalším zdroji (databázi B) pomocí IRI. Tím se rozšíří dostupné informace o dané entitě.

RDF lze serializovat pomocí různých formátů, V této práci je využívána především turtle notace[11], kde se v základní podobě oddělují subjekt, predikát a objekt mezerami a triplet se ukončuje tečkou. Praktická ukázka s vysvětlením bude uvedena později v praktické části. jsou ale také podporovány i další formáty jako například RDF/XML, N-Triples, N-Quads a N3.

Ukázka turtle formátu na dřívějším příkladu s Petrem, který se přátelí s Karlem:

```
@base <http://example.org/>.
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<#Petr>
   rel:friend <#Karel> ;
   rdf:type foaf:Person ;
   foaf:name "Petr Novak" .

<#Karel>
   rdf:type foaf:Person ;
   foaf:name "Petr Zabijak".
```

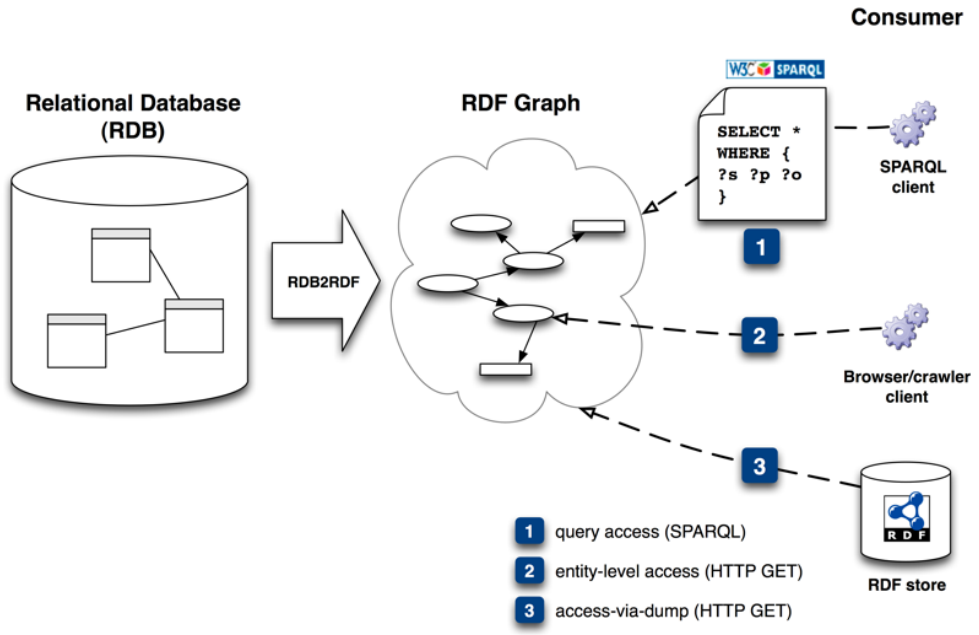
Zde jsou v ukázce na začátku použity prefixy pro zkrácení zápisu dlouhých IRI, používaných v tripletech.

Dále následují již jednotlivé triplety s popisem grafu. Středník na konci tripletu značí, že v dalším tripletu se použije stejný subjekt, a nemusí se znovu psát. K subjektu <#Petr> se tedy vztahují všechny tři dvojice objekt-predikát pod ním. Poslední triplet vztahující se k tomu samému subjektu je ukončen tečkou.

1.1.5 Mapování

Relační databáze mají tu výhodu, že jsou při některých operacích velmi efektivní a rychlé. V grafových databázích se zase lépe zjišťují neznámé vazby, a častěji jsou použity standardizované slovníky, což usnadňuje orientaci v datech získaných z jiných systémů.

Data lze ovšem mít také uložena v klasické, relační podobě, a přidat k nim navíc tzv. překladovou či **mapovací vrstvu**. Jak je vidět na obrázku 1.2), kde je mapování znázorněno šipkou RDB2RDF (relational database to RDF)
.



Obrázek 1.2: mapování v pohledu business usecase. Zdroj: `https://www.w3.org/TR/rdb2rdf-ucr/`

Díky této vrstvě se navenek „virtualizuje“ požadovaná část grafu na základě relačních dat. Pak dokáží specializované nástroje i nad relačními daty uloženými v tabulkách vykonávat SPARQL dotazy, a vracet výsledky v podobě, kterou by vracela grafová databáze.

Mapování můžeme tedy chápat jako soubor pravidel a vzorů, které definují vztah mezi relačními daty v databázi, a tím, jak mají vypadat navenek v grafové podobě. Tato vrstva je schopna tvořit (simulovat) na základě databázových schémat šablony grafů a vazby mezi nimi. Z údajů uložených v databázi poté tvořit podle zadefinovaných pravidel IRI a triplety. V mapování se nachází například informace, že IRI ve tvaru ***protein:id*** lze vytvořit za pomoci dat z **tabulky „protein_bases“** a **sloupečku „id“**. Pak pro **hodnotu 725** z *protein_bases.id* může vygenerovaná **IRI** vypadat například ***protein:725***. Dále lze v mapování používat další konstrukty jako jsou podmínky (obdoba WHERE podmínek v relačních databázích), ale i přímo SQL příkazy i join tabulek.

Uživatel nebo program, který pracuje s daty tak v ideálním případě ani nepozná, zda je v pozadí stále grafová databáze, čí relační databáze s online mapováním.

1.1.5.1 R2RML standard

Standardem pro tento druh práce s daty je jazyk R2RML (**RDB to RDF Mapping Language**) [10]. Samotné R2RML mapování je tvořeno RDF grafy zapsanými v Turtle notaci, a umožňuje různé implementace. Mapování dle tohoto standardu je implementováno například v nástrojích Openlink Virtuoso, DB2Triples, Capsenta Ultrawrap a dalších. Standard také specifikuje, jak správně překládat/mapovat data z jednotlivých tabulek v relační formě, a vztahy mezi tabulkami.

Mapování může být využito více způsoby:

- například pro **jednorázový export všech dat z relační databáze do grafové podoby/souboru**. Tento způsob je označován jako RDF dump relační databáze. Výsledný soubor lze poté importovat do grafové databáze, a dotazovat se na data přímo pomocí SPARQL jazyka, jelikož jde již o grafovou podobu. Ovšem dump databáze není interaktivní operace, a je tedy vhodný spíše pro neměnící se databáze nebo může být využit v některých případech pro zálohu/distribuci dat. Také pokud chceme databázi převést do RDF databáze, tak můžeme tuto možnost využít.

- Dále je možné na SPARQL dotazy díky mapování vrátit **z relačních dat zkonstruovaný graf** v RDF (či jiném podporovaném) formátu a ten například uložit na disk nebo s ním dále pracovat. Tento způsob již neexportuje kompletně celou databázi, ale pouze **podmnožinu dat**, na základě pravidel vytvoří nový **graf odpovídající výsledku SPARQL dotazu**. Tento způsob využití mapování může být vhodný například pro zálohy jen určité části databáze, nebo pokud nechceme mapování používat při každém SPARQL dotazu (třeba z výkonnostních důvodů), tak potřebnou část můžeme na začátku práce „převést z relační do grafové podoby“.

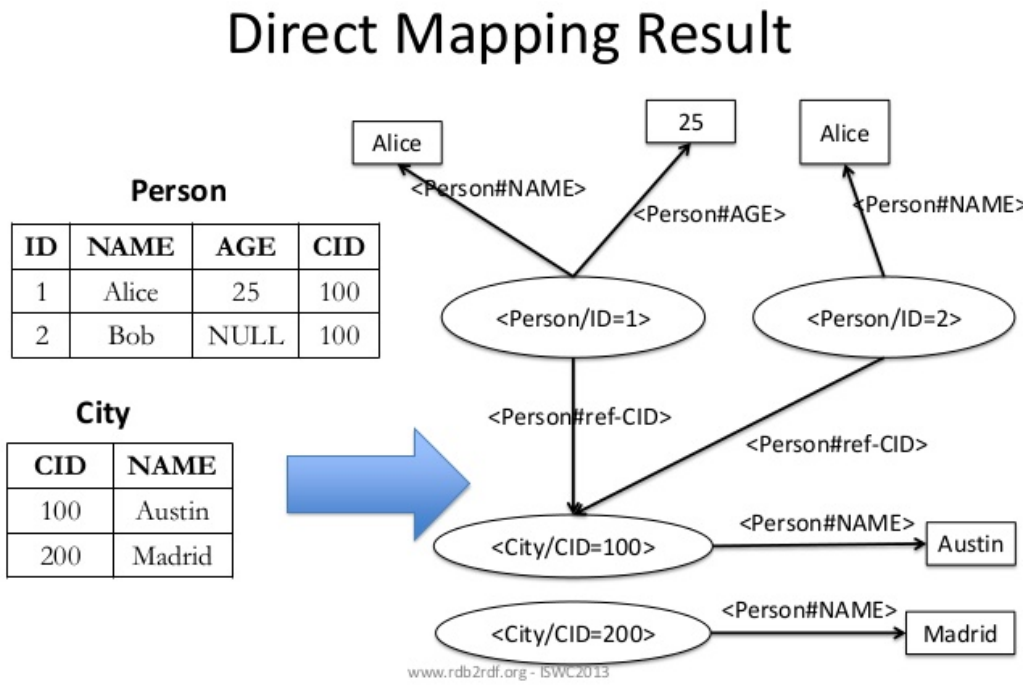
- Způsob použitý v rámci této práce spočívá ve **vyhodnocování SPARQL dotazů online nad relačními daty za pomoci mapovacího souboru**. Tento způsob vrátí přímo výsledky ve formě tabulky a je nejvhodnější pro práci s daty v reálném čase, bez potřeby vytvářet mezisoubory. V diplomové práci se budu věnovat především tomuto způsobu práce s mapováním, mimo jiné kvůli návaznosti na již existující systémy a webové služby pracující s online dotazy.

- **Direct mapping - automaticky vygenerované mapování**. Toto mapování je vhodné pokud chcete sdílet relační data v prostředí sémantického webu. Názvy uzlů a hran se automaticky odvodí od názvů tabulek a sloupců. Základní struktura pak vzniká tak, že se triplet „Subjekt - vazba - Objekt“ odvodí jako„Hodnota primárního klíče - název atributu - hodnota atributu“.

Na obrázku 1.3 z www.rdb2rdf.org je vidět výsledek přímého mapování v praxi na jednoduché ukázce. Jedna tabulka obsahuje informace o osobách, jejich ID, jméno, věk a identifikátor města, ve kterém žijí (cizí klíč). Druhá tabulka pak obsahuje základní informace o městech, atributy s identifikátorem jako primárním klíčem, a celé jméno města.

Z primárního klíče tabulky se vytvoří sémantický uzel, a atributy se k němu poté z každého sloupečku s hodnotou navážou pomocí dalších tripletů. navíc zde díky znalosti o cizím klíči dojde i k propojení uzlů reprezentujících osobu s uzly reprezentujícími město.

NULL hodnota ve věku Boba se projeví neexistencí tripletu/vazby v grafu boba s jeho věkem (na rozdíl od Alice, kterou reprezentuje <Person/ID=1>, a má vazbu na věk s hodnotou 25).



Obrázek 1.3: Ukázka mapování pomocí direct-mapping. Zdroj: `www.rdb2rdf.org`

1.2 Nástroje

Během práce jsem zkoušel různé postupy a nástroje implementující RDB2RDF standard[12]. Při rešerši se zpočátku zdálo, že je k dispozici mnoho vhodných nástrojů pro tuto problematiku, bohužel tomu tak ve výsledku nebylo. Velké množství nástrojů jsou pouze ve fázi raného vývoje alfa a beta verzí, implementují jen minimální funkčnost, či nepracují se standardními formáty-/databázemi. zde uvádím stručný výčet nejzajímavějších nástrojů s největším potenciálem s odůvodněním jejich vhodnosti či nevhodnosti.

- **OpenLink Virtuoso**[2] - Nástroj s podporou hybridní architektury databáze. Dokáže operovat jak se standardní relační databází na bázi tabulek a sloupců, tak i spravovat grafové uložiště (SPARQL RDF based Quad Store), což je v podstatě databáze uchovávající triplety (navíc s informací do kterého grafu patří) místo klasických sloupcových tabulek. Dále umožňuje práci s relačními daty SPARQL jazykem **za pomoci mapování**. Virtuoso také umožňuje práci se základními formáty jako jsou HTML, TEXT, TURTLE, RDF/XML, JSON a XML. Dále také poskytuje webový aplikační server umožňující práci pomocí SOAP nebo REST api. Tento systém je navíc schopen některých optimalizací během převodu dotazů ze SPARQL do SQL podoby, což často zvyšuje rychlost vykonání dotazu a snižuje zátěž databáze oproti naivnímu překladu. Tento nástroj se bohužel ukazuje v některých situacích nestabilní.

- **XSPARQL**[3] - nástroj pro práci s XML a RDF soubory jako uložištěm dat. XSPARQL dokáže pomocí XSLT transformací a dalších postupů provést konverzi mezi daty uloženými v XML a RDF formátem. Ač se tento nástroj hodí pro extrakci dat z XML souborů a jejich převod do grafové podoby, v našem případě nemáme zdrojová data v XML formátu, a potřebujeme nástroj pro práci nad relační databází, ideálně s možností online dotazů v jazyce SPARQL.

- **Ultrawrap**[4] se jeví jako velmi mocný nástroj v oblasti práce s grafovými databázemi. Podle údajů na webových stránkách podporuje Ultrawrap jednorázový dump (export) databází/souborů do RDF formátu, wrapper, který dokáže vykonávat SPARQL dotazy nad relační databází v reálném čase. Navíc proklamuje vylepšený výkon díky pokročilým optimalizacím, podporu SPARQL 1.1 (tedy agregační funkce, property paths atp, které jsou také potřeba).

[2] **OpenLink Virtuoso** - 'https://virtuoso.openlinksw.com/'
aktuálně nasazený systém pro práci s daty pomocí mapování

[3] **XSPARQL** - 'http://xsparql.deri.org/' Nástroj pro převod dat ve formátu XML na formát RDF

[4] **Ultrawrap** - 'https://capsenta.com' od firmy Capsenta, technologicky asi nejvhodnější alternativa za dosavadní nástroj Virtuoso

Bohužel, pro tento nástroj se nám nepodařilo získat licenci ani po přímém vyjednávání s firmou Capsenta. Z telefonátu vyplynulo, že mají příliš mnoho práce s velkými zákazníky a my pro ně nejsme dostatečně zajímaví (nezávisle na ceně). Tudíž zůstává nezodpovězenou otázkou, na kolik by byl tento systém vhodný a funkční v našem prostředí.

- **RDF-RDB2RDF**[5] - program v Perlu s podporou direct mapping technologie a klasického mapování. Tento program je bohužel zatím jen ve vývoji, s posledním commitem z roku 2013 a také nepodporuje některé důležité operace (podmínky, join...).

- **D2RQ**[6] - Nakonec nejvhodnější dostupný nástroj pro přístup ke klasické relační databázi SPARQL dotazy za pomoci vlastního mapování. Vlastní mapování je potřeba ze dvou důvodů, jedním je efektivita, a druhým je původní podoba dat z projektu PubChemRDF, kterým by mělo mapování co nejvíce odpovídat. Což by automaticky vygenerované mapování samozřejmě nesplňovalo.

  D2RQ nabízí možnosti jako dotazování se na negrafovou databázi pomocí SPARQL, přistupovat k obsahu databáze jako k linkovaným datům pomocí webového rozhraní, vytvořit vlastní dump celé databáze.

  Tento nástroj je navíc opensource, tudíž lze v případě potřeby upravovat jako Java projekt.

V rámci této práce jsem vyzkoušel i několik dalších nástrojů, ale žádný z nalezených nebyl vhodný svou připraveností či funkcionalitou pro nasazení do produkčního prostředí, či neměl akceptovatelné licenční podmínky.

1.3 Doména projektu PubChem a PubChemRDF

Systém **PubChem** poskytuje od roku 2004 informace o **biologických vlastnostech a účincích malých molekul**. Spravuje ho Národní centrum pro biotechnologické informace (National Center for Biotechnology Information, NCBI), které je součástí Národní lékařské knihovny.

V databázi PubChem lze zdarma zadávat **dotazy přes webové rozhraní** na adrese `https://pubchem.ncbi.nlm.nih.gov/search/`. Pomocí tohoto nástroje je možné vyhledávat například chemické sloučeniny podle jejich názvu, označení, struktury či chemického složení. Dále lze vyhledávat v bioassays (vědeckých experimentech prováděných na chemických látkách) například protokoly, popisy, vstupní a výstupní data. Ale také je zde část pro hledání v patentech souvisejících s chemií, speciální sekce pro geny, proteiny, a další[15].

[5] **RDF-RDB2RDF** - 'https://metacpan.org/release/RDF-RDB2RDF' Nástroj s podporou mapování i direct mapping, bohužel zůstal pouze v rozpracované verzi

[6] **D2RQ** - 'http://d2rq.org/'Nástroj s podporou mapování pro relační databáze do RDF podoby

Celý systém PubChem je organizován do tří, navzájem prolinkovaných databází v rámci NCBI: PubChem Substance, PubChem Compound a PubChem BioAssay. Což jsou relační databáze nesoucí informace o chemických látkách, sloučeninách a bioassays[24]. Navíc jsou tyto databáze propojeny s dalšími systémy z NCBI, jako jsou například PubMed[17] (databáze vědecké literatury a citací) a databáze interaktivních 3D modelů proteinů[18]. Linky z PubChem do dalších databází poskytují informace o biologických vlastnostech.

1.3.1 Užitečnost dat a souvislosti

Celý projekt plní funkci velké knihovny nebo databáze chemických látek (substances), a jejich biochemických vlastností. Data se dají rozdělit do tří hlavních částí, první částí jsou samotné chemické látky, které lze získat od různých distributorů, plus různé další informace k těmto látkám.

Další velkou částí databáze jsou chemické sloučeniny. Sloučenina v tomto kontextu popisuje normovanou chemickou strukturu. Různí distributoři dodávají někdy tytéž chemické látky pod různými názvy a s různým popisem. Aby bylo možné snáze určit, že konkrétní látka od jednoho výrobce je shodná s látkou poskytovanou druhým výrobcem, každá látka je popsána právě jednou sloučeninou (normovaná podoba látky). Když má více látek stejnou normovanou podobu (jsou popsány tou samou sloučeninou), dají se považovat za totožné.

Díky tomu lze poté například propojit informace z různých pokusů na různých látkách, protože to je ve skutečnosti tatáž chemická sloučenina, jen pod jinými obchodními názvy.

Třetí velkou částí databáze jsou pak informace o samotných biologických experimentech (bioassays). Experimenty se skládají z několika measusregroup. Výsledky (reaguje, a při jaké koncentraci, nebo nereaguje) těchto experimentů (endpoints) jsou poté zaznamenány do databáze.

Díky této struktuře můžeme zjistit, která chemická sloučenina má jaké vlastnosti (nezávisle na distributorovi a obchodních značkách), s čím a jak dobře reaguje, ale i další informace, jako kde se o dané chemické sloučenině psalo, nebo kdo jí má patentovanou.

Také lze v databázi podobností zjišťovat, zda existují nějaké další podobné chemické sloučeniny, a jak moc se shodují s námi zkoumanou sloučeninou. U podobných sloučenin lze předpokládat, že by mohly mít i podobné biochemické vlastnosti a reagovat podobně. Mohou být ale pod jiným patentem, mít více dostupných informací v literatuře či nemít nějaký nežádoucí účinek.

Schéma vztahů v systému PubChem lze dobře vidět například na obrázku 1.5. Biologická měření bioassay vlevo složená z testů jednotlivých skupin proteinů (measure groups) proti konkrétním látkám (substances). Výsledky jednotlivých měření proteinu proti chemické látce se ukládají ve formě tzv. endpointů.

Aby výsledky experimentů byly použitelné nezávisle na distributorovi látek, jsou látky připojeny na svou standardizovanou podobu (compounds). Většina věcí v systému (od measuregroup po jednotlivé části proteinů nebo různé látky) má také vazbu na reference (publikace, kde se o konkrétních věcech pojednává).

1.3.2 O projektu PubChemRDF

Projekt **PubChemRDF**[16] vznikl za účelem poskytnout uživatelům **ke stažení obsah databází PubChem**[15][16]. Webové rozhraní PubChemu sice umožňuje poměrně komplexní vyhledávání pomocí různých dotazů a vrací požadované výsledky, ovšem pouze je zobrazí ve webovém prohlížeči. Tedy v HTML jazyce (zobrazí výsledky v různých oknech na stránce), jazyce primárně určenému k popisu vizuální podoby stránky a nikoli datové struktury. To se hodí pro lepší orientaci člověka na stránce a ve výsledcích, nikoliv však pro strojové/automatické zpracování.

Díky projektu PubChemRDF je možné z FTP serveru **stáhnout** požadovaná **data ve formátu RDF** (ftp://ftp.ncbi.nlm.nih.gov/pubchem/RDF/). Tato data lze poté například nahrát do RDF databáze (databáze specializované k ukládání RDF tripletů ve vlastní interní interpretaci a vyhodnocování SPARQL dotazů nad uloženými daty). Data jsou na FTP serveru uspořádána tak, aby nebylo nutné stahovat je kompletní, ale aby stačilo stáhnout jen tu část, se kterou potřebujeme zrovna pracovat, například informace o proteinech, bioassays atp. Pokud nás tedy zajímají například pouze popisy chemických vlastností sloučenin, a jejich případné patenty, můžeme si stáhnout jen tyto podčásti.

1.3.3 Zdrojová data

Samotný projekt PubChem interně ukládá data v relační podobě[24]. Veřejně ale poskytují zdarma data ve formátu RDF, prostřednictvím PubChemRDF projektu[16].

Tato data je pak možné stahovat z FTP serveru, ale kompletní databáze má velikost v řádu stovek GB. Zde je stručný popis dostupných datových sad, které se dají samostatně stahovat a provozovat. Nebo se dají provozovat jako jeden celek, pouze načtením několika/všech RDF souborů[16].

- **bioassay** - PubChem BioAssay RDF triplety obsahují informace o typu, titulku, zdroji dat a link na measure groups pro daný esej. Jedná se v podstatě o popis konkrétních měření.
- **biosystem** - Biologické systémy, mají informace o typu, titulku, organismu ve kterém se nachází, zdroji odkud informace pochází, a případně reference na odbornou literaturu, kde je zmiňován. Může obsahovat cross-linky do databáze Reactome RDF.

- **compound** - RDF triplety, popisující propojení sloučenin a jejich deskriptorů, vzájemně provázaných sloučenin (pomocí compound identity groups), sloučenin a inchikeys. Kvůli velkému množství sousedů značících podobnost (2D i 3D), je seznam podobností ve zvláštním souboru. Dále jsou zde i informace o jednotlivých komponentách.

- **concept** - Subdoména „concept“ popisuje biomedicínské koncepty používané pro popis jednotlivých zdrojů v PubChemRDF. Například WHO ATC kódy jsou využívány pro označení jednotlivých synonym. Tudíž jsou zde uloženy typy konceptů, schéma, kterým se řídí, zdroj, rodičovský koncept a popisek konceptu.

- **conserveddomain** - PubChem conserved domain RDF triplety zachycují informace o jednotlivých, evolučně zakonzervovaných (nachází se ve stejné podobě ve velkém množství organismů, nedotčeny evolucí), částech proteinů, které jsou víceméně autonomní, a mohly by fungovat i samy o sobě. Jsou zde informace o typu, titulku a případných referencích na danou zakonzervovanou doménu proteinu.

- **descriptor** - PubChem descriptor RDF poskytuje triplety s typem, hodnotou a jednotkou daného descriptoru. Například že se jedná o hmotnost, s hodnotou 223 a jednotkou jsou mikrogramy.

- **endpoint** - PubChem endpoint RDF triplety zachycující výsledky experimentů jednotlivých proteinů s konkrétní látkou. Triplety odkazují na typ, hodnotu a jednotku měření, poté konkrétní látku (substanci), se kterou protein reaguje, a odkaz na referenci (například článek), kde se o výsledku píše.

- **gene** - PubChem gene RDF nese informace o jednotlivých genech. Jsou zde záznamy o typu, titulku, slovním popisu, organismů, které daný gen mají, symbol genu a případné reference na literaturu.

- **inchikey** - International Chemical Identifier je zahashovaný, 27 znaků dlouhý identifikátor standardizované chemické sloučeniny. InChIKey má hlavní význam při vyhledávání sloučenin na webu a je používán jako klíč pro hledání.

- **measuregroup** - experimentální měření (o kterých je pak veden záznam v bioassays), a k nim patřičné endpointy (výsledky měření). Tato část obsahuje informace o typu a titulku měřené skupiny, proteinech, které obsahuje a výsledcích testů jednotlivých proteinů.

- **protein** - PubChem protein RDF triplety obsahují informace o typu proteinu (z Protein Ontology systému), titulek, podobné proteiny, conserved domains (uvsoběstačné součásti ze kterých je celý protein složen),

zakódované geny, odkazy na organismy, biosystémy, reference na literaturu, a cross-link do databáze UniProt RDF a NCBI Protein. Pokud má protein krystalickou 3D strukturu v PDB databázi, navíc obsahuje i link do RDF-based PDB resource.

- **reference** PubChem reference RDF jsou záznamy o jednotlivých odborných publikacích, které se zmiňují od chemických látkách, sloučeninách a dalších souvisejících věcí. Obsahuje informace o typu publikace (článek...), citaci, titulek, nadpisy z databáze MeSH a PubMed, abstrakt reference se zmínkou látky, nemoc které se článek týká a datum publikace.

- **neighbor** - PubChem neighbor RDF popisuje podobnostní vztahy mezi jednotlivými chemickými záznamy, proteiny atp. a další související informace. Tyto triplety popisují typ podobnosti (2-D, 3-D...), sloučeninu se kterou je daný záznam podobný, a hodnotu ohodnocení podobnosti plus typ hodnocení této podobnosti.

- **source** - PubChem data source RDF obsahuje informace o zdroji záznamu. Jednak o typu zdroje, titulku, přispěvateli a informacích o něm, a také o kategorizaci látek.

- **substance** - RDF triplety, popisující propojení látek s jejich chemickým deskriptorem, látek s jejich standardizovaným vzorem sloučeniny (compound), a látek s jejich zdroji dat. Pokud byly údaje o látce poskytnuty ChEMBL nebo NCBI MMDB databází, je poskytnut cross-link do těchto databází.

- **synonym** - PubChem Synonym slouží k identifikování shodných látek od různých distributorů. Může být prolinkováno na MeSH concept a WHO INN databáze.

1.3.3.1 Množství dat

Velikost jednotlivých datasetů ve zdrojové podobě je následující:

Největší datasety jsou tedy sloučeniny, deskriptory, látky a inchikey identifikátory. **Celkem mají RDF data PubChemu 401 GB**, pro efektivní práci s databází je potřeba, aby neběžela z pevného disku, ale ideálně byla **načtená celá v paměti**.

V rámci tohoto textu se nepracuje s databází podobností sloučenin (compound/nbr3d a compound/nbr2d).

bioassay	6,3M	endpoint	2,5G
biosystem	24M	gene	15M
compound/general	7,8G	inchikey	3,3G
compound/nbr2d	216G	measuregroup	899M
compound/nbr3d	129G	protein	13M
concept	152K	reference	1,9G
conserveddomain	616K	source	24K
descriptor/compound	20G	substance	9,1G
descriptor/substance	1,3G	synonym	12G
total	401G		

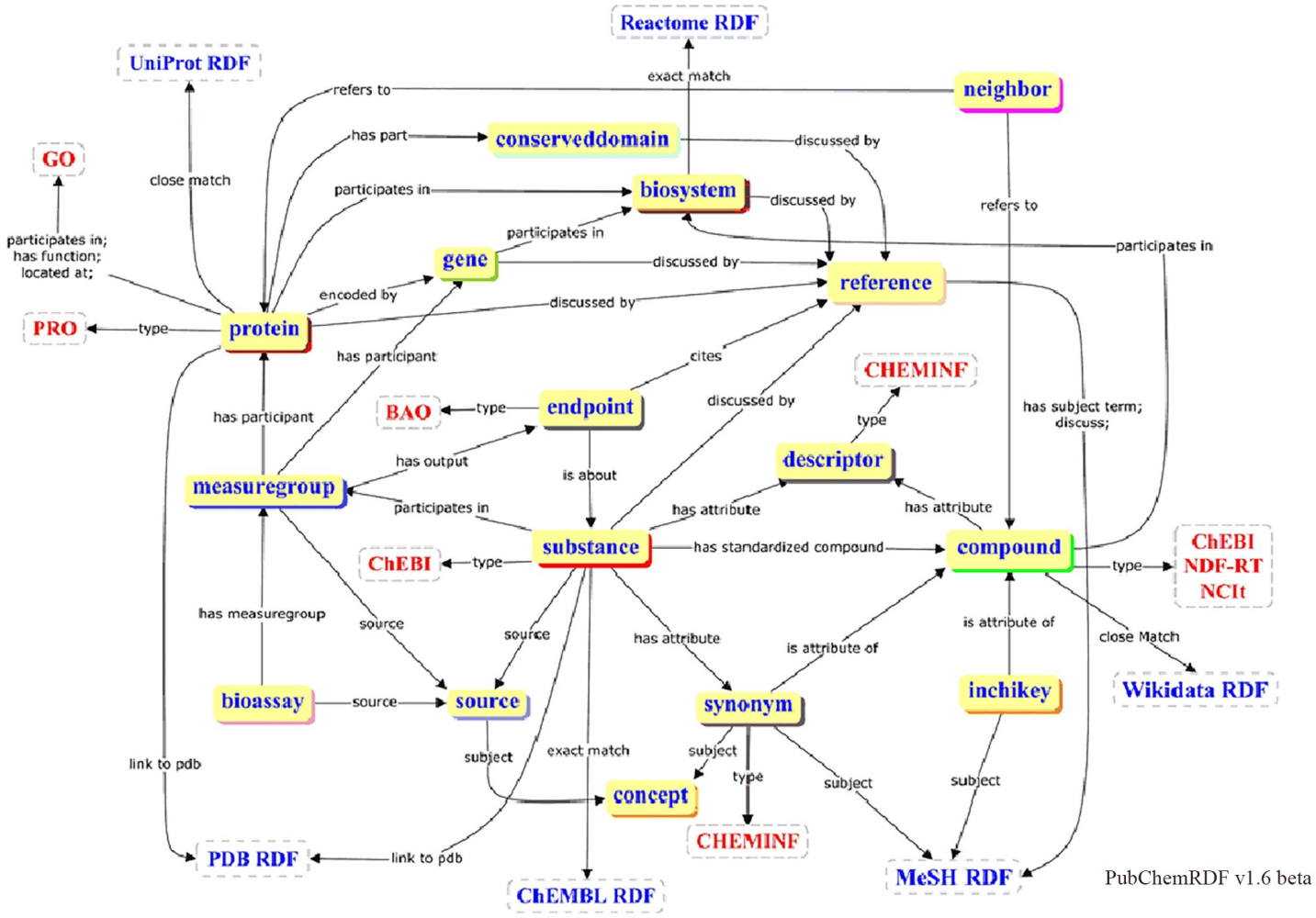
Obrázek 1.4: Přehled jednotlivých kategorií dat podle velikosti

1.3.3.2 Sémantická podoba dat

Celková architektura a struktura dat

Jak je znázorněno na obrázku 1.5, Sémantická struktura PubChemRDF se skládá ze souboru základních entit (zde v podobě obdélníků), jako jsou substance, measuregroup, descriptor, biosystem, protein... Tyto entity jsou pak pomocí predikátů (zde šipek, vztahů) propojeny navzájem mezi sebou, ale mohou se odkazovat i do jiných databází a systémů (zde čerchované obdélníky), jako například do MeSH RDF, ChEMBL RDF, ChEBI, BAO.

Struktura dat navíc umožňuje mít odkazy do nějakého subsystému, aniž bychom nutně tento systém potřebovali mít připojený.



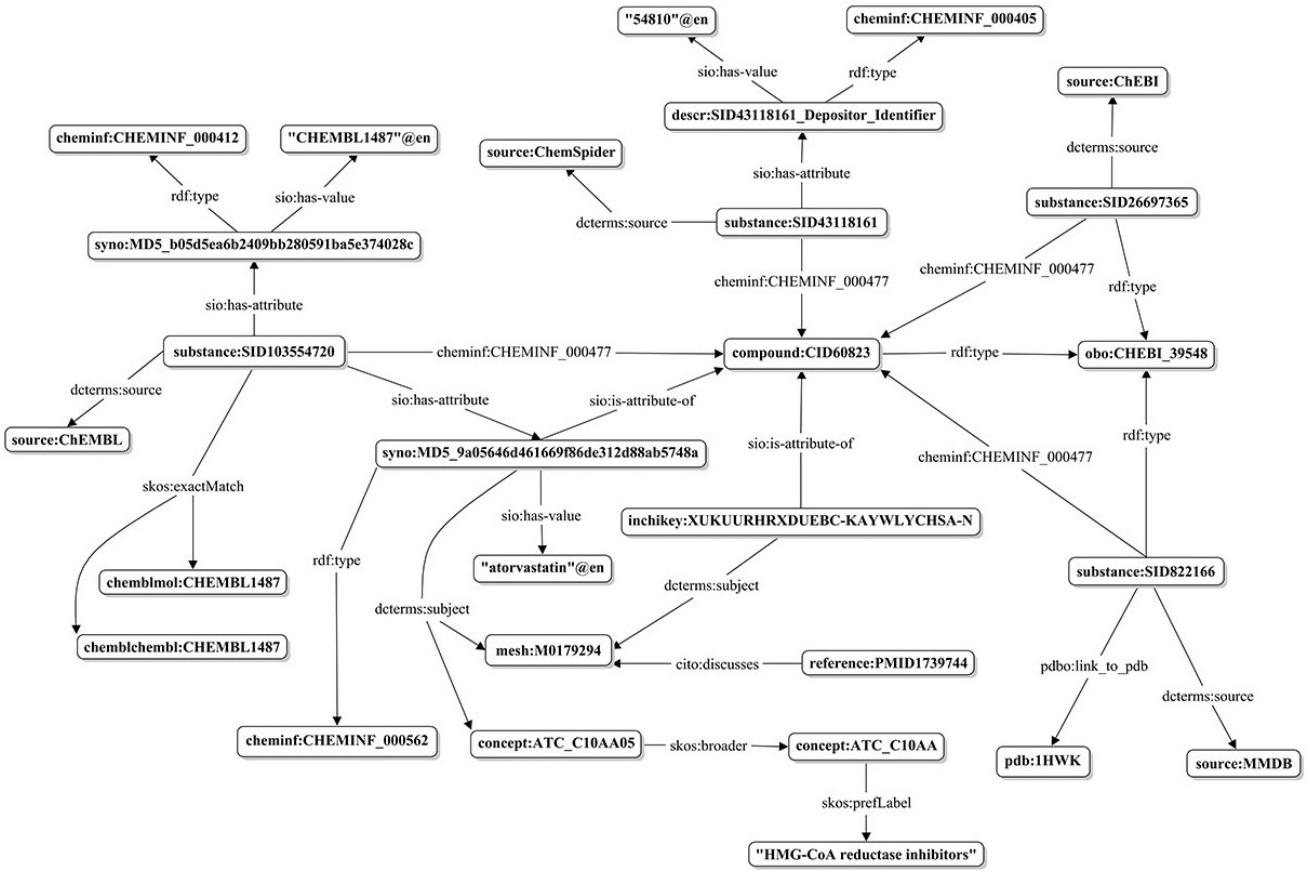
Obrázek 1.5: Obecný přehled sémantických vztahů v PubChemRDF. Zdroj: `https://pubchem.ncbi.nlm.nih.gov/rdf`

Ukázka na konkrétních reálných datech

Na dalším obrázku 1.6 je pak vidět ukázka struktury prolinkovaných dat na konkrétních záznamech z PubChem/PubChemRDF. Například chemická látka (substance) vlevo s *ID 103554720* (zapsáno pomocí zkrácené IRI jako *substance:SID103554720*) je zde propojena s entitou standardizované sloučeniny „ID 060823“ (zapsáno pomocí IRI jako *compound:CID060823*).

Také je v pravé části vidět, že k jedné standardizované chemické sloučenině je navázáno více různých chemických látek (ještě substance substance:SID26697365, substance:SID822166 a substance:SID43118161 ze shora). U těchto látek je vždy záznam o zdroji této látky přes vazbu *dcterms:source*, například *source:ChEMBL*.

U látek v pravé části je pak definován stejný typ *obo:CHEBI_39548* pomocí vazby *rdf:type*. Z obrázku 1.5 je pak patrné, že tato vazba na typ vede do jiné databáze, a to sice „ChEBI NDF-RT NCIt“. Pokud data z ChEBI nemáme k dispozici, zjistíme pouze tuto informaci, ale jestliže máme k dispozici ChEBI databázi, můžeme se dozvědět další informace o typu.

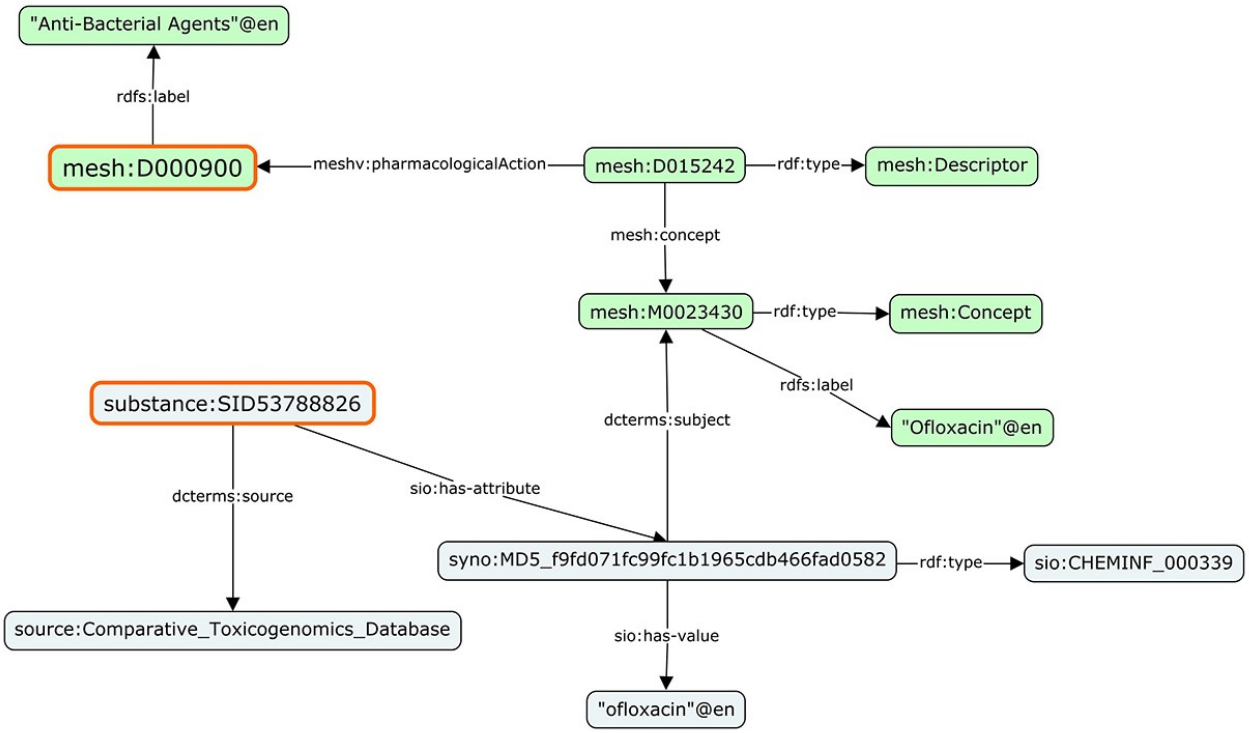


Obrázek 1.6: Struktura PubChemRDF s konkrétní substancí „substance::SID103554720“ a compound „compound:CID060823“, na kterou se odkazuje. Zdroj: `https://pubchem.ncbi.nlm.nih.gov/rdf`

Linkování dalších systémů pomocí IRI

Na dalším obrázku 1.7 je pak znázorněno, jak se pracuje s daty u propojení systému PubChem s dalšími systémy. V tomto příkladě konkrétně propojení s databází MeSH RDF přes IRI *mesh:M0023430* (uprostřed). Pokud bychom databázi MeSH RDF neměli, tento uzel by byl list grafu, a nešlo by z něj dále pokračovat. V případě, že připojíme další systém, automaticky se nám rozšíří možnosti procházení grafu a práce i s navázanými daty z MeSH RDF

databáze.



Obrázek 1.7: Integrace PubChemRDF a MeSH RDF, od substance pomocí prolinkování až k farmakologickému využití. Zdroj: `https://pubchem.ncbi.nlm.nih.gov/rdf`

1.3.4 Formát zdrojových dat

Zdrojová data poskytovaná v rámci projektu PubChemRDF jsou v RDF formátu, zapsaném turtle notací. Dále v příkladu souboru s látkami (rdf/substance) je vidět nejzákladnější forma zápisu RDF v této notaci.

```
@prefix sio: <http://semanticscience.org/resource/> .
@prefix substance: <http://rdf.ncbi.nlm.nih.gov/pubchem/substance/> .
@prefix compound: <http://rdf.ncbi.nlm.nih.gov/pubchem/compound/> .
substance:SID1049255 sio:CHEMINF_000477 compound:CID814449 .
substance:SID1049259 sio:CHEMINF_000477 compound:CID617052 .
substance:SID1049279 sio:CHEMINF_000477 compound:CID814471 .
substance:SID1049284 sio:CHEMINF_000477 compound:CID814476 .
.
.
.
```

Na prvních třech řádcích uvozených „*@prefix*“ jsou takzvané **prefixy**. Prefixy slouží ke **zkrácení zápisu IRI** v později popisovaných tripletech. Z prvního řádku vyplývá, že v dokumentu stačí psát „sio:“ místo celého „<http://semanticscience.org/resource/>“.

Na dalších řádcích jsou pak jednotlivé triplety, neboli trojice subjekt-predikát-objekt. **Triplet je rozdělen mezerami na tři části a ukončen tečkou**.

„substance:SID1049255“ jako **subjekt** představuje látku s označením SID1049255, což je konkrétní chemická látka prodávána nějakým výrobcem pod nějakým konkrétním obchodním názvem.

„sio:CHEMINF_000477“ jako prostřední člen (predikát) popisuje to, jaký vztah má první člen (látka) ke třetímu členu (sloučenině). Konkrétně například „CHEMINF_000477“ představuje vztah „has PubChem normalized counterpart“, tedy normalizovaný tvar dané chemické látky.

Třetí člen, „compound:CID814449“ popisuje chemickou sloučeninu s označením „CID814449“, která představuje normalizovaný a popsaný tvar, nezávislý na konkrétních výrobcích a distributorech.

Dohromady má tedy triplet význam spojení látky a jejího normalizovaného tvaru. Takto jsou postupně popsány všechny látky, o kterých je v databázi záznam.

V dalších souborech jsou potom tímto způsobem například propojeny látky s jejich zdrojem, nebo záznamy experimentů (bioassay) s konkrétními measuregroupami, se kterými byl experiment prováděn, a podobně.

Kapitola 2

Realizace

Cílem této diplomové práce je najít efektivní způsob práce s daty týkajících se chemických látek a malých molekul z projektu PubChemRDF[16], za pomoci dříve popsaných metod.

Prvním krokem je tedy získání zdrojových RDF dat a jejich nahrání do Virtuosa. Doporučení jak na to je popsán v projektu PubChemRDF. Těžší část spočívá v převodu RDF dat do relační podoby, a vytvoření mapování.

Schéma takto navržené relační databáze by mělo být zvoleno s ohledem na provádění zpětného mapování, to znamená zachovat co nejvíce informací a vazeb z originálních RDF souborů. Zároveň také rozdělit efektivně data do tabulek.

Mapování zpět do RDF formy zajistí možnost dotazovat se na PubChem data pomocí SPARQL dotazů. Toto mapování by mělo co nejvíce odpovídat původní RDF formě získané z projektu PubChemRDF, aby s ní byly kompatibilní dosavadní systémy.

Základní podoba uložení relačních dat do virtuosa a vytvoření mapování již byly zhotoveny v rámci ÚOCHB mimo tuto práci. Tyto věci jsem využil, a zopakoval celý postup pro lokální testování. Nakonec jsem pro toto řešení navrhl některé optimalizace popsané v kapitole 4.

Poté se data nahrají v relační podobě do zvláštní databáze (postgreSQL), nad kterou je potřeba také vytvořit nové mapování pomocí nástroje D2RQ. Data i mapování by měly být pro porovnání výkonu co nejpodobnější.

Nakonec je potřeba všechny tyto přístupy porovnat výkonově a funkčně.

Vlastnosti relační databáze Virtuosa i postgreSQL jsou rozdílné, od syntaxe až k optimalizacím jednotlivých SQL dotazů a jejich plánování. Také engine D2RQ funguje jinak než Virtuoso při interpretaci mapovacích pravidel a převodu původní SPARQL dotazu na SQL dotaz.

2.1 Virtuoso

Openlink Virtuoso je softwarový nástroj pro správu a integraci relačních databází. Kromě standardního R2RML mapování obsahuje také proprietární systém pro práci s relačními daty SPARQL jazykem. Vlastní **ekvivalent R2RML mapování** nazývaný **Linked Data Views**, který pro mapování využívá takzvaný Meta Schema Mapping Language. Navíc disponuje podporou hybridní architektury databáze. Dokáže tedy operovat nejen se standardní relační databází na bázi tabulek a sloupců, ale i spravovat grafové uložiště (SPARQL RDF based) místo klasických sloupcových tabulek. Virtuoso také umožňuje práci se základními formáty jako jsou HTML, TEXT, TURTLE, RDF/XML, JSON a XML. Také poskytuje webový aplikační server umožňující práci pomocí SOAP nebo REST API. Tento systém je navíc schopen některých optimalizací během převodu dotazů ze SPARQL do SQL podoby, což často zvyšuje rychlost vykonání dotazu a snižuje zátěž databáze oproti neoptimalizovanému, přímočarému překladu.

2.2 Získání zdrojových dat

Kompletní databáze chemických sloučenin Pubchem[15] je zdarma veřejně dostupná díky projektu PubChemRDF[16]. Data jsou zde na FTP serveru rozdělena do jednotlivých kategorií, v závislosti na tom, který subjekt popisují. Největší složky jsou pak compound, descriptor a substance. Úplně celá databáze po stažení v RDF formátu zabírá přes 400 Gigabytů, proto jsem lokálně pracoval pouze s částí dat, podmnožinou o velikosti cca 20GB. Na zprovoznění kompletní databáze jsem posléze potřeboval využít „produkční“ server s dostatečným výkonem a pamětí (bez načtení informací o podobnosti cca 60 GB RAM).

Zdrojová data v PubChemRDF jsou na tento druh práce uzpůsobena, a jsou rozdělena do souborů o velikosti 20 - 100 MB v jednotlivých složkách. Pro experimenty s různými nástroji a jejich nastavením jsem tedy využíval malou část dat. Malé domény jsem nahrál kompletní, a z velkých domén, jako například compound (řádově desítky až stovky GB), jsem vybral asi gigabytové podmnožiny.

Prakticky chybějící data při pokusech na lokálním stroji nejsou zase až takový problém, ale pouze nedostaneme v některých případech všechny výsledky. Což na odlaďování syntaxe a nástrojů nevadí. Pokud je SPARQL dotaz směřován na zdroje (source) týkající se nahraných látek v databázi, tak zdroje mají necelý 1MB a tudíž máme všechny. Ke každé naší látce bude tedy nalezen její zdroj, a další informace o zdrojích budou vráceny správně (každá naše látka má v databázi i svůj zdroj). Pokud ovšem budeme prohledávat graf směrem například od inchikey (hashovanému identifikátoru) ke sloučeninám, nedostaneme všechny očekávané výsledky. V databázi bude totiž velké

množství inchikey identifikátorů, které budou popisovat chemické sloučeniny (a mít na ně IRI odkaz s ID sloučeniny), ale tyto sloučeniny nebudou fyzicky v databázi nahrané. U identifikátoru například naštěstí víme, že každý identifikátor má svou sloučeninu, ale například pokud budeme hledat sousedy sloučenin (podobné sloučeniny), tak dost dobře nepoznáme, zda sloučenina souseda nemá, či jen nemáme záznam o podobnosti ještě načtený.

2.3 Načtení dat do Virtuosa v relační podobě

Načítání a konverze nativních RDF dat do relační databáze ve Virtuosu je realizováno pomocí externího Java programu vyvinutého na ÚOCHB. Pro načtení některých souborů se používá volání modulů z **Jeny**[21]. **Apache Jena** je free a open source nástroj pro tvorbu sémantického webu a práci s *linked data*.

Nejprve je potřeba připravit relační databázi, do které chceme nahrávat data. Zejména přístupnou adresu a port databáze, přístupové jméno a heslo s nastavenými patřičnými oprávněními pro vkládání a čtení záznamů. Dále je vhodné nastavit například také timeouty spojení a maximální počet připojení do databáze, což umožní paralelní zpracování.

Ve vlastních Java třídách je poté Jena využívána pro načtení některých komplexních vstupních datových RDF souborů. Poté je třeba rozparsovat zadaný RDF soubor, a podle námi určených pravidel data zpracovat podle toho, čeho se konkrétní záznamy týkají. Toho je využito především u souborů zachycujících složitější datové struktury. U jednodušších vstupních souborů probíhá parsování i ukládání dat do databáze kvůli efektivitě bez pomoci parsovacích modulů Jeny.

Pro každou doménu je vytvořena zvláštní Java třída, s předdefinovanými vzorci, které se v dané doméně vyskytují. Poté se během parsování vybere vhodný pattern, kterému aktuální záznam odpovídá, a vykoná se potřebná operace.

Například pokud je v tripletu u compound predikát se zadanou hodnotou „http://purl.obolibrary.org/obo/has-role“, třída určená pro práci se sloučeninami pozná, že se jedná o definici role. Na základě toho je rozhodnuto, že se triplet zpracuje určeným způsobem, a data jsou uložena do tabulky *compound_roles*. Subjekt (označující chemickou sloučeninu pomocí jejího ID) je rozparsován, a ID sloučeniny je ve sloupečku „compound“ použito podobně jako cizí klíč do základní tabulky sloučenin „compound_bases“ (ID ve sloupečku compound odpovídá sloučenině s tímto ID v základní tabulce). To která pravidla budou pro rozbor souboru využita se rozhoduje na základě názvu vstupního souboru.

Objekt je poté z tripletu také rozparsován a použit pro označení typu role ve sloupečku „roleid“. Naprostá většina sloupečků poté neobsahuje IRI, ale pouze variabilní část, lišící se v u každého prvku v dané skupině. Například

z Compounds (ve tvaru compound:CID814449) se ukládá pouze číselná hodnota (814449), podle které lze jedinečně identifikovat sloučeninu. Při zpětné tvorbě IRI je na to ale třeba pamatovat, a „compound:CID“ opět na začátek přidat.

2.3.1 Struktura relačních dat

Data ze stažených RDF souborů jsou načítána do předem připravených databázových struktur. Projekt PubChem má pevně definovaná pravidla, jako například která doména může být s kterou propojena a jakým způsobem. Víceméně především jen díky tomu lze připravit databázové tabulky odpovídající vztahům v linkovaných datech původního projektu. Sémantická struktura PubChem databáze je tedy „rozebrána“ na jednotlivé vazby, které jsou přeneseny do relační podoby. Díky tomu, že je možnost vazeb mezi jednotlivými doménami předem specifikována, je možné v programu zajišťující load dat rozhodnout, jak každý z těchto sémantických vztahů interpretovat v relační podobě.

Jako základ pro každou doménu je vytvořena tabulka „$Domain_bases“, kde „$Domain“ část v názvu tabulky značí nějakou konkrétní doménu z PubChemRDF dat. Například proteiny mají základní tabulku „protein_bases“, sloučeniny „compound_bases“ a látky „substance_bases“. V této základní tabulce jsou vždy obsaženy všechny entity dané domény.

Základní (*_bases) tabulky mohou v některých případech obsahovat pouze jeden sloupec s identifikátory všech entit dané domény. Například „compound_bases“ obsahuje pouze sloupec „id“, který obsahuje všechny zaznamenané chemické sloučeniny (resp. jejich identifikátor), které v databázi existují.

Tyto základní tabulky mohou ale mít i další sloupce, například tabulka všech proteinů „protein_bases“ se skládá ze čtyř sloupců. [„id“ | „name“ | „organism“ | „title“]. V základní tabulce proteinů je tedy navíc záznam o názvu proteinu, organismu ve kterém se vyskytuje, a jeho popis.

Záznam takového proteinu (část ze základní tabulky) pak vypadá například následovně: id = „1“, name = „GI2624488“, organism = „9913“ (což je třeba v tomto případě dobytek), title = „Chain B, Gamma-Chymotrypsin L-Naphthyl-1-Acetamido Boronic Acid Acid Inhibitor Complex“

Do základních tabulek se v podstatě mohou kumulovat informace, které představují vazbu *1:1* nebo *N:1*. Kde každý element v doméně má přiřazenou maximálně jednu takovou vazbu. Vhodné je tedy do základní tabulky zaznamenat například název (právě jeden, vazba 1:1) či zdroj dané věci/záznamu (právě jeden, ale vazba N:1, více různých záznamů se může odkazovat na stejný zdroj, který poskytl data) nebo jejich typ/kategorii.

Tímto způsobem se ovšem nedají řešit vztahy, které mají kardinalitu *M:N*, což je například vztah konkrétní chemické látky a reference (vědecké práce, panelu, článku). O jedné chemické látce se mohlo psát ve více různých vědeckých publikacích, a zároveň v publikaci může být zmíněno více chemických

látek. Pro tyto případy je potřeba v relační databázi vytvořit speciální vazební tabulky. V těchto tabulkách je pak obvykle pouze informace o vztahu dvou entit ze dvou různých domén, tedy pouze dva databázové sloupečky. Například *substance* | *reference*. V prvním sloupečku jsou chemické látky, a v pravém k nim příslušné reference.

2.4 Mapování relačních dat do RDF podoby ve Virtuosu

Mapování je v podstatě přesně opačný proces, než který byl použit při načítání dat z původního RDF formátu. Při načítání dat bylo potřeba uložit informace o sémantické struktuře do jednotlivých tabulek, mapování nám na druhou stranu umožňuje z relačních tabulek vytvářet zase zpět sémantické konstrukce a vazby na základě námi definovaných pravidel.

Cílem mapování je vytvořit rozhraní, které bude v reálném čase vyhodnocovat SPARQL dotazy, ale nad relačními daty místo nad grafovou databází. Navíc je potřeba, aby se námi vytvořené rozhraní co nejvíce podobalo původní struktuře sémantického pohledu. Jednak, aby byl nový systém kompatibilní s již používanými systémy, a ty bylo možno přepojit, ale také aby se správně provázala potřebná data na lokální úrovni. Například pokud se IRI generuje dynamicky na různých místech z různých tabulek, je potřeba, aby se výsledná podoba IRI shodovala, pokud na počátku v originálních PubChem RDF datech označovala totožnou entitu. Pro větší efektivity jsou místo IRI použity jednoduché SQL typy. Mezi těmito typy a IRI je pak definována bijekce. To znamená, že IRI se dá jednoznačně vytvořit pouze jedním způsobem ze zadaných parametrů, a zároveň jsme zpětně z IRI schopni zjistit hodnotu těchto parametrů.

Pokud bijekci správně nedefinujeme, při mapování je potřeba opět složit celý textový řetězec a identifikátor z databáze použít například formou proměnné a vložit jí do generované IRI. Bez bijekce se tedy zbytečně vytváří řetězce a porovnávají se stringově, místo aby se hledalo v celočíselných hodnotách (s využitím indexu), a ty se pak porovnávaly rovnou jako číslo s číslem. Díky bijekci engine pozná, že konkrétní IRI se generuje jen tímto způsobem na základě toho a toho atributu, a může pak porovnávat/joinovat přímo hodnoty atributů.

Virtuoso umožňuje pro **mapování** používat **SQL skripty**, je tedy možné rozdělit mapování do více souborů podle účelu a domény. Mapování téměř každé domény je zde rozděleno do **několika základních částí**. V principu se ale jedná o **samotné mapování** jako seznam pravidel, které popisuje pro které sémantické vztahy se berou data ze kterých tabulek a sloupců, a jak je vzájemně propojit. V dalších částech jsou pak případně popsány složitější vztahy a funkce pro danou doménu.

2.4.1 Mapovací soubory

Vhodná ukázka takového základního mapovacího souboru je například **biosystem.sql**2.1, na kterém jsou dobře patrné nejčastější konstrukce používané pro definici mapování. Prvních 9 řádků je úvodní hlavička mapování (použitá syntaxe podobná SPARQL a nastavení pojmenování, zdrojových tabulek). Na řádku 10 již poté začíná vlastní **mapovací funkce** definovaná pomocí tripletů. Na řádku 10 je popis **subjektu**, na řádku 11 pak popis **predikátu** (rdf:type) a **objektu** (bp:Pathway). Jelikož není řádek 11 ukončen **tečkou**, ale **středníkem**, znamená to, že pokračování na řádku 12 se stále vztahuje ke **stejnému subjektu** (iri:biosystem na řádku 10). Toto zkracování pomocí středníku se používá, aby nemusel být stále dokola opisován stejný subjekt.

Řádek 10 konkrétně popisuje, že toto mapování (a načítání dat z patřičných částí databáze) se provede pouze pokud subjekt našeho SPARQL dotazu bude **odpovídat šabloně** pro iri:biosystem(). Pokud je do SPARQL dotazu potřeba najít jiný subjekt než ten popsaný šablonou, tato část se zcela přeskočí. Na řádku 11 se nachází definice predikátu „rdf:type“ a objektu „bp:Pathway“. Tyto tři části (subjekt z řádku 10 a predikát s objektem z řádku 11) dohromady definují šablonu pro jeden triplet, tedy popisují jeden vztah. **Subjekt je zde popsán funkcí**, to znamená, že celý tvar IRI je **parametrizován**, a může se lišit i ve více než jednom místě. objekt a predikát jsou konstantní IRI. Tento triplet tedy jinými slovy říká, že všechny biosystémy jsou typu „bp:Pathway“.

Na řádku 12 je pak situace obdobná, pouze **objekt** již **není konstantní**, ale vybírá se pro každý subjekt hodnota ze sloupečku biosystem_bases.title (příslušící hodnotě subjektu ze sloupečku „biosystem_bases.id“). Toto je příklad vazby s kardinalitou 1:1, kde každý biosystém má nejvýše jeden název, tedy je možné názvy mít přímo v základní tabulce.

Na řádku 13 je také **nekonstantní objekt**, ale pro změnu tvořen funkcí „iri:source()“.

Na řádku 14 je pak stejná situace jako na předchozím řádku, ale navíc má **omezující podmínku** (viz řádek 15), že se tento vztah vygeneruje pouze pokud pro zadaný biosystém v databázi existuje organismus („biosystem_bases.organism“ není NULL). Pokud je hodnota NULL, triplet se nevygeneruje.

Řádek 17 nám pak popisuje další triplety. Tentokrát se však na rozdíl od prvního případu na řádku 10 týkají vazeb typu M:N, tedy každý biosystém se může skládat z několika komponent. Hodnoty subjektu se proto tentokrát berou z „biosystem_components.biosystem“ - viz parametr funkce.

Toto bylo několik základních konstruktů, pomocí kterých se tvoří mapovací soubory. Další mapovací soubory (pro ostatní domény) jsou také založeny na obdobných principech, pouze rozsáhlejší či složitější na ukázku.

Ukázka kódu 2.1: mapping/biosystem.sql

```
1  sparql
2  alter quad storage virtrdf:PubchemQuadStorage
3          from DB.rdf.biosystem_bases as biosystem_bases
4          from DB.rdf.biosystem_components as biosystem_components
5          from DB.rdf.biosystem_references as biosystem_references
6          from DB.rdf.biosystem_matches as biosystem_matches
7  {
8          create map:biosystem as graph pubchem:biosystem
9          {
10                 iri:biosystem(biosystem_bases.id)
11                         rdf:type bp:Pathway ;
12                         dcterms:title biosystem_bases.title ;
13                         dcterms:source iri:source(biosystem_bases.source) ;
14                         bp:organism iri:taxonomy(biosystem_bases.organism)
15                         where (^{biosystem_bases.}^.organism is not null) .
16
17                 iri:biosystem(biosystem_components.biosystem)
18                         bp:pathwayComponent iri:biosystem(biosystem_components.component) .
19
20                 iri:biosystem(biosystem_references.biosystem)
21                         cito:isDiscussedBy iri:reference(biosystem_references.reference) .
22
23                 iri:biosystem(biosystem_matches.biosystem)
24                         skos:exactMatch iri:wikipathway(biosystem_matches.wikipathway) .
25         }.
26 }.;
```

2.4.2 IRI-classes soubory

Další část potřebná pro efektivní zajištění mapování jsou soubory s popisem jednotlivých IRI tříd, tedy šablona nebo pattern, podle kterého vzniká výsledná IRI. Na tento soubor se odkazují mapovací soubory (viz výše), pokud není IRI konstantní, ale je třeba jí vytvořit na základě dat z relační tabulky nějakou funkcí. Pro příklad jsem tentokrát zvolil soubor2.2 popisující IRI třídy bioassays.

Další význam tohoto souboru je optimalizační, při překladu SPARQL dotazu se engine podívá na tento soubor, a zjistí, jestli požadovaná IRI odpovídá některé z masek, a lze ji tedy tímto způsobem získat. V tomto souboru by hledaná IRI musela být v podobě začínající tímto řetězcem *http://rdf.ncbi.nlm.nih.gov/pubchem/bioassay/AID* (viz řádek 2 a 12-14). Pokud požadovaná IRI tímto řetězcem nezačíná, pak víme, že IRI class nemůže toto IRI vytvořit pro žádná svoje vstupní data. Pokud ovšem požadovaná IRI skutečně odpovídá masce, zde zjistíme, jakým způsobem se má vygenerovat z relačních dat správná hodnota.

Ukázka kódu 2.2: iri-classes/bioassay.sql

```
1  sparql
2  create iri class iri:bioassay "http://rdf.ncbi.nlm.nih.gov/pubchem/bioassay/AID%d"
3          (in bioassay integer not null) option (bijection) .;
4
5
6  sparql
7  create iri class iri:bioassay_data using
8          function db.rdf.iri_bioassay_data (in id1 integer, in id2 integer) returns varchar,
9          function db.rdf.iri_bioassay_data_INV_1 (in id varchar) returns integer,
10         function db.rdf.iri_bioassay_data_INV_2 (in id varchar) returns integer
11         option (bijection,
12                 returns "http://rdf.ncbi.nlm.nih.gov/pubchem/bioassay/AID%d_Description"
13                 union "http://rdf.ncbi.nlm.nih.gov/pubchem/bioassay/AID%d_Protocol"
14                 union "http://rdf.ncbi.nlm.nih.gov/pubchem/bioassay/AID%d_Comment" ).;
```

2. řádek v příkladu představuje IRI v zadaném tvaru a **%d** nakonci značí, že řetězec končí celým číslem, které se vezme ze vstupního parametru funkce. Jedna IRI tak může být složena i z více proměnných. Pokud tedy bude požadovaná IRI přesně tohoto formátu, vyhodnocovací engine pozná, že v mapovacích souborech ho mají zajímat triplety, které odpovídají IRI class *iri:bioassay()*.

druhá sekce (řádek 6-14) potom značí, že pokud hledaná IRI má takový začátek, ale nakonci má navíc jedno z klíčových slov „_Description“, „_Protocol“ nebo „_Comment“, bude to odpovídat funkci *iri:bioassay_data()* v mapovacím souboru.

Tento typ souborů s IRI třídami se tedy používá s dvojím účelem. Jedním směrem jako předpis, jak tvořit IRI z relačních dat (a naopak) podle potřeby, druhým směrem pro optimalizaci vyhodnocení SPARQL dotazů. Klíčovým slovem bijection u definice šablony se navíc zefektivní vyhledávání, protože optimalizátor ví, že tento tvar IRI si odpovídá navzájem právě s jednou mapovací funkcí s jedněmi parametry.

Jak je vidět z řádku 8, tak má funkce iri:bioassay_data() dva vstupní parametry. Jaké varianty IRI můžou vzniknout nám ukazuje nápověda pro optimalizátor na řádcích 12, 13 a 14, ale jak konkrétně se o konkrétním výsledku rozhodne? Pro tuto informaci musíme jít do poslední kategorie souborů potřebných k mapování, a to sice IRI funkcí.

2.4.3 IRI-functions soubory

IRI-functions soubory jsou u některých domén potřeba, pokud mají IRI těchto domén nějaký netriviální formát, nebo pravidla s jakými jsou tvořeny. Ve třetí ukázce2.3 je část souboru s funkcemi pro bioassay. Mapovací funkce iri_bioassay_data() dva vstupní parametry, a přesný tvar IRI je složitější odvodit, proto není popsán přímo v iri-classes, ale až zde, mezi funkcemi.

Hlavní motivace pro takto složité generování IRI je potřeba co největší shody s originálními daty.

Ukázka kódu 2.3: functions/bioassay.sql

```
1  create function iri_bioassay_data (in bioassay integer, in type integer) returns varchar
2  {
3  vectored;
4
5  if(type = 136)
6  return sprintf ('http://rdf.ncbi.nlm.nih.gov/pubchem/bioassay/AID%d_Description', bioassay);
7
8  if(type = 1041)
9  return sprintf ('http://rdf.ncbi.nlm.nih.gov/pubchem/bioassay/AID%d_Protocol', bioassay);
10
11 if(type = 1167)
12 return sprintf ('http://rdf.ncbi.nlm.nih.gov/pubchem/bioassay/AID%d_Comment', bioassay);
13
14 return null;
15 };
```

V této doméně je naštěstí převod poměrně jednoduchý. První proměnná představuje **ID** bioassaye, druhá proměnná potom **typ** bioassaye. Pokud je hodnota proměnné *typ* rovna 136, vytvoří se IRI s „_Description“ na konci. Pokud má typ 1041, IRI má na konci „_Protocol“. V případě, že je typ 1167, bude se jednat o komentář, a tedy na konci IRI bude řetězec „_Comment“. Samotná proměnná bioassay se dosadí za „%d“ jako obvykle, viz zápis funkce pro výpis sprintf().

V dalších doménách se ale používají i mnohem složitější konstrukce pro tvorbu IRI. Kombinují se různé prefixy a části celých IRI. Tvar výsledné IRI může záviset i na několika různých proměnných. Navíc jsou používány i různé formátovací filtry, třeba jako zarovnání ID na 8 číslic, doplněných zleva nulami, získání absolutní hodnoty parametru a podobně.

2.5 D2RQ

D2RQ Platform[22] je jeden ze systémů, poskytující možnost pracovat s obsahem relační databáze jako s „linked data“ pomocí SPARQL bez potřeby data duplikovat. D2RQ je free opensource projekt[22] napsaný v jazyce Java, takže jej lze v případě potřeby měnit a optimalizovat. V rámci této práce je použita jako relační databáze **postgreSQL** D2RQ umožňuje vytvořit také samostatný dump relační databáze do RDF formátu, a tím umožnit vyhodnocovat SPARQL dotazy přímo proti grafové podobě dat. Ale pro naši potřebu je vhodnější, stejně jako u Virtuosa, vytvořit mapování, které bude moci zajistit překlady dotazů ze SPARQL do SQL a mít pouze jedno uložiště, a to sice to v relační podobě.

Mapování můžeme vytvořit buď ručně, za pomoci vlastního *D2RQ Mapping Language*, nebo si ho nechat vygenerovat pomocí D2RQ funkce *generate-mapping*. Funkce generate-mapping je implementace *direct mapping*, který popisuje jak ze schémat a dat relační databáze automaticky vygenerovat sémantické struktury. Základní myšlenka je taková, že se triplet složí jako trojice „*primární klíč - název atributu - hodnota atributu*“. Například z tabulky uživatelů by vypadal triplet třeba takto „*uzivatel_ID:5 - Jmeno - Karel Novák*“.

D2RQ přidává různé možnosti při tvorbě mapování jako například vynechání některých schémat, tabulek či sloupců. Bohužel, našim cílem je, aby rozhraní poskytovalo co nejpodobnější formát dat, jako mají originální data z PubChemRDF projektu. Navíc takto provedené automatické mapování je vhodné například při vystavování jednoduché databázové struktury na webu, kde jsou pouze jednoduché atributy a vazby, a jsou čitelné a pochopitelné již z metadat v relační databázi. PubChem data jsou bohužel velmi komplexní a už jen aby bylo možné je do relační databáze uložit je potřeba některé informace různě zakódovat do přidaných sloupců a podobně.

Proto v rámci práce použiji opět možnost manuálně vytvořených pravidel pro mapování z relační podoby do RDF.

2.6 Načtení dat do postgreSQL v relační podobě

Pro načtení do postgreSQL databáze jsou použita stejná data z PubChemRDF projektu, která byla použita v nástroji Virtuoso. Načítání a konverze nativních RDF dat do relační databáze postgreSQL je realizováno opět pomocí mírně upravených Java tříd za pomoci Apache Jena. Tyto třídy zajistí rozparsování všech potřebných RDF souborů, a triplety uloží do relační databáze ve správné formě. Každá doména PubChem dat má opět svou vlastní třídu a vlastní způsob jak se správně zkonvertuje.

Opět je zde třeba rozlišovat vazby typu 1:1 a M:N, a podle toho připravit dopředu všechna potřebná databázová schémata a tabulky (doménové **_base* tabulky a tabulky vazební pro N:M vztahy).

2.7 Mapování relačních dat do RDF podoby v D2RQ

Mapování opět slouží ke kompozici sémantické struktury z relační podoby rozdělené do jednotlivých tabulek a sloupců. Cílem je přiblížit se co nejvíce struktuře originálního RDF repozitáře poskytovaného projektem PubChemRDF[16] a zajistit tak zpětnou kompatibilitu.

Jako výsledek potřebujeme rozhraní interpretující SPARQL dotazy co nejpodobněji originálnímu rozhraní, ale s postgreSQL relační databází jako uložištěm dat.

Ukázka kódu 2.4: D2RQ-prefixes

```
1  @prefix map: <http://example.com/test/> .
2  @prefix : <http://annotation.semanticweb.org/iswc/iswc.daml#> .
3
4  @prefix vocab: <vocab/> .
5  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
6  @prefix d2rq: <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#> .
7  @prefix dcterms: <http://purl.org/dc/terms/> .
8
9  @prefix conserveddomain: <http://rdf.ncbi.nlm.nih.gov/pubchem/conserveddomain/>.
10 @prefix compound: <http://rdf.ncbi.nlm.nih.gov/pubchem/compound/>.
```

D2RQ má vlastní jazyk popisující mapování SPARQL dotazů. Na rozdíl od Virtuosa lze ovšem použít pouze jeden soubor (v Turtle notaci) a nelze využívat složitější funkce pro pokročilejší formátování a vyhodnocování podmínek. Nelze zde tedy rozdělit mapování do více souborů jako ve Virtuosu, ale všechno musí být v jednom mapovacím souboru pouze za využití D2RQ konstrukcí.

2.7.1 Mapovací soubor pro D2RQ

Na názorných ukázkách předvedu opět klíčové základní konstrukce v mapovacím souboru. Jelikož samotné mapování se skládá z několika tisíc řádků, ukázky jsou jen výňatkem několika řádků z téhož mapovacího souboru.

Definice prefixů

na ukázce ze začátku mapovacího souboru2.4 vidíme část prefixů. Prefixy slouží ke zkracování zápisu IRI. Společná část pro všechny prvky z jedné domény se může označit tímto způsobem jako @prefix a definuje se její zkrácená podoba, poté lze ve zbytku souboru používat tuto zkrácenou podobu, a ručně psát pouze konec IRI, tedy část, která se u jednotlivých elementů v doméně bude lišit. Na prvním řádku je prefix (@prefix map:) označující celé mapování jako jeden velký graf zajišťující překlad.

V prostřední části (řádek 4 až 7) jsou pak globálně uznávané a standardizované IRI z různých oblastí zkrácené pomocí prefixů. Plus D2RQ má také vlastní IRI pro interní popis objektů a vztahů.

Poslední část2.4 v příkladu (řádek 9 a 10) popisuje již identifikátory z domény PubChem, viz plný tvar IRI. Tyto identifikátory se řídí buď standardem a nebo jsou přímo převzaty z projektu PubChemRDF.

Ukázka kódu 2.5: D2RQ-initialisacion

```
1  map:Configuration a d2rq:Configuration;
2  d2rq:useAllOptimizations true. #umozni optimalizace behem prekladu
3
4  map:database a d2rq:Database; #zalozeni promenne map:database s nasledujicimi vlastnostmi
5  d2rq:jdbcDriver "org.postgreSQL.Driver"; #soubor s driverem pro nasi DB
6  d2rq:jdbcDSN "jdbc:postgreSQL://address.com"; #adresa, na kterou se posilaji dotazy
7  d2rq:username "username"; #prihlasovaci jmeno
8  d2rq:password "password"; #heslo
9  jdbc:keepAlive "3600"; #timeout
10
11 d2rq:resultSizeLimit 20; #omezeni poctu vracenych vysledku na strane SQL databaze
12 .
```

Konfigurační část

Po seznamu prefixů následuje konfigurační část mapování - viz ukázka2.5. Konfigurace začíná dvěma řádky pro aktivaci všech optimalizací při překladu ze SPARQL jazyka do SQL.

Nastavení databázového připojení probíhá na řádcích 4-9, kde se konfigurují základní parametry pro připojení. Použitý driver (v našem případě pro postgreSQL), adresa vzdálené (nebo lokální) databáze, přihlašovací údaje . Dále je důležitý název/entita spojení, pod kterým se na něj budeme odkazovat, což je na řádku 4 „map:database“.

Velmi důležité pro ladění je u velkých dat také omezení na řádku 11, které za každý přeložený dotaz v SQL ještě přidá LIMIT 20. V praxi je omezení výsledků spíše nežádoucí, tudíž se tento parametr ve finále nepoužije. LIMIT lze použít na omezení počtu výsledků i přímo ve SPARQL dotazu, ale tento limit se bohužel v mapování nepřeloží. SQL dotaz se tak vyhodnotí bez limitu, a celý výsledek (klidně stovky tisíc záznamů) se posílá z databáze do enginu D2RQ. Až ten pak ořízne a vrátí požadovaných 20 výsledků. U malého počtu dat to zřejmě moc nevadí (i když to není optimální a plýtvá to zdroji serveru), ale v rozsáhlých datech ze systému PubChem to často znamená zbytečné posílání gigabytů dat po síti.

Mapování tripletů

V mapování se výsledný triplet skládá ze dvou částí. Část se subjektem, jak ukazuje příklad 2.6, a potom část s predikátem a objektem dohromady, jak ukazuje 2.7.

V subjektové části (2.6) je opět patrné pojmenování subjektu jako *map:Bioassay* (toto pojmenování je pouze interní, a má funkci jako název proměnné, na kterou se později díky tomu lze odkázat).

Dále se definuje databáze, odkud se data berou.

Ukázka kódu 2.6: D2RQ-mapping subject

```
map:Bioassay a d2rq:ClassMap; #nazev mapovani
d2rq:dataStorage map:database; #pracuje se s tabulkami z vyse definovane databaze
d2rq:class bao:BAO_0000015; #typ generovanych entit
d2rq:uriPattern "http://rdf.ncbi.nlm.nih.gov/pubchem/bioassay/AID@@bioassay_bases.id@@";
#pattern pro generovani IRI s promennou uvnitr
.
```

d2rq:class je speciální konstrukce jazyka D2RQ pro popis typu subjektu.

A na posledním řádku se nachází nejdůležitější část, a to sice vzor pro generování IRI. První část je konstantní řetězec, část mezi dvojitými zavináči označuje sloupeček v databázi (bioassay_bases.id). Tento vzor říká, že pokud bude SPARQL dotaz požadovat například titulek bioassay s IRI

http://rdf.ncbi.nlm.nih.gov/pubchem/bioassay/AID75528,

D2RQ vyhodnotí, že souhlasí s touto maskou, a ve sloupečku *bioassay_bases.id* se pokusí najít hodnotu *75528*.

V objektové části (2.7) je na prvním řádku opět název tohoto elementu mapování. Druhý řádek určuje, ke kterému subjektu se dané informace vztahují (má na ně v RDF datech vazbu). *d2rq:property* určuje typ vazby mezi subjektem a objektem, neboli predikát. V tomto případě se jedná o *dcterms:title*, neboli vazbu na název. Poslední dvě hodnoty *column* a *datatype* označují způsob, jakým se získá objekt. Tedy že je to přímo hodnota sloupce *bioassay_bases.title*, který je typu string. V IRI identifikátoru objektu lze také použít nějaký základní vzor stejně jako u subjektu, ale pro predikát lze používat pouze konstantní IRI bez proměnných.

Toto jsou pouze základní konstrukce, které lze použít pro popis tripletu.

Nejčastěji z dalších konstruktů je také použitá konstrukce objektu s podmínkou, takže subjekt má nějakou vazbu na objekt pouze v zadaném případě. Jelikož zde nejde použít IF podmínku jako v programovacích jazycích, řeším tímto způsobem například rozlišení, zda se jedná o popis bioassaye, protokol, či komentář. IRI jsou stejné, pouze mají na konci *_description*, *_protocol* nebo *_comment*. Rozlišení zajistí tři různé definice objektů, každá s jinou podmínkou, např [*d2rq:condition "bioassay_data.type = 1167";*] Tedy že dotyčný objekt se vygeneruje pouze při patřičném typu vazby v databázi.

Nedostatky mapování

Cílem mapování je, přiblížit se co nejblíže originální podobě RDF dat z projektu PubChemRDF. Jelikož je to ale velmi komplexní doména chemických sloučenin a malých molekul, některé IRI jsou poměrně složité a skládají se z několika částí.

Ukázka kódu 2.7: D2RQ-mapping object

```
map:BioassayTitle a d2rq:PropertyBridge; #nazev mapovani
d2rq:belongsToClassMap map:Bioassay; #ke kteremu subjektu nalezi
d2rq:property dcterms:title; #IRI adresa predikatu
d2rq:column "bioassay_bases.title"; #hodnota objektu je primo hodnota sloupce
d2rq:datatype xsd:string; #sloupec je typu string
.
```

Jelikož je v jazyce D2RQ jen poměrně omezená podpora například pro práci se stringy a vyhodnocování podmínek, některé komplexnější IRI bylo potřeba zjednodušit, a tedy se liší od originálu. Pokud je v IRI například více proměnných či hodnota proměnné nějak ovlivňuje celkovou podobu IRI, je poměrně problém tuto situaci zachytit omezenou vyjadřovací sadou jazyka D2RQ. Například některá IRI jsou parametrizována třemi proměnnými. Jedna proměnná určuje, zda je na konci IRI napsáno *_comment* či jiné klíčové slovo. Další proměnná určuje například jaký bude začátek IRI, a zda se nějaká část identifikátoru vůbec vygeneruje. Třetí hodnota pak může být vypsána v absolutní hodnotě, nebo jinak upravena.

V SQL funkci je toto ošetřeno pár řádky s podmínkami, ale v D2RQ by v lepším případě bylo potřeba duplikovat pro každou možnou kombinaci celou část s objektem a napsat k ní rozsáhlou podmínku. Počet těchto duplikátů (různých šablon IRI), každý se svou unikátní podmínkou, by se zvětšoval násobně spolu s počtem proměnných. Některé proměnné mají navíc ještě různá pravidla, podle toho jakou mají hodnotu. Například zda jde o pozitivní, negativní, nulovou respektive maximální či minimální hodnotu Integeru.

Jelikož je Virtuoso momentálně primární systém, pro účely základních testů v rámci práce některé odchylky od formátování zdrojových dat nebudu opravovat k dokonalosti. Klíčové je, zda dotazy vrací správně všechny požadované výsledky (jen trochu jinak naformátované) a za jak dlouhou dobu.

Další problematická část je speciální formátování čísel. V originálních datech jsou například vygenerovaná číselná ID entit v IRI zarovnána na pevný počet číslic (6-9 číslic, v závislosti na konkrétním IRI). Toho ale bohužel nelze na rozdíl od SQL funkcí v jazyce D2RQ jednoduše dosáhnout, a číselné ID generuji bez uvozovacích nul. Například „*d2rq:class bao:BAO_0000015;*“ se vygeneruje pouze jako IRI „*bao:BAO_15;*“, čímž se bude tato entita lišit od originálu. V případě, že s tím ale budeme počítat, a všude jsou IRI generovány stejně, tak se data lokálně provážou bez problému (ořezané počáteční nuly budou mít všechny entity, takže na sebe budou zase pasovat).

Tyto nedostatky se ovšem týkají pouze několika málo predikátů a objektů (soubor s kompletním výčtem změn je přiložen na CD u této práce).

Kapitola 3

Sada testovacích dotazů

3.1 Výběr vhodných dotazů

Sada testovacích dotazů by měla představovat typické požadavky na databázi malých molekul a chemických sloučenin PubChem.

Jelikož nejsem odborníkem v oblasti biochemie, pro testování a optimalizaci výkonu jsem využil doporučenou sadu SPARQL dotazů přímo z projektu PubChemRDF[16].

Nejčastěji se jedná o vyhledávání látek s určitou vlastností, či látek, které reagují s konkrétními proteiny za požadovaných podmínek. Například vyhledání všech účinných látek inhibujících jeden z proteinů nacházejících se ve zkoumané bakterii. Dále pak vyhledávání různých podobností v chemických vlastnostech a složení.

Když hledáme účinné látky fungující proti konkrétnímu proteinu, mohou nám pomoci látky účinné proti podobným proteinům. Nebo naopak nechceme, aby látka poškodila některý z proteinů potřebných pro lidský organismus, ale kvůli podobnosti by k tomu mohlo dojít.

Poté jsou také z dat agregovány různé statistiky o počtu provedených měření, kolik látek bylo proti danému proteinu testováno atp.

Testovací sadu dotazů bylo potřeba nejprve upravit tak, aby obsahovala pouze funkční IRI popisující aktuálně poskytovaná data. Bez toho některé dotazy nevrátí žádné výsledky. Dále pak byly přímo na dotazech provedeny optimalizace popsané ve zvláštní kapitole Optimalizace.

Na těchto vzorových dotazech probíhalo měření všech nástrojů. Virtuosa s nativními RDF daty i s mapováním do relační formy a D2RQ s mapováním. Nakonec z těchto dotazů byla odvozena i ručně optimalizovaná SQL podoba.

- **Query 1:** Které proteiny z databáze inhibuje *donepezil* (v databázi uložen pod označením CHEBI_53289) s IC50[4] menším než 10 mikromolů na litr?

Ukázka kódu 3.1: SPARQL testing query 1

```
SELECT distinct ?protein ?title
WHERE { ?sub rdf:type obo:CHEBI_53289 ; obo:BFO_0000056 ?mg .
  ?mg obo:BFO_0000057 ?protein ; obo:OBI_0000299 ?ep .
  ?protein rdf:type bp:Protein ; dcterms:title ?title .
  ?ep rdf:type bao:BAO_0000190 ; obo:IAO_0000136 ?sub ; sio:has−value ?value .
  filter (?value < 10 ) }
```

Donepezil je medikament používaný například při paliatické léčbě Alzheimerovy choroby. *Query 1* nám zjistí, které všechny proteiny je tato látka schopna inhibovat alespoň o 50% při koncentraci této látky menší, než 10 mikromolů na 1 litr.[7]

- **Query 2:** Které farmakologické role jsou v ChEBI definovány pro substanci SID46505803 ?

Ukázka kódu 3.2: SPARQL testing query 2

```
SELECT DISTINCT ?rolelabel
WHERE { substance:SID46505803 sio:CHEMINF_000477 ?comp .
  ?comp rdf:type ?chebi .
  ?chebi rdfs:subClassOf [ a owl:Restriction ;
  owl:onProperty obo:RO_0000087 ; owl:someValuesFrom ?role ] .
  ?role rdfs:label ?rolelabel . }
```

- **Query 3:** Která sloučenina má farmakologickou roli NSAID („Nonsteroidal anti-inflammatory drug“ definovanou v ChEBI) a zároveň molární hmotnost menší než 200g/mol?

Ukázka kódu 3.3: SPARQL testing query 3

```
SELECT distinct ?compound
WHERE { ?compound rdf:type ?chebi .
  ?chebi rdfs:subClassOf [ a owl:Restriction ;
  owl:onProperty obo:RO_0000087 ; owl:someValuesFrom obo:CHEBI_35475 ] .
  ?compound sio:has−attribute ?MW .
  ?MW rdf:type sio:CHEMINF_000334 .
  ?MW sio:has−value ?MWValue .
  filter (?MWValue < 200 ) }
```

- **Query 4:** Která substance má farmakologickou roli NSAID (definovanou v ChEBI) a kdo o ní poskytl informace do systému?

[7] Mol je jednotka SI, popisující množství chemické látky (počet částic). 1 mol látky odpovídá zhruba $6*10^{23}$ částic (atomů, molekul, iontů) = Avogadrova konstanta. Toto číslo reprezentuje množství atomů ve 12 gramech uhlíku C^{12}[1].

Ukázka kódu 3.4: SPARQL testing query 4

```
SELECT DISTINCT ?substance ?source
WHERE { ?substance dcterms:source ?source .
  ?source dcterms:subject concept:Chemical_Vendors .
  ?substance rdf:type ?chebi .
  ?chebi rdfs:subClassOf [ a owl:Restriction ;
  owl:onProperty obo:RO_0000087 ;
  owl:someValuesFrom obo:CHEBI_35475 ] . }
```

- **Query 5:** Které proteiny z databáze jsou inhibovány látkami s IC50[4] menším než 10 mikromolů na litr, které mají zároveň v ChEBI roli „cholinesterase inhibitors“?

Ukázka kódu 3.5: SPARQL testing query 5

```
SELECT distinct ?title
WHERE { ?chebi rdfs:subClassOf [ a owl:Restriction ;
  owl:onProperty obo:RO_0000087 ; owl:someValuesFrom obo:CHEBI_37733 ] .
  ?sub rdf:type ?chebi ; obo:BFO_0000056 ?mg .
  ?mg obo:BFO_0000057 ?protein ; obo:OBI_0000299 ?ep .
  ?protein rdf:type bp:Protein ; dcterms:title ?title .
  ?ep rdf:type bao:BAO_0000190 ; obo:IAO_0000136 ?sub ;
  sio:has−value ?value .
  filter (?value < 10 ) }
```

- **Query 6:** Které chemické látky (substance) jsou schopny inhibovat proteiny podobné proteinu *GI548481* (=Prostaglandin G/H synthase 1) s funkční doménou PSSMID188648 (=Prostaglandin endoperoxide synthase)?

Ukázka kódu 3.6: SPARQL testing query 6

```
select distinct ?substance ?protein (avg(?value) as ?valueX)
where { ?substance obo:BFO_0000056 ?measuregroup .
  ?measuregroup obo:BFO_0000057 ?protein .
  protein:GI129900 vocab:hasSimilarProtein ?protein .
  ?protein obo:BFO_0000110 conserveddomain:PSSMID188648 .
  ?measuregroup obo:OBI_0000299 ?endpoint .
  ?endpoint obo:IAO_0000136 ?substance .
  ?endpoint rdf:type bao:BAO_0000190 .
  ?endpoint sio:has−value ?value . }
group by ?substance ?protein
```

Tento protein (hormon) je součástí téměř všech organických tkání člověka i většiny zvířat. Ovlivňuje lokálně řadu jevů, jako prokrvení, srážení krve a zánětlivé procesy[3]. Tyto proteiny jsou tvořeny během chemických reakcí probíhajících v místě poranění, či jiných problémů.

- **Query 7:** Které proteiny z databáze jsou inhibovány látkami s IC50[4] menším než 10 mikromolů na litr, a mají stejnou standardizovanou strukturu (CID3152)?

Ukázka kódu 3.7: SPARQL testing query 7

```
1  select distinct ?sub ?protein ?title
2  where { ?sub sio:CHEMINF_000477 compound:CID3152 ;
3    obo:BFO_0000056 ?mg .
4    ?mg obo:BFO_0000057 ?protein ;
5    obo:OBI_0000299 ?ep .
6    ?protein rdf:type bp:Protein ;
7    dcterms:title ?title .
8    ?ep rdf:type bao:BAO_0000190 ;
9    obo:IAO_0000136 ?sub ;
10   sio:has-value ?value .
11 filter (?value < 10 ) }
12 }
```

Zde se pracuje s konkrétními chemickými látkami (substances) od konkrétních distributorů, které jsou odvozeny od standardizované sloučeniny CID3152 (compound donepezil). Zajímá nás, které všechny proteiny jsou tyto látky schopné inhibovat s IC50 při koncentraci menší než10 μMol/litr.

- **Query 8:** Které chemické látky inhibují proteiny zapojené do biologického procesu pathway:prostaglandin (biosyntetický proces GO:0001516) s IC50 při koncentraci této látky menší než10 μMol/litr?

Ukázka kódu 3.8: SPARQL testing query 8

```
1  select distinct ?substance ?protein
2  where { ?substance obo:BFO_0000056 ?measuregroup .
3    ?measuregroup obo:BFO_0000057 ?protein .
4    ?protein obo:BFO_0000056 obo:GO_0001516 .
5    ?measuregroup obo:OBI_0000299 ?endpoint .
6    ?endpoint obo:IAO_0000136 ?substance .
7    ?endpoint rdf:type bao:BAO_0000190 .
8    ?endpoint sio:has-value ?value .
9    filter (?value < 10) }
```

- **Query 9:** Jaké farmakologické role definované v ChEBI mají látky, které inhibují protein s označením GI17531135 s IC50 při koncentraci této látky menší než10 μMol/litr?

Ukázka kódu 3.9: SPARQL testing query 9

```
SELECT distinct ?rolelabel
WHERE { ?sub obo:BFO_0000056 ?mg .
  ?mg obo:BFO_0000057 protein:GI7531135 ;
  obo:OBI_0000299 ?ep .
  ?sub rdf:type ?chebi .
  ?chebi rdfs:subClassOf _:I . _:I a owl:Restriction .
  _:I owl:onProperty obo:RO_0000087 .
  _:I owl:someValuesFrom ?role .
  ?role rdfs:label ?rolelabel .
  ?ep obo:IAO_0000136 ?sub ;
  rdf:type bao:BAO_0000190 ;
  sio:has-value ?value .
  filter (?value < 10 ) }
```

- **Query 10:** Statistický souhrn celkového počtu chemických látek testovaných v rámci PubChem databáze proti každému proteinu. (toto může být časově velmi náročná operace)

Ukázka kódu 3.10: SPARQL testing query 10

```
select (count(?sub) as ?subcnt) ?protein
where { ?sub obo:BFO_0000056 ?mg .
  ?mg obo:BFO_0000057 ?protein .
  ?protein rdf:type bp:Protein .
  ?mg obo:OBI_0000299 ?ep .
  ?ep rdf:type bao:BAO_0000190 ; obo:IAO_0000136 ?sub ; sio:has-value ?value .
}
group by ?protein
order by desc (count(?sub))
```

Kapitola 4

Optimalizace

Převést původní data do relační podoby lze více různými způsoby, stejně tak poté řešit jejich mapování a uložení v databázi.

Také přesné znění dotazu ve SPARQL může ovlivnit výsledný překlad do SQL (nebo nativní vyhodnocení) a tedy i výkon celého procesu. V této kapitole se pokusím popsat některé z možných a prováděných optimalizací.

V rámci diplomové práce se zaměřuji zejména na následující optimalizace:

4.1 Optimalizace výkonu na úrovni překladu mapování

Překlad ze SPARQL dotazu na SQL dotaz provádí enginy virtuosa a D2RQ různým způsobem. Navíc mají oba nástroje jiný popis mapování i vyjadřovací prostředky.

4.1.1 Eliminace nadbytečných unionů

Některým tripletům ze SPARQL dotazů můžou odpovídat data z více tabulek, to vede přirozeně na union těchto tabulek. Obvykle ale díky bližší specifikaci dalších tripletů lze některé z těchto tabulek úplně vynechat.

Například triplet

```
?X :has_label ?L
```

popisuje vlastnost „has_label“, která podle mapování odpovídá datům z tabulek protein_label a gene_label.

Když se ale ve SPARQL dotazu nachází navíc triplet

```
?X rdf:type :Protein
```

víme, že *?X* je protein, a má tedy smysl hledat label pouze v tabulce protein_label.

Naivní mapování vyhodnotí každý triplet zvlášť, a výsledky zjoinuje:

(PROTEIN_BASES) ∩ (PROTEIN_LABEL ∪ GENE_LABEL)

Nejprve se provede sjednocení (union) tabulek labelů obou entit, a nakonec se provede join s tabulkou protein_bases.

Pokud se ovšem „roznásobí“ výpočet na jednotlivé join operace dvojic tabulek, lze provést optimalizace na základě pravidel mapování. Nebudou se tak zbytečně hledat labely v tabulce gene_label, protože engine pozná, že hledá label pro protein, nikoliv pro gen.

$$(\texttt{PROTEIN\_BASES} \cap \texttt{PROTEIN\_LABEL}) \cup (\cancel{\texttt{PROTEIN\_BASES} \cap \texttt{GENE\_LABEL}})$$

Některé joiny je tak možno úplně vyloučit, a snížit tím značně výpočetní náročnost vyhodnocení celého dotazu. Tyto optimalizace byly aplikovány v ručně převedených SQL dotazech, a i oba nástroje je do jisté míry provádí.

4.1.2 Nadbytečné selfjoiny

Během překladu pomocí mapování dochází také k tomu, že se při získávání více atributů téhož záznamu (v jedné tabulce) použije selfjoin celé tabulky, namísto aby se vzaly rovnou oba atributy najednou.

Například:

Ukázka kódu 4.1: Překlad s nadbytečným selfjoinem

```
1  select A.name, B.value
2  from Table as A, Table as B
3  where A.id == B.id and A.criterion == 10;
```

Zde jsou obě hodnoty, „name“ i „value“ v tabulce „table“, a tedy je zbytečné provádět selfjoin nad touto tabulkou, Protože join probíhá na základě klíče.

Optimalizovaný dotaz bez zbytečného joinu vybere obě hodnoty naráz:

Ukázka kódu 4.2: Optimalizace bez zbytečného selfjoinu

```
1  select name, value from Table where criterion == 10;
```

Přebytečné selfjoiny jsou v ručně vytvořených SQL dotazech odstraněny, a D2RQ i Virtuoso se je snaží nevytvářet, když k tomu mají dostatek informací.

4.1.3 Nevyužívání znalosti o cizím klíči

Další častá neoptimální situace nastává při práci s vazebními M:N tabulkami. Pokud chceme ID entity, a výsledek měření, které jsou uloženy v tabulce TABLE_MEASUREMENT, kontroluje se stejně existence entity proti základní tabulce (TABLE_BASES). A to i když je ID cizím klíčem do základní tabulky, a tedy už víme, že existuje. Vznikají tak proto další nadbytečné join operace, kvůli redundantnímu ověřování existenci entity. Například:

Ukázka kódu 4.3: nevyužívá se znalost o cizím klíči

```
1  select TABLE_MEASUREMENT.value, TABLE_MEASUREMENT.id
2  from TABLE_MEASUREMENT, TABLE_BASES
3  where TABLE_MEASUREMENT.id = TABLE_BASES.id
```

A přitom stačí vzít obě požadované hodnoty přímo z tabulky s měřením, protože díky FK vazbě (a významu tabulky měření, kde nemá smysl měřit neexistující entity) máme jistotu existence entity v BASE tabulce.

Viz:

Ukázka kódu 4.4: zde už se ID bere z tabulky měření

```
1  select id, value
2  from TABLE_MEASUREMENT;
```

Čímž se opět ušetří další join operace.

V ručních SQL dotazech je toto eliminováno, a základní tabulky jsou používány pouze v případě potřeby. Virtuoso ani D2RQ tento druh optimalizací neprovádí.

4.1.4 IRI generována více způsoby

Převod původních RDF dat do relační podoby je možné provést různými způsoby, a je třeba zvolit vhodnou interpretaci/uložení jednotlivých informací. Původní IRI (textové řetězce) lze do tabulek například většinou uložit jen jako jedno či více čísel nebo řetězců, a původní IRI zase zpětně složit podle předem definovaných pravidel. Tím se ušetří jednak místo na disku/v paměti, ale také lze s čísly dobře pracovat, indexovat je atp.

V některých případech se ovšem stejné IRI může generovat z různých tabulek různými způsoby. Například informace o ChEBI ontologiích lze uchovávat v jednom sloupečku relační databáze v podobě chebi ID, což je výhodné z hlediska prostorové náročnosti, i rychlosti vyhledávání v indexu. Problém může nastat, když některé entity mají možnost mít i jiný typ ontologie nežli ChEBI. V tom případě je potřeba brát ohled nejen na ID v dané ontologii, ale také na typ ontologie (zda odpovídá ChEBI, a nebo je to ID z jiné ontologie). Protože však engine nezná vztah mezi těmito dvěma mapováními (tedy že stačí pouze porovnat hodnoty ID, a ověřit, že typ má případně konstantní hodnotu odpovídající ChEBI). Proto engine nejprve vygeneruje pro každý záznam celou IRI v textové podobě, a až poté porovnává takto vygenerované IRI jako textové řetězce, což je samozřejmě pomalejší, než porovnat dvě celočíselné hodnoty.

V tomto případě pomůže rozdělit mapování na více částí tak, aby měla každá ontologie vlastní pravidlo, použité pouze v případě, že hodnota atributu typ odpovídá právě požadované ontologii. Engine tak může porovnávat již pouze celočíselné hodnoty ID. Tím, že už není nutné vytvářet celou IRI a porovnávat textové řetězce, dochází opět ke zrychlení.

V ručně vytvořeném mapování se porovnávají rovnou celočíselné hodnoty, a nepracuje se s IRI, pokud není potřeba. Virtuoso umožňuje tuto úpravu, a byla provedena pro ChEBI ontologii, kde docházelo ke zbytečným výpočtům navíc v ukázkových dotazech. V praxi by bylo vhodné vytvořit zvláštní mapování pro všechny případy. Tato optimalizace byla provedena i v D2RQ pomocí condition konstrukce.

4.2 Optimalizace výkonu na úrovni databázových struktur

Jelikož Virtuoso ani D2RQ neoptimalizují dotazy na základě cizích klíčů, vyplatí se pro snížení počtu zbytečných joinů data přesunout do jedné tabulky pokud to jde.

U atributů, které jsou nepovinné, a mají kardinalitu 1:1 či 1:N (více entit má stejnou hodnotu atributu) může být vhodné udržovat tyto atributy ve zvláštní tabulce. Jelikož ale testované nástroje nepracují efektivně s cizími klíči, joinují zbytečně základní tabulku *_bases.

Tudíž byly v rámci optimalizace dány dohromady data z tabulek endpoints_bases a endpoints_measurements do nové tabulky endpoints_merged, která obsahovala data z obou tabulek. Kvůli tomuto kroku se sice tabulka zvětší, a některé atributy jsou často NULL, ale sníží to počet prováděných join operací při práci s endpointy. Po změně bylo potřeba také patřičně upravit mapování pro oba nástroje, aby reflektovalo novou strukturu a názvy tabulek. Výsledek této úpravy na mapování ve Virtuosu je pak vidět v tabulce 5.2 s porovnáním obou variant.

Tento druh úprav dokáže částečně odstranit problém s cizími klíči popsaný výše. Ale to pouze za předpokladu, že se jedná o vazební tabulku s kardinalitou vztahu 1:1 a 1:N, pokud jde o M:N vazbu, do jedné tabulky již tímto způsobem atributy nevměstnáme.

Optimalizace byly také provedeny indexací databáze (již od první verze mapování z ÚOCHB). V relačních databázích byl přidán index a sdružený index na všechny potřebné sloupce, v RDF uložišti Virtuosa pak byly použity základní indexy plus doporučený index z projektu PubChemRDF.

4.3 Přizpůsobení dotazů

Podoba výsledného SQL dotazu, vygenerovaného za pomoci mapování, a jeho exekuční plán se může lišit nástroj od nástroje. Ovšem i drobné úpravy přímo v původním SPARQL dotazu mohou velmi ovlivnit výpočetní čas celého požadavku.

V rámci performance testů pracuji zejména se dvěma různými variantami dotazů. Jedna varianta je se specifikací „FROM“ na začátku dotazu (podobně,

jako v SQL určuje FROM klauzule požadovanou tabulku, ve SPARQL označuje FROM klauzule požadovaný podgraf).

Jak je patrné z výsledků měření v další kapitole, tato úprava může mít poměrně zásadní vliv na rychlost zpracování dotazu. Zatímco v D2RQ a Virtuosu s mapováním tato změna nemá téměř měřitelný vliv na výkon, u Virtuosa s nativní podporou SPARQL dotazů naopak FROM klauzule ovlivní rychlost výpočtu negativně.

Pokud FROM klauzuli u nativního virtuosa vynecháme, je sice rychlejší (nemusí ověřovat, že jsou data z požadovaného grafu), ale v některých případech díky tomu může vrátit jiné výsledky, než jsme požadovali. Proto je potřeba v takových případech některé FROM klauzule zachovat.

Další optimalizace spočívá například u query 8 v odstranění nadbytečného tripletu „?protein rdf:type bp:Protein“. Tuto informaci lze vyvodit z ostatních tripletů, a kontrola navíc jen způsobovala zpomalování během překladu a vyhodnocování dotazu.

Také bylo potřeba přistoupit u query 6 ke změně hledaného výrazu, jelikož se v různých nástrojích pracuje různě s desetinnými čísly, a bez agregační funkce vracel každý nástroj trochu jiný výsledek.

Úprava byla provedena i v query 4, kde se místo dotazu na 3D strukturu ptáme na dodavatele.

4.4 Další úpravy

D2RQ při každém dotazu nejprve načte celý soubor s mapováním, ten zpracuje, a až poté začne SPARQL dotaz překládat do SQL podoby. Tato operace trvá řádově sekundy, a ve výsledných testech není zahrnuta do výsledného času. Pokud by bylo D2RQ používáno v produkčním prostředí, bylo by vhodné zajistit, aby se mapování načetlo a rozparsovalo pouze jednou, a nikoliv při každém dotazu.

Dále D2RQ nepřeloží limit na požadovaný počet výsledků ze SPARQL do SQL, ale pošle do databáze SQL dotaz bez limitu, a až po vrácení všech výsledků zpátky do programu se aplikuje na tyto výsledky limit. To je problém zejména pokud je D2RQ aplikace na jiném fyzickém stroji než databáze, a musí se posílat všechna data po síti zbytečně, a oříznou se až v cíli.

Toto se dá řešit například nastavením pevného SQL limitu pro všechny dotazy přímo v mapovacím souboru, a nebo nastavením limitu pro jednotlivé mapované triplety.

Kapitola 5

Výsledky měření

V této kapitole jsou zpracovány výsledky všech měření výkonu. Jednotlivá měření byla prováděna na testovacím serveru CERIT-SC (https://www.cerit-sc.cz/).

5.1 Specifikace serveru a měření

Server je virtuální stroj se 40 procesorovými jádry, 515 GB operační paměti a operačním systémem CentOS Linux 7.4. Virtuoso ve verzi 07.20.3217, postgreSQL verze 9.2.23 a D2RQ 0.83.

Všechny testy a měření probíhaly automaticky. Pro tuto činnost byl v rámci diplomové práce vyvinut javový testovací nástroj, který se poté spouštěl přímo na serveru, a postupně měřil jednotlivé nástroje a vytvářel statistiky naměřených hodnot. Výsledky by tedy neměla negativně ovlivnit ani latence způsobená síťovým přenosem.

Testy probíhaly v době, kdy na serveru neběžely jiné výkonově náročné aplikace a zároveň byl vždy spuštěn jen jeden měřený nástroj s jednou databází tak, aby byly dostupné stejné prostředky pro všechny testy.

Postupně bylo zkoušeno deset typických dotazů pro databázi malých molekul popsaných ve zvláštní kapitole *Sada testovacích dotazů*. Každý z dotazů měl nejprve dva „zahřívací“, tedy neměřené běhy, a poté, když byl relativně stabilizovaný výkon a nacashované databázové struktury v paměti, následovalo teprve deset měřených běhů. Aby se lépe využila cash (a také eliminoval rozdíl v tom, že některé věci jsou a jiné ještě nejsou v cashi), provedly se vždy všechna potřebná měření na jednom SPARQL dotazu, a až poté se začalo s měřením dalšího.

5.2 Výsledky měření

5.2.1 Naměřená data

Naměřené (průměrné) hodnoty pro všechny testované nástroje a SPARQL/SQL dotazy ukazuje následující tabulka:

Tabulka 5.1: Měření všech nástrojů (čas v ms)

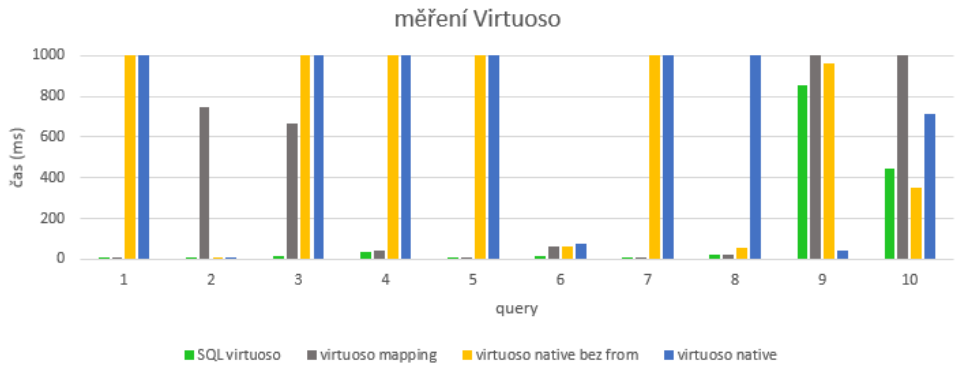
Query	D2RQ	Virtuoso mapping	Virtuoso native bez FROM	Virtuoso native	SQL postgreSQL	SQL Virtuoso
1	8003.61	1.38	2040.70	2675.85	7.10	**0.77**
2	16.34	754.44	0.79	0.84	5.08	**0.33**
3	62.03	749.15	34616.65	42517.16	**3.53**	12.59
4	176.53	**36.02**	35505.89	41625.45	76.57	37.41
5	13920.50	4.25	1535.34	95778.15	1040.43	**4.00**
6	277.35	65.00	63.15	78.62	29.66	**17.43**
7	185.22	0.76	2007.33	2663.68	5.15	**0.57**
8	6857.09	21.55	54.82	1234.50	**19.35**	21.57
9	263.34	4547.65	962.77	45.20	**27.24**	854.16
10	16315.01	214140.82	**354.64**	712.77	1652.94	446.84

Sloupce v tabulce určují měřený nástroj. U mapovaného Virtuosa a D2RQ je pro přehlednost jen hodnota s FROM klauzulí, její odebrání nemělo měřitelný vliv na výkon. U nativního Virtuosa uvádím i variantu bez FROM klauzulí, jelikož zde způsobuje přidání/absence FROM klauzule značné rozdíly ve výkonu.

Řádky pak znázorňují jednotlivé SPARQL dotazy (u posledních dvou sloupečků tedy ručně převedenou a optimalizovanou SQL podobu dotazu).

5.2.2 Virtuoso

Pro srovnání různých způsobů práce s daty pomocí Virtuosa slouží graf na obrázku 5.1



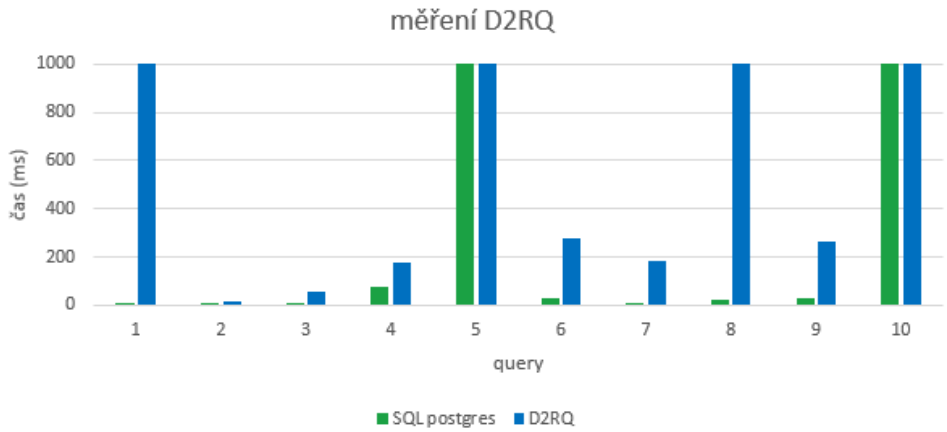
Obrázek 5.1: Výsledky měření Virtuoso

Kromě query 9 a query 10 doběhne mapované virtuoso vždy do jedné sekundy, a ve většině případů i výrazně rychleji nežli virtuoso s nativními RDF daty. Query 9 i 10 jsou obtížnější na vyhodnocení v SQL, jak je vidět z tabulky 5.1 (i optimalizované SQL dotazy trvají dlouho).

Zatímco ručně optimalizovaná podoba query 9 je oproti automaticky vygenerovanému dotazu cca 5x rychlejší, u query 10 byla ručně zoptimalizovaná varianta dokonce 480x rychlejší než překlad vytvořený mapováním (ručně jde díky popisovaným optimalizacím snížit počet potřebných join operací na pouze 2).

U nativního virtuosa s RDF je pak v tabulce 5.1 patrné zrychlení u všech dotazů kromě query 9 jen tím, že se ze SPARQL dotazu odeberou FROM klauzule, které nemají vliv na množinu výsledků. FROM klauzule určují grafy, ze kterých se data mají brát. Na jednu stranu FROM klauzule omezí počet záznamů, se kterými se pracuje, na druhou stranu je ale potřeba u každého záznamu ověřovat, zda spadá do určeného grafu, než se s ním začne pracovat. Proto může dojít i ke zpomalení oproti variantě bez FROM klauzulí.

5.2.3 D2RQ



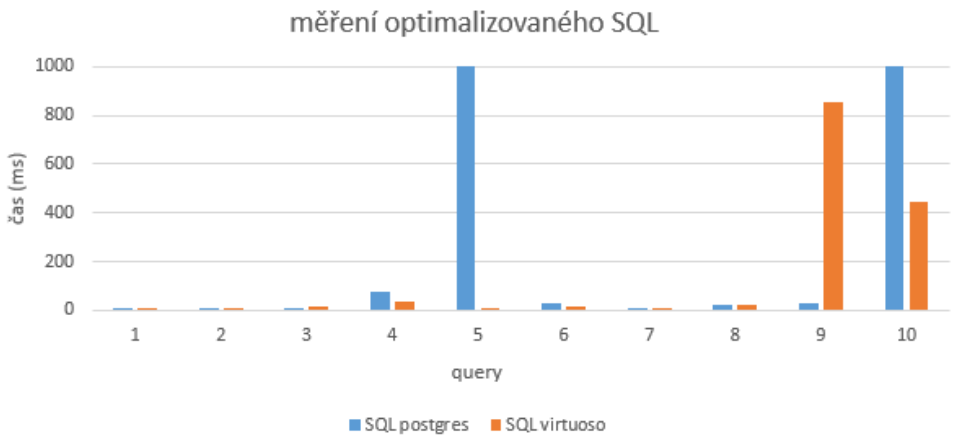
Obrázek 5.2: Výsledky měření D2RQ

na obrázku 5.2 je srovnání D2RQ s ručně vytvořenou verzí dotazů. Čtyři query trvají déle než jednu sekundu. Z grafu i tabulky 5.1 lze pak vyčíst, že u query 1 a 8 se jedná o neefektivní překlad ze SPARQL do SQL, jelikož ručně optimalizovaný SQL dotaz vrátí výsledky mnohem rychleji. Ovšem u Query 5 a 10 trvá déle než jednu sekundu i optimalizovaný ruční překlad, tudíž to není nutně nedostatek mapování.

5.2.4 Ručně optimalizované SQL dotazy

V následujícím grafu 5.3 je patrné s jakou rychlostí lze nalézt výsledky, když položíme dotaz přímo v SQL jazyce. Jelikož jsou zde ručně provedeny všechny optimalizace popisované v předchozí kapitole, nedochází při vyhodnocování dotazů ke zbytečným joinům a selfjoinům. Data se také berou přímo z tabulek, které jsou pro konkrétní dotaz nejvýhodnější.

Naprostá většina dotazů vrátí výsledek do desítek milisekund, pouze postgreSQL přesáhne u dvou dotazů jednu sekundu potřebnou k vyhodnocení. Pomalejší vyhodnocení je do jisté míry způsobené tím, že nasazená verze postgreSQL na serveru nepodporuje paralelní zpracování dotazů, zatímco Virtuoso dokáže využít v některých případech více procesorů naráz.



Obrázek 5.3: Výsledky měření optimalizovaných dotazů v SQL

5.2.5 Celkový přehled

Na posledním grafu 5.4 jsou pro celkové srovnání vykreslena všechna měření dohromady v logaritmickém měřítku. Zelenou barvu zde mají ručně optimalizované dotazy v čistém SQL, a je patrné, že v rychlosti jsou na tom velmi dobře oproti ostatním měřeným způsobům.

V grafu jsou také, stejně jako v tabulce v úvodu, pro zpřehlednění vynechána měření bez FROM klauzule u D2RQ i Virtuosa s mapováním. U Virtuosa s nativními RDF daty jsou ponechány obě varianty pro srovnání způsobeného rozdílu.

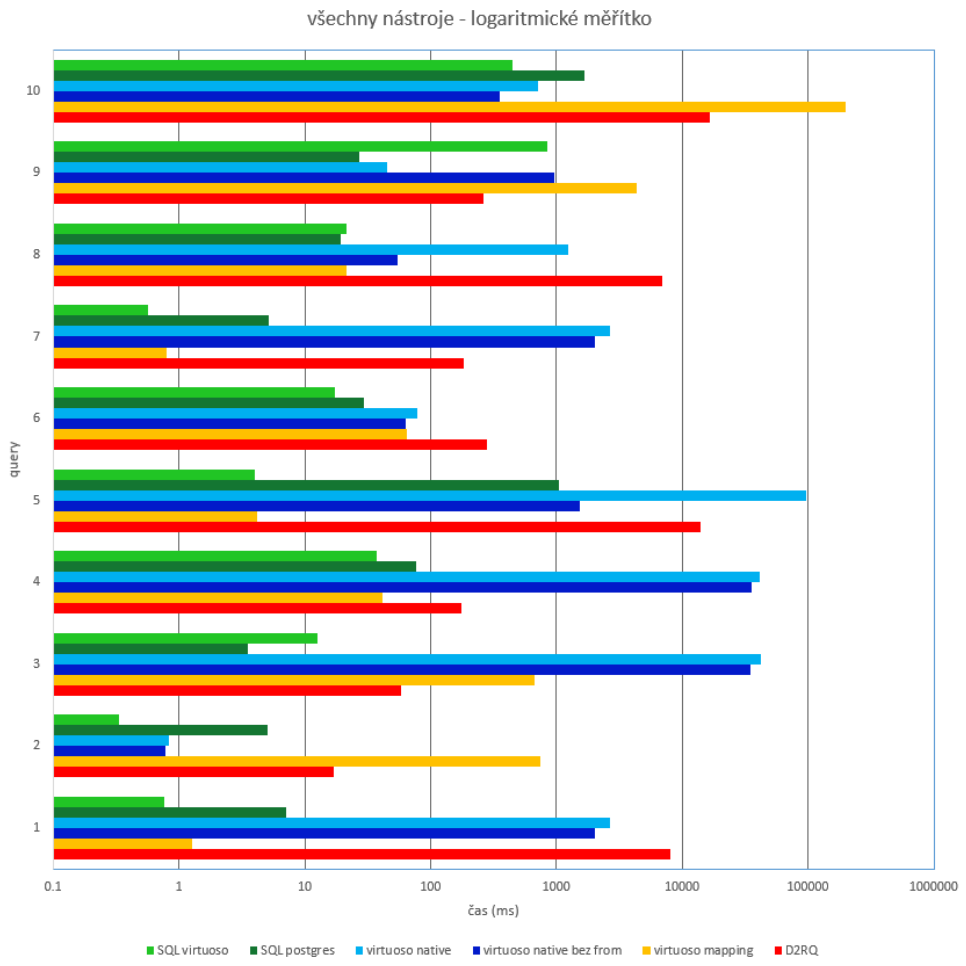
Tabulka 5.2 obsahuje počty join operací v jednotlivých dotazech. Každý přístup generuje SQL dotaz z původního SPARQL dotazu jiným způsobem, tudíž se pro jednotlivé nástroje hodnoty liší. Čím méně join operací je potřeba, tím rychleji zpravidla dotaz proběhne.

Ruční překlad obsahuje nejméně joinů, jelikož bere hodnoty pouze z tabulek, kde najde potřebná data, a vynechává nadbytečné join operace.

Virtuoso s mapováním je zde ve dvou verzích. Nejprve s použitím původních tabulek, a poté s tabulkami endpointů spojených do jedné tabulky, jak je popsáno v kapitole 4.2. V dotazech, kde se pracuje s endpointy je v takto optimalizované verzi méně join operací, než v původním návrhu.

Poslední je D2RQ, které už rovnou využívá novou strukturu tabulek a má tomu rovněž přizpůsobené i mapování.

Všechna měření výkonu proběhla již na takto zoptimalizovaných nástrojích a datech.



Obrázek 5.4: Výsledky měření D2RQ

Tabulka 5.2: Počet joinů získaný různými přístupy k relačním datům. Ruční překlad by měl být optimalizovaný a tudíž mít nejméně joinů

	Počet joinů ve výsledném SQL dotazu			
Query	ruční překlad	Virtuoso (neoptimalizované)	Virtuoso	D2RQ
1	3	5	4	5
2	3	3	3	4
3	2	2	2	5
4	3	3	3	4
5	4	6	5	6
6	3	5	4	5
7	3	5	4	5
8	2	5	4	4
9	5	6	5	6
10	2	4	3	3

5.3 Vyhodnocení

Jak z naměřených výsledků vyplývá, při převodu původních RDF dat do relační podoby lze v mnoha případech dosáhnout značného zrychlení. Největší zrychlení pozorujeme u ručně vytvořených SQL dotazů, jelikož již neobsahují neoptimálnosti způsobené automatickým překladem, a zároveň využívají výhod relační podoby. Navíc u ručního překladu můžeme využít naší znalosti uložení dat, respektive smyslu dotazu, a tím dosáhnout ještě lepšího výsledku.

Ručně překládat každý dotaz uživatele ale v praxi není možné, a ruční překlad zde slouží spíše pro určení spodní hranice časové složitosti, abychom měli srovnání, jak výkonné mapování je, a které dotazy trvají dlouho kvůli mapování, a které kvůli složitosti dotazu.

Měření tak potvrdila, že lze pomocí mapování původních RDF dat na relační databázi dosáhnout značného zrychlení, a tedy je vhodné v praxi využívat tyto mapovací nástroje, a snažit se je co nejvíce zoptimalizovat.

Závěr

Hlavní motivací pro vznik této diplomové práce byla potřeba najít efektivní způsob práce s rozsáhlými daty z projektu PubChemRDF. Tato data budou využívána v rámci výzkumu Ústavu organické chemie a biochemie AV ČR.

Cílem diplomové práce bylo prozkoumat efektivitu vybraných přístupů a technologií umožňujících práci s těmito daty.

Navázal jsem na již existující (ale nikoliv uspokojivé) řešení vytvořené pomocí programu Openlink Virtuoso, které umožňuje práci s nativními RDF daty i s mapováním do relační databáze. Toto řešení jsem rozšířil a provedl v něm několik optimalizací. Virtuoso ovšem v některých situacích není stabilní, a na některé typy dotazů vrací špatné výsledky.

Dále jsem prozkoumal alternativní programy, které by mohly Virtuoso nahradit. V této oblasti není ale příliš nástrojů řešící danou problematiku, které by byly zároveň vyvinuty do produkční verze. Většina programů zkoušených v této práci je ve fázi (zastaveného) raného vývoje s minimem implementovaných funkcí.

Proto byl nakonec zvolen jako nejvhodnější nástroj D2RQ. Ač má pouze poměrně základní vyjadřovací schopnosti v oblasti mapování, byla zde stále možnost, že bude stabilnější než Virtuoso, nebo bude vhodnější z nějakého jiného důvodu.

Jak se ukázalo, aktuálně vydaná verze D2RQ 0.8.1 nepodporuje téměř žádné automatické optimalizace, a je velmi pomalá. Novější, zatím nevydaná verze 0.8.3 má již možnost optimalizace mapování zapnout, a rychlostí tak může konkurovat Virtuosu s užitím mapování. Tato verze ve vývoji byla použita pro finální testy výkonu. Jako nejvhodnější kandidát na náhradu Virtuosa ale nedisponuje takovými prostředky, aby bylo možné namapovat PubChemRDF data bez úprav v IRI. To znamená, že sice samo o sobě funguje, ale není možné pomocí něj dosáhnout úplné kompatibility (tj. mapovat všechna data do původních IRI).

Z naměřených hodnot v praktické části pak vyplývá, že je převod RDF dat na relační podobu s mapováním efektivní řešení. Pro většinu ukázkových

dotazů vrací relační podoba s využitím mapování výsledky rychleji než nativní RDF forma. Je to tedy patrně vhodnější způsob, jak s takto rozsáhlými daty pracovat.

Budoucí práce

Jak se ukázalo, testované programy nejsou buď vhodné pro nasazení na produkční prostředí, nemají dostatečně robustní mapování, nevyužívají potenciál k optimalizaci, či jsou nestabilní.

V rámci ÚOCHB tedy začal vznikat nový mapovací nástroj, který by měl tyto nedostatky do velké míry řešit. Být schopný automaticky provádět potřebné optimalizace popsané v této práci, být stabilní a také musí jít napojit na ostatní stávající systémy a zachovat tak kompatibilitu.

Acknowledgements

This work was supported by ELIXIR CZ research infrastructure project (MEYS Grant No: LM2015047) including access to computing and storage facilities.

Access to the CERIT-SC computing and storage facilities provided by the CERIT-SC Center, under the programme uvProjects of Large Research, Development, and Innovations Infrastructures (CERIT Scientific Cloud LM2015085), is greatly appreciated.

Literatura

[1] The International System of Units (SI), 8th edition. Organisation Intergouvernementale de la Convention du Mètre, 2006.

[2] Lee, J.-H.; Jeong, S.-K.; Kim, B. C.; Park, K. W.; Dash, A., "Donepezil across the spectrum of Alzheimer's disease: dose optimization and clinical relevance". Acta Neurologica Scandinavica. Str. 259-267. [1.5.2015]

[3] What Does Prostaglandins Do?, Hormone Health Network, 2.2.2017 available at:http://www.hormone.org/hormones-and-health/what-do-hormones-do/prostaglandins

[4] Guidelines for accurate EC50/IC50 estimation., National Center for Biotechnology Information, U.S. National Library of Medicine, 8600 Rockville Pike, Bethesda, MD 20894, 2.2.2017 available at:https://www.ncbi.nlm.nih.gov/pubmed/22328315

[5] Resource Description Framework (RDF), The World Wide Web Consortium (W3C), 2.2.2017 available at:https://www.w3.org/RDF/

[6] Semantic web, The World Wide Web Consortium (W3C), 2.2.2017 available at:https://www.w3.org/standards/semanticweb/

[7] W3C Semantic Web Activity, The World Wide Web Consortium (W3C), 2.2.2017 available at:https://www.w3.org/2001/sw/

[8] Web Ontology Language (OWL), The World Wide Web Consortium (W3C), 2.2.2017 available at:https://www.w3.org/OWL/

[9] SPARQL Query Language for RDF, The World Wide Web Consortium (W3C), 2.2.2017 available at:https://www.w3.org/TR/rdf-sparql-query/

[10] R2RML: RDB to RDF Mapping Language, The World Wide Web Consortium (W3C), 2.2.2017 available at:https://www.w3.org/TR/r2rml/

[11] Terse RDF Triple Language, TTL, The World Wide Web Consortium (W3C), 2.2.2017 available at:https://www.w3.org/TR/turtle/

[12] RDB2RDF W3C Standards and Notes, The World Wide Web Consortium (W3C), 2.2.2017 available at:https://www.w3.org/2001/sw/rdb2rdf/

[13] Relational vs. graph databases: Which to use and when?, 2.2.2017 available at:http://sdtimes.com/guest-view-relational-vs-graph-databases-use/

[14] Relational Database, The Tech Terms Computer Dictionary, 2.2.2017 available at:https://techterms.com/definition/relational_database

[15] PubChem , National Center for Biotechnology Information, U.S. National Library of Medicine, 8600 Rockville Pike, Bethesda, MD 20894, 2.2.2017 available at:https://pubchem.ncbi.nlm.nih.gov/about.html

[16] PubChemRDF project, National Center for Biotechnology Information, U.S. National Library of Medicine, 8600 Rockville Pike, Bethesda, MD 20894, 2.2.2017 available at:https://pubchem.ncbi.nlm.nih.gov/rdf/

[17] PubMed , National Center for Biotechnology Information, U.S. National Library of Medicine, 8600 Rockville Pike, Bethesda, MD 20894, 2.2.2017 available at:https://www.ncbi.nlm.nih.gov/pubmed/

[18] Structure , National Center for Biotechnology Information, U.S. National Library of Medicine, 8600 Rockville Pike, Bethesda, MD 20894, 2.2.2017 available at:https://www.ncbi.nlm.nih.gov/Structure/MMDB/mmdb.shtml

[19] Neo4j News: Relational vs. graph databases: Which to use and when?, 2.2.2017 available at:https://neo4j.com/news/relational-vs-graph-databases/

[20] Virtuoso, OpenLink Software, 2.2.2017 available at:https://www.openlinksw.com/

[21] Jena, Apache , 2.2.2017 available at:https://jena.apache.org/

[22] D2RQ Platform, system for accessing relational databases as virtual, read-only RDF graphs, 2.2.2017 available at:http://d2rq.org/

[23] FOAF Vocabulary Specification 0.99, Namespace Document 14 January 2014 - Paddington Edition, 2.2.2017 available at:http://xmlns.com/foaf/spec/

[24] Exploiting PubChem for Virtual Screening, Xiang-Qun Xie, 2.2.2017 available at:https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3117665/

Příloha **A**

Seznam použitých zkratek

FK cizí klíč v relační tabulce (odkaz na primární klíč jiné tabulky)

NCBI National Center for Biotechnology Information - provozovatel PubChem systému

OWL Web Ontology Language

PK primární klíč v relační tabulce

RDF Resource Description Framework - popisuje data pomocí trojic (tripletů)

RD Relational Database - relační databáze, tabulková struktura

R2RML RDB to RDF Mapping Language - jazyk pro mapování dat z relační do RDF formy

SQL Structured Query Language - jazyk pro získání dat z relačních databází

triple trojice subjekt-predikát-objekt

ÚOCHB Ústav organické chemie a biochemie Akademie Věd ČR

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `executable` .. adresář s javovým testovačem výkonu, použitá mapování a nastavení
- `srcources` ..... zdrojové obrázky a dokumenty
- `text` ..... zdrojová forma práce ve formátu LaTeX i PDF
- `vysledky_mereni` ..... Všechny naměřené hodnoty z praktické části