Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Benchmark RDF úložišť

Plzeň 2013 Martin Kryl

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 8. května 2013

Martin Kryl

Abstract

The goal of this work is to measure the performance of current RDF store systems with respect to import speed and SPARQL query resolve speed. To do that, I choose an existing test suite and run a generic test against the stores. Afterwards, the testing dataset is changed to medical data. I use self-made queries against the stores in that step. Finally, the store performances are compared to see which store is the best candidate to store the medical data in.

Obsah

1 Úvod

Na Katedře informatiky a výpočetní techniky probíhá v současnosti vývoj medicínských aplikací. K reprezentaci medicínských dat se využívá metadatový model RDF. Problémem je značná rozsáhlost takových dat. Samotná doba načítání dat se může pohybovat v řádech hodin a jsou zde již poznat rozdíly mezi jednotlivými úložišti. Některá úložiště ani nemusí být schopna tolik dat pojmout.

Cílem této práce je otestovat výkon současných RDF úložišť při zpracovávání rozsáhlých dat s ohledem na rychlost plnění daty a rychlost vyhledávání.

K tomu je potřeba seznámit se se samotným modelem RDF, zmapovat existující metody testování a porovnávání úložišť a vyhledat výsledky proběhlých měření od jiných autorů. Dále je nutné zvolit vhodnou metodu testování a navrhnout SPARQL dotazy pro konkrétní medicínská data na základě používaných ontologií. Samotné testování je následně provedeno.

Po zpracování získaných údajů, jejich diskuzi a porovnání s výsledky ostatních autorů by mělo být zřejmé, které z testovaných úložišť vyhovuje nejlépe požadavkům.

2 Model RDF

2.1 RDF

Resource Description Framework, dále jen RDF, je rodina specifikací vydaných World Wide Web Consortium (W3C). RDF Primer [18] deklaruje RDF jako jazyk pro reprezentaci informací o webových zdrojích. Původně se mělo jednat hlavně o metadata zdrojů jako jméno autora, název dokumentu nebo informace o autorských právech. Zobecněním konceptu „webového zdroje“ je ale možné využívat RDF i k identifikaci jiných věcí než webových dokumentů.

Informace se zaznamenává jako trojice subject, predicate, object. Subject udává, jakého zdroje se informace týká. Predicate je vlastnost, která je trojicí definována. Object je hodnotou vlastnosti, kterou subject nabývá. Soubor RDF trojic představuje pojmenovaný orientovaný graf. Ukázka takové trojice je například ve výpisu 2.1.

```
Tento dokument - Autor - Martin Kryl .
```

Výpis 2.1: Jednoduchá informační trojice

Trojice ale nemá žádnou informační hodnotu. Subject `Tento dokument` je bez kontextu pouze obecným popisem prvku. Jméno autora není bez bližší specifikace rovněž jednoznačné. K dosažení jednoznačnosti se využívá RDF identifikátory Uniform Resource Identifier (URI). Části URI jsou v některých případech substituovány definicí pomocí klíčového slova `prefix`. Trojice ve výpisu 2.2 je ekvivalentní s blokem dat ve výpisu 2.3.

```
<http://example.org/document1> <http://example.org/Autor>
   <http://example.org/person1> .
```

Výpis 2.2: Jednoduchá RDF trojice

```
@prefix ex: <http://example.org/> .
ex:document1 ex:Autor ex:person1 .
```

Výpis 2.3: RDF trojice s použitím substituce

Výhodou RDF je, podobně jako dalších metadatových modelů, že informace takto zapsaná může být zpracována poměrně jednoduše strojem.

2.2 Ontologie

Aby bylo možné informace zpracovávat, je potřeba formálně definovat slovník, logické vztahy mezi termíny a koncepty v popisované sadě informací. Taková formalizace je označována jako ontologie.

Základní slovníky používané při tvorbě ontologií jsou RDF a RDF schéma (RDFS). RDFS je sada tříd sloužících k hierarchickému členění popisovaných objektů. Vrchní třídou je `rdfs:Resource`, což je všechno, co je popisované pomocí RDF. Třída `rdfs:Class` definuje resource jako třídu pro jiné resource. Pomocí vlastnosti `rdfs:subClassOf` lze definovat hierarchickou strukturu mezi třídami a vlastnost `rdf:type` slouží k označení, že daná resource je instance nějaké třídy.

Vrátíme-li se k předchozímu příkladu, můžeme vlastnostmi `rdfs:domain` a `rdfs:range` zadefinovat vlastnost `Autor` tak, že subject je instance třídy `Document` a object instance třídy `Person`. Takové definici odpovídá výpis 2.4.

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ex:   <http://example.org/> .
ex:Autor rdfs:domain ex:Document .
ex:Autor rdfs:range ex:Person .
```

Výpis 2.4: Definice vlastnosti ex:Autor

Software, který zpracovává data z výpisů 2.3 a 2.4 může v důsledku vyvodit, že `person1` je instancí třídy `Person`, aniž by to muselo být v datech explicitně zapsáno. Případně může být takové vyvozování použito k ověření konzistence získaných informací, pokud je v datech informace o `person1` již obsažená. Software schopný tímto způsobem vyvozovat souvislosti se označuje jako sémantický reasoner nebo pouze reasoner.

Web Ontology Language (OWL) je jazyk pro ontologie vytvořený k reprezentování komplexních znalostí a vztahů mezi nimi. Je to nadstavba RDFS umožňující pestřejší výrazy. Lze tak popisovat vztahy mezi třídami, rovnosti dvou prvků nebo definovat omezení u vlastností tříd. Například je možné deklarovat, že prvek třídy A nemůže být prvkem třídy B. Dále lze deklarovat symetričnost a tranzitivitu vlastností.

2.3 Úložiště a práce s daty

K uchování RDF dat je potřeba použít RDF úložiště. Některá úložiště, jako například Mulgara, byla vybudována přímo za účelem uchovávání RDF dat. Jiná byla postavena na enginu již existujících relačních databází.

Úložiště mohou poskytovat persistentní nebo nepersistentní modely pro uchování dat. Persistentní model je takový, které je fyzicky uchováván na pevném disku a po restartování stroje je s ním znovu možné pracovat. Naproti tomu nepersistentní, což je případ in-memory úložišť, je po celou dobu nahraný pouze v paměti RAM. Práce s ním je tak rychlejší, protože není nutné přistupovat na disk, ale svůj obsah udrží pouze omezenou dobu.

K dotazování se nad úložišti, která uchovávají data v RDF formátu, je využíván jazyk SPARQL (SPARQL Protocol and RDF Query Language). Syntaxe dotazovacího jazyka je popsána ve W3C specifikaci [19]. Pro dotazování se používají čtyři různé formy - `SELECT`, `ASK`, `CONSTRUCT` a `DESCRIBE`. V těle dotazu se nachází blok `WHERE`. Ten pomocí trojic popisuje podmínku, která vymezuje dotazované informace.

Dotaz `SELECT` je nejčastěji používanou formou. Vrací data ve formě tabulky, kde sloupce jsou definované proměnnými uvedenými za klíčovým slovem SELECT. Příkladem takového dotazu je dotaz 6 v příloze.

Forma `Construct` získanou informaci převede do RDF grafu. V dotazu 12 je použitá k získání informací o určité nabídce a zapsání dat do vlastního formátu.

Dotazem `DESCRIBE` jsou požadovány veškeré trojice, které popisují určitý prvek. V dotazu 9 jsou tak získávány informace o recenzentovi, který napsal hodnocení produktu na trhu.

Na dotaz s klíčovým slovem `ASK` je odpovězeno logickou hodnotu true nebo false podle toho, jestli je v datasetu soubor trojic popisovaný v dotazu.

Úložiště mohou jako část svého kódu obsahovat službu, která naslouchá na určité adrese a obsluhuje tam příchozí SPARQL dotazy. To se označuje jako SPARQL endpoint.

3 Metody benchmarkingu RDF úložišť

V praxi se používá řada různých metodik testování a porovnávání RDF datových úložišť. Mezi nejvýznamnější z nich se řadí Berlin SPARQL Benchmark (BSBM), Lehigh University Benchmark (LUBM) a DBpedia SPARQL Benchmark (DBPSB).

3.1 Berlin SPARQL Benchmark

Podle [9] byl BSBM navrhnut tak, aby splňoval tři cíle.

1. Benchmark by měl umožnit srovnávání úložišť, která mají SPARQL endpoint, nezávisle na architektuře
2. Benchmark by měl simulovat komerční prostředí. Několik klientů zároveň vytváří zátěž posíláním sady dotazů odpovídajících reálnému chování uživatelů v daném prostředí.
3. Nemělo by být potřeba komplexního vyvozování sémantickým reasonerem. Mělo by jít především o integraci a vizualizaci velkého množství dat z různých zdrojů.

Benchmarková sada obsahuje program na generování dat a program k testování. Generátor umí vygenerovat libovolně velký dataset. Kromě generování dat v RDF formátu je možné vygenerovat stejná data v SQL, tedy relačním modelu.

Testování probíhá nad umělými daty. Data popisují prostředí internetových obchodů se zbožím a webů hodnotících kvalitu produktů. K dotazování je použita posloupnost dotazů, která se snaží vystihnout chování lidí při online nakupování. Benchmark poskytuje dvanáct různých SPARQL dotazů a využívá různé možnosti jazyka jako `CONSTRUCT` a `DESCRIBE` dotazy nebo komplexní seřazení výsledků.

Program na testování posílá posloupnosti SPARQL dotazů na SPARQL endpoint testovaného úložiště. Umožňuje také simulovat přístup několika uživatelů souběžně. Jednotlivé dotazy obsahují parametry, za které jsou dosazovány pseudonáhodné hodnoty z datasetu. Lze generovat stejnou náhodnou posloupnost při použití stejného seedu. Pro testování relačních databází je připravena SQL verze SPARQL dotazů.

Testovací program měří vykonané dotazy za vteřinu pro každý typ dotazu. Dále počet vykonaných sad dotazů za hodinu a celkovou dobu běhu všech dotazů. Dobu načítání dat si musí uživatel změřit sám.

3.2 Lehigh University Benchmark

LUBM je jedním z prvních benchmarků použitých k testování RDF úložišť. V článku [13] popisují autoři, jaké cíle vytvořením benchmarku sledovali.

1. Podpora extenzivních dotazů, tj. dotazů vyžadujících určitý stupeň vyvozování reasoneru, aby dokázal na dotaz správně odpovědět.
2. Libovolná volba velikosti dat - autoři v budoucnu očekávají, že objem dat na sémantickém webu se bude zvětšovat a je tedy potřeba zkoumat, jak si systémy poradí s nárůstem dat.
3. Ontologie střední velikosti a komplexnosti - v té době se především zkoumaly schopnosti reasoneru nad rozsáhlými ontologiemi nebo výkon RDF systémů nad jednoduchými schématy. Nicméně nebyl žádný benchmark, který by šel střední cestou.

Benchmark používá ontologii `Univ-Bench` popisující univerzitní prostředí, jednotlivá oddělení a činnosti tam probíhající. Ontologie je zapsána v OWL Lite a je záměrně vyjádřena tak, aby bylo nutné některé vztahy mezi třídami vyvodit. Podobně jako v BSBM i zde jsou data pseudonáhodně generována speciálním programem.

Jako měřené metriky autoři zvolili dobu načítání a zpracování dat, velikost uložených dat na disku, kombinovanou metriku. Kombinovaná metrika se počítá na základě doby odpovědi na dotaz a kvality odpovědi. Kvalitou odpovědi se rozumí poměr vrácených řádků ku očekávanému počtu řádků v odpovědi.

3.3 DBpedia SPARQL Benchmark

DBPSB je významný tím, že na rozdíl od předchozích metodik využívá k testování reálná data. Ta jsou získána z Dbpedia znalostní báze [1], což je projekt snažící se o extrahování strukturovaného obsahu z internetové encyklopedie Wikipedia. K testování se používají dotazy, které skutečně byly uživateli použity nad danými daty.

4 Testování

4.1 BSBM

V první fázi testování jsem zvolil jeden z existujících benchmarků a provedl testování podle dané metodiky. Volba padla na Berlin SPARQL Benchmark, konkrétně specifikaci verze 3.1 [10].

4.1.1 Volba benchmarku

Hlavním důvodem pro tuto volbu byl především fakt, že benchmark klade důraz na schopnost systému zpracovávat velké množství dat spíše než na schopnost reasoneru vyvozovat ze získaných dat další informace. Jedná se také o jeden z novějších benchmarků, což může znamenat, že výsledky benchmarku mají lepší vypovídající hodnotu o současném stavu používání RDF. Nejnovější verze BSBM byla zveřejněna v roce 2011 [10]. V roce 2012 byl aktualizován DBPSB [14]. Oproti tomu poslední verze LUBM je z roku 2005 [13].

Jako nevýhoda se může jevit skutečnost, že využívá uměle vytvořená data s relativně jednoduchou ontologií. Druhá fáze testování nicméně bude probíhat nad reálnými medicínskými daty, a tak volba umělých dat v tomto kroku umožní pozdější srovnání mezi různými typy dat. BSBM se také jeví jako poměrně populární benchmark a existuje tedy řada výsledků získaných touto metodikou. To opět umožní zajímavé srovnání.

Tato fáze testování probíhala na dvou různých strojích. V tabulce 4.1 je zachycena konfigurace obou strojů. Stroje se liší především v kapacitě paměti RAM a ve výkonu procesoru. Další veličinou, která může na výsledky mít vliv, je rychlost otáček pevného disku.

Úložiště, jenž byla testována, jsou Virtuoso OpenSource, OpenRDF Sesame, Mulgara, Apache Jena TDB, Apache Jena SDB. Jak v případě Sesame, tak i u Jena TDB je testováno in-memory úložiště a nativní verze persistentních úložišť. Jena SDB je testována nad relačními databázemi MySQL a PostgreSQL. Posledním testovaným úložištěm je Oracle databáze.

	Stroj 1	Stroj 2
Procesor	AMD Turion 64 2 x 2GHz	Intel Core i5-2300 4 x 2.80GHz
RAM paměť	2GB	16GB
Hard disk	320GB, 5400 rpm	1TB, 7200 rmp
Operační systém	Debian GNU/Linux 6.0.6 Linux 2.6.32-5-amd64 #1 SMP x86_64 GNU/Linux	Debian GNU/Linux 7.0 Linux 3.2.0-4-amd64 #1 SMP Debian 3.2.41-2 x86_64 GNU/Linux

Tabulka 4.1: Konfigurace obou strojů

4.1.2 Dataset

Autoři BSBM poskytují softwarový balík `BSBM Tools v0.2` na Sourceforge stránce projektu[1]. Pomocí programu `generate`, obsaženého v balíku, lze vygenerovat data k testování. Program generuje data deterministicky, je tedy možné stejný test provádět na více počítačích bez nutnosti přenosu velkého objemu dat mezi stroji. Data byla vygenerována a zapsána ve formátu N-Triples, což je jednoduchý formát pro uchovávání RDF dat. Tento formát neumožňuje zkracování pomocí definování prefixů. Data takto zapsaná jsou tak několikanásobně větší než ekvivalentní data v jiných formátech.

Pro testování byl generován dataset přibližně o padesáti milionech RDF trojic. Data byla rozdělena do souborů po jednom milionu trojic. Přesné počty vygenerovaných prvků je možné vidět v dokumentu `vysledky.ods` na listu `BSBM dataset`. Dokument je v elektronické podobě na přiloženém CD.

Vygenerovaná data jsou strukturovaná takovým způsobem, aby systém zpracovávající data nemusel provádět žádné logické vyvozování. K vygenerování dat byl generátor spuštěn s několika parametry uvedenými ve výpisu 4.5.

```
./generate -fc -pc 142500 -nof 50 -fn data50_
```

Výpis 4.5: Parametry při spuštění generátoru dat

[1] http://sourceforge.net/projects/bsbmtools/

4.1.3 Sada dotazů

BSBM nabízí tři různé sady dotazů, v angličtině označované jako querymix. Sada pro Explore Use Case obsahuje dvanáct typů dotazů, které mají odpovídat chování zákazníka, který hledá vhodný předmět ke koupi. Explore and Update Use Case je nadstavbou předešlého. Do sady jsou přidány i dotazy využívající prvky SPARQL 1.1 Update jazyka. Ty tak simulují změnu datasetu, když se například stáhne nějaká nabídka z trhu. Poslední je Business Intelligence Use Case využívající možnosti jazyku SPARQL 1.1 Working Draft. Většinou jde o analytické otázky nad daným datasetem.

K testování jsem vybral Explore Use Case, neboť se mi jevil jako nejbližší stavu, jak je SPARQL v případě medicínských dat používán. Výhodou této volby je, že využívá pouze základní verzi SPARQL jazyka. Mělo by se tak tedy předejít problému s případnou nedostatečnou implementací prostředků pro práci s novější verzí jazyka v úložištích.

Prototypy všech BSBM dotazů jsou uvedeny v příloze, viz dotazy 1-12. Dotaz 1 reprezentuje chování spotřebitele, který hledá produkty s určitými obecnými charakteristikami. Jde tedy o zákazníka, který by chtěl něco koupit, ale nemá přehled o trhu a tak hledá velmi obecně. Následně dotazem 2 zjišťuje zákazník všechny informace o konkrétním produktu, který našel.

V dotazu 3 je použita negace ve filtru k tomu, aby byly vypsány jen produkty, které nemají určitou vlastnost. Další způsoby, jakým spotřebitel může hledat určitý produkt, jsou vyjádřeny dotazy 4-6.

Když je nalezen vhodný produkt, očekává se, že budou zjišťovány informace o obchodnících nabízející takový produkt. V dotazu 7 se kromě toho hledají recenze spotřebitelů na daný produkt. V dotazu 8 je použit `langMatches` filtr k nalezení pouze anglických recenzí.

Dotaz 9 využívá `DESCRIBE` formu k nalezení veškerých informací o člověku, který napsal konkrétní recenzi na výrobek. Dotazem 10 zákazník hledá stále platnou nabídku od prodejce z USA, který je schopen zboží dodat do tří dnů. Nabídky jsou řazeny podle ceny.

V dotazu 11 zákazník shání veškeré informace o konkrétní nabídce, tj. takové trojice, ve kterých se URI nabídky vyskytuje na pozici subjectu nebo objectu. Dotazem 12 si informace o nalezené nabídce exportuje do svého RDF schématu pomocí `CONSTRUCT` formy dotazu.

4.1.4 Postup testování

Na počítači byly před začátkem testu zastaveny všechny nepotřebné procesy. Po nastartování úložiště byl spuštěn skript na jeho případné vyčištění a následně se začala importovat data do prázdného modelu. Po dokončení nahrávání bylo úložiště zastaveno a znovu spuštěno. V případě in-memory úložišť, u kterých by restartování znamenalo ztrátu načtených dat, se tento krok vynechal.

Programem `testdriver` z BSBM Tools sady bylo následně prováděno dotazování na HTTP SPARQL endpoint úložiště. Dotazování obsahovalo na začátku zahřívací část, z níž se naměřená data nezapočítávají do výsledných metrik. `Testdriver`, který zároveň i generuje dotazy, dostal parametrem pro všechna úložiště seed 808080. To znamená, že odpovědi na dotazy mají být stejné ze všech úložišť. Tato vlastnost byla použita pro kontrolu správnosti vrácených odpovědí, viz kapitola 6.2.3.

Program umožňuje simulovat přístup více uživatelů současně k endpointu. Tato možnost při testování nebyla využita, neboť v současnosti není úložiště s medicínskými daty využíváno paralelně.

4.2 Vlastní testování

V druhé fázi jsem pracoval s medicínskými daty. Pro testování jsem navrhl vlastní sadu SPARQL dotazů. Při jejich tvorbě jsem vycházel z prototypů BSBM dotazů a snažil se je modifikovat k užití nad medicínskými daty tak, aby komplexnost a specifika jednotlivých dotazů zůstala stejná. Z toho vyplývá, že takto vytvořené dotazy pravděpodobně neodpovídají způsobu, jakým je s daty běžně zacházeno.

V této fázi testování nevyužívám HTTP SPARQL endpointy, jako tomu bylo v první fázi. Dotazování provádím přes konzole daných úložišť, případně přes skripty, které jsou za tímto účelem poskytovány.

4.2.1 Dataset

V České republice je využíván Datový standard pro lékařské a laboratorní zprávy, který se označuje jako DASTA. Na základě tohoto standardu byla pro popis dat v RDF formátu vytvořena ontologie DASTA Ontology. Hierarchicky nejvyšší třídou v ontologii je `DASTA`, která popisuje zdrojový nosný soubor formátu DASTA. Pod ní je třída `Patient`, která kromě základních osobních informací má také podtřídu `Clinical Event`. To může být libovolná klinická událost týkající se pacienta, jako například provedené vyšetření, laboratorní zpráva nebo příjem k hospitalizaci.

K popisu medicínských dat slouží dále další tři ontologie. První z nich je DICOM. Ta byla vytvořena na Stanfordské univerzitě. DICOM se používá jako model pro uchování informací o pořízených snímcích z vyšetření. Z DASTA ontologie je odkazováno na DICOM soubor pomocí vlastnosti `hasAttachment`. Každý snímek patří do nějaké `Series`. `Series` je pořízena na konkrétním vybavení laboratoře. `Study` obsahuje několik sérií a je vytvořena při příjmu pacienta.

National Institute of Health Stroke Scale Ontology (NIHSS) slouží k zaznamenání průběhu testování pacientů po mozkové mrtvici. Testování probíhá pozorováním a dotazováním se pacienta. Existují přesné manuály jak tímto způsobem hodnotit. Poslední ontologií je SITS. Tou se popisuje léčba pacientů po mrtvici.

K účelu testování úložišť mi byla poskytnuta část lékařských dat, která byla předem anonymizována. Soubor s daty obsahoval padesát milionů RDF trojic. Nicméně kvůli duplicitním záznamům a několika chybným řádkům se výsledná velikost snížila na 47 540 216 trojic.

4.2.2 Sada dotazů

Jak již bylo zmíněno výše, dotazy by strukturou měly být podobné těm z první fáze testování. Dotazy jsou v elektronické příloze ve složce MREquery na přiloženém CD a dále jsou také vypsány v příloze jako dotazy 13-24. Pořadí dotazů odpovídá BSBM

Dotaz 13 se týká hledání pacientů podle dvou diagnóz. Jako třetí hledaný parametr je minimální počet diagnóz hledaného. Jako dotaz 14 jsem pou-

žil ukázkový dotaz vypisující základní informace o konkrétním pacientovi. Dotazy 15 a 16 opět hledají pacienty podle diagnóz.

V dotazu 17 je hledán snímek podobný zadanému snímku. Tento dotaz je složitější než BSBM varianta, neboť je zde nutné přetypovat hodnoty zapsané ve stringu na integer. Dotaz 18 je vyhledávání v textu lékařských zpráv.

Dotazem 19 je požadován výpis vyšetření, která konkrétní pacient podstoupil od nějakého data. Dotaz 20 vyhledá série, které se týkají konkrétního pacienta. Tento dotaz byl ochuzen o filtr na jazyky, ten se v medicínských datech nepoužívá. Dotaz 21 pomocí `DESCRIBE` vyhledá informace o konkrétní sérii.

Dotaz 22 vybere snímky týkající se pacienta, které byly pořízeny od určitého data, a seřadí je podle pozice, kde byly pořízeny. Dotazem 23 se zjišťují všechny trojice, ve kterých je použito URI konkrétního pacienta. Dotaz 24 je `CONSTRUCT`, kterým se převedou informace o konkrétní studii do jiného RDF schématu.

V dotazech se několikrát vyskytlo porovnávání data interpretovaného jako string. To v důsledku nevadí, pokud je datum zapsán ve formátu `YYMMDD` nebo podobném, protože alfabetické řazení odpovídá řazení věcnému.

4.2.3 Postup testování

K tomuto testování nebylo možné použít program `testdriver` z BSBM, neboť soubor se slovníkem, z kterého program získává hodnoty k dosazení při generování dotazů, je v binární podobě. Změnil jsem tedy postup a do prototypů dotazů dosadil manuálně hodnoty získané prohledáním datasetu. V některých případech jsem záměrně volil neexistující URI. Pro každý dotaz jsem udělal pět variant, celkem tedy šedesát dotazů.

Testování probíhalo pouze na druhém, tedy výkonnějším, stroji. Po načtení dat bylo úložiště restartováno, pokud to architektura umožňovala. Následně proběhlo zpracování všech dotazů. Nebral jsem ohled na pořadí, v jakém se dotazy zpracovávají, protože to by v některých případech znamenalo vypnout dotazovací skript a spustit ho znovu. To by mohlo vést k tomu, že úložiště uvolní svoji cache, resp. uvolní paměť, a při následujícím dotazu by byla zdržena opětovným zahříváním se.

5 Testovaná úložiště

5.1 Virtuoso OpenSource

Virtuoso OpenSource, dále jen Virtuoso, má domovskou stránku projektu na [1]. Jde o open source verzi multiplatformního databázového enginu pod licencí GNU GPLv2. Virtuoso je vybudováno na objektově orientované relační SQL databázi. Postupně byla přidána i podpora pro data v RDF formátu a SPARQL dotazy. Testování proběhlo na dvou různých verzích Virtuosa. Verze 6.1, která je dostupná z Debian repository, byla nainstalována na prvním stroji. Na druhém, výkonnějším, stroji byla použita novější verze 6.1.6.

5.1.1 Instalace verze 6.1

Debian umožňuje použít předpřipravený balík Virtuosa k instalaci. V závislosti na používané verzi Debianovských repositářů - stable nebo testing - jsou k dispozici různé verze Virtuosa. V době psaní této práce byla ve stable právě verze 6.1.

Obnovíme tedy index dostupných balíků a vypíšeme Virtuoso balíky. K instalaci použijeme balík `virtuoso-opensource`, viz výpis 5.6.

```
# apt-get update
# apt-cache search '^virtuoso'
# apt-get install virtuoso-opensource
```

Výpis 5.6: Instalace Virtuosa z repository

Během instalace je vhodné zadat heslo pro administrátorské účty `dba` a `dav`. Účet `dba` ja hlavní administrátor databáze, `dav` je administrátor WebDAV. WebDAV je rozhraní pro správu souborů na webovém serveru. V mém případě došlo k chybě během vytváření účtu. Virtuoso o tom uživatele informuje a administrátorské účty jsou vytvořené s výchozím heslem `dba`.

[1] http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/

5.1.2 Instalace verze 6.1.6

Nejnovější verze nebývá dostupná jako binární soubor a je tedy nutné ji zkompilovat. Zdrojové kódy jsou ke stažení na Sourceforge stránkách projektu[2]. K sestavení je potřeba několik balíků, které lze získat z Debianovského repositáře příkazem ve výpisu 5.7.

```
# apt-get install autoconf automake libtool flex bison gperf
  gawk m4 make openssl libssl-dev
```

Výpis 5.7: Instalace balíků potřebných ke kompilaci

Kompilace probíhá standardním způsobem. V konfiguraci je vhodné nastavit adresář, do kterého se má Virtuoso vytvořit, jako je tomu ve výpisu 5.8. Dále je potřeba přidat parametr pro přejmenování příkazu `isql`, který slouží jako konzole pro ovládání Virtuosa, pokud již takový program v systému existuje.

```
$ ./configure --prefix=/slozka/pro/instalaci
  --program-transform-name="s/isql/isql-v/"
$ make
$ make check
$ make install
```

Výpis 5.8: Nastavení instalačního adresáře, přejmenování isql a instalace

5.1.3 Spuštění

Po dokončení instalace by již měla instance Virtuosa běžet, pokud tomu tak není, je možné ji spustit příkazem na výpisu 5.9.

```
$ cd /etc/virtuoso-opensource-6.1/
$ virtuoso-t -df
```

Výpis 5.9: Spuštění Virtuosa ze standardního adresáře

Příkaz na změnu adresáře ve výpisu 5.9 je nutný, neboť Virtuoso používá adresář, ze kterého je spuštěn, jako místo pro uložení databáze, inicializačního

[2] http://sourceforge.net/projects/virtuoso/files/virtuoso/

souboru a dalších souborů potřebných k běhu. Takto spuštěná instance se zastaví, pokud bude zavřeno okno s terminálem. Pokud má být Virtuoso spuštěno jako daemon, tedy jako proces běžící na pozadí, je potřeba vynechat parametr `-df`.

Pokud nastane chyba se zámkem „ERROR: Unable to create file virtuoso.lck (Permission denied)“, pak uživatel pravděpodobně nemá oprávnění k zápisu do `/etc/virtuoso-opensource-6.1/`. To lze vyřešit rekurzivním přivlastněním adresáře uživateli nebo skupině, pod kterým má být instance vytvářena. V případě deklarování celé skupiny je potřeba přidat ji opět rekurzivně právo k zápisu. Kromě tohoto adresáře je potřeba postup opakovat i na `/var/lib/virtuoso-opensource-6.1/`. Postup je vypsán ve výpisu 5.10. Pokud byl zadán jiný než výchozí adresář při instalaci, neměl by tento problém vyvstat.

```
# chown -R uzivatel /etc/virtuoso-opensource-6.1/ /var/lib/virt
  uoso-opensource-6.1/
# chgrp -R skupina /etc/virtuoso-opensource-6.1/ /var/lib/virtu
  oso-opensource-6.1/
# chmod g+w virtuoso-opensource-6.1 -R
```

Výpis 5.10: Změna přístupových práv k adresáři s Virtuosem

Během prvního spuštění by se měl nainstalovat balík obsahující Virtuoso Conductor. To je webová služba umožňující správu úložiště. Conductor je dostupný na `http://localhost:8890/conductor/`, pokud již nebylo v konfiguraci změněno číslo dotyčného portu. Virtuoso také poskytuje na portu 1111 SQL listener, ke kterému je možné se připojit nástrojem `isql-v`, umístěným v `/usr/bin/`. Isql využívám v skriptu pro plnění a čištění tohoto úložiště.

Virtuoso poskytuje řadu možností zabezpečení. Jednou z nich je parametr `DirsAllowed` v konfiguračním souboru, který vymezuje adresáře, z kterých se může číst. Pokud se například bude uživatel snažit importovat data ze souboru ležící mimo tuto skupinu, Virtuoso zahlásí chybu a import neproběhne. Je tedy vhodné rozmyslet, do kterých adresářů má mít program přístup, a v konfiguračním souboru je vyjmenovat.

	Stroj 1	Stroj 2
NumberOfBuffers	170000	1360000
MaxDirtyBuffers	130000	1000000
MaxCheckpointRemap	320000	320000

Tabulka 5.1: Změny nastavení v souboru virtuoso.ini

5.1.4 Změny v konfiguraci

V článku [15] na stránce výrobce je několik doporučení, díky nimž lze dosáhnout lepší výkonnosti Virtuosa při práci s velkým objemem RDF dat. V konfiguračním souboru `virtuoso.ini` jsem nastavil doporučované hodnoty velikosti bufferů v závislosti na velikosti RAM a zvětšil jsem hodnotu `MaxCheckpointRemap` na hodnotu odpovídající 10GB velké databázi. V tabulce 5.1 uvádím přehled změněných hodnot.

Dále jsem na prvním stroji změnil systémovou hodnotu `swappiness` v `/etc/sysctl.conf` z 60 na 10. Tento parametr jádra udává, jak moc je preferován swap prostor na disku místo RAM. Vysoká hodnota znamená, že jádro bude mít větší tendenci uvolňovat namapované stránky paměti.

5.2 Mulgara

Mulgara[3] je open source databáze vytvořená za účelem ukládání metadat a práce s nimi. Mulgara není založena na relační databázi. Jedná se o fork Kowari projektu. Celá databáze je psaná v Javě a měla by tak tedy být zajištěna nezávislost na platformě. K běhu programu je potřeba Java verze 1.5 nebo vyšší.

K testování byla použita Mulgara verze 2.1.13.

5.2.1 Instalace

Nejnovější verze je k dispozici na domovské stránce v sekci Download. K účelům testování postačí varianta označená jako „Binary Download“. Archiv

[3] http://www.mulgara.org/

stačí rozbalit a soubory přesunout na vhodné místo.

5.2.2 Spuštění

Mulgara se spouští jako Java aplikace ze souboru .jar, viz výpis 5.11. Standardně používá adresář, ve kterém je umístěna, jako adresář pro ukládání dat. Toto chování lze změnit spouštěním s parametrem `--path umístění`.

```
$ java -Xms512m -Xmx1200m -jar mulgara-2.1.13.jar
```

Výpis 5.11: Spuštění Mulgary s Java heap-space parametry

Mulgara používá při svém běhu několik portů. RMI registr, tj. registr pro Java API umožňující z jednoho virtuálního stroje volat metody objektů jiného virtuálního stroje, je na portu 1099. 8080 je port pro HTTP přístup s právy čtení a zápis. Port 8081 je pro HTTP přístup pouze ke čtení. Pokud je některý z nich již používaný, server se vypne. Pro změnu portů, na kterých Mulgara poslouchá, lze použít spouštěcí parametry `-r číslo_portu`, `-p číslo_portu` a `-u číslo_portu`. Kompletní výpis spouštěcích parametrů je dostupný na stránkách projektu [11].

Kromě serveru je v archivu také soubor `querylang-x.x.x.jar`, který lze použít jako nástroj pro práci s databází z příkazové řádky. Tento nástroj využívám k importu dat do grafů.

5.2.3 Změny v konfiguraci

Jedinou změnou je nastavení Java heap-space parametrů. Pro první stroj volím `-Xms512m -Xmx1200m` a pro druhý `-Xms8g -Xmx15g`. Heap-space parametry určují, jak velkou část paměti si mohou Java aplikace přisvojit.

5.3 OpenRDF Sesame

Sesame[4] je open source framework pro práci s RDF daty vytvořený nizozemskou společností Aduna. Sesame může být použit nad řadou úložných

[4]http://www.openrdf.org/index.jsp

systémů. Mimo jiné umožňuje ukládání dat do relačních databází MySQL a PostgreSQL. Při testování byla použita verze 2.6.10. Pro ukládání byly použity nativní persistentní layout a in-memory verze úložiště.

5.3.1 Instalace

Nejnovější release verze Sesame je ke stažení na SourceForge stránce projektu[5].

Sesame jako server funguje na principu Java Servletů. Servlet je kus Java kódu na serveru, který se spustí a vykoná jako reakce na HTTP POST požadavek. Je tedy nutné nainstalovat službu, která takové servlety bude spravovat. Dle doporučení [3] jsem zvolil Apache Tomcat verze 7.0.35, v té době poslední stable release.

Na oficiálních stránkách Apache Tomcat[6] je ke stažení nejnovější verze softwaru. Pro účely testu postačí „Core“ distribuce.

Tomcat ke správnému fungování svých spouštěcích skriptů potřebuje nastavit několik proměnných prostředí (environment variables). Adresář, do kterého se Tomcat rozbalil, je nutné zapsat do proměnné `CATALINA_HOME`. Kromě této proměnné je dále potřeba mít nastavenou jednu z proměnných `JAVA_HOME` nebo `JRE_HOME`. První z nich ukazuje na umístění Java Development Kit, druhý na umístění Java Runtime Environment. Verze 7 Tomcatu vyžaduje Javu verze 1.6 a vyšší. V obou případech lze nastavit proměnnou příkazem export. Pokud chceme tyto proměnné prostředí využívat dlouhodobě, je vhodné zapsat je do souboru s nastavením interpreta příkazové řádky. V případě používání Bash tedy jako ve výpisu 5.12.

```
$ echo export CATALINA_HOME="/cesta/k/Tomcat" >> ~/.bashrc
$ echo export JAVA_HOME="/usr/lib/jvm/verze_Javy" >> ~/.bashrc
```

Výpis 5.12: Zapsání proměnných prostředí

Po rozbalení Sesame ze složky `war` zkopírujeme oba soubory, tedy Sesame server a Workbench, do `CATALINA_HOME/webapps`. Zde je po spuštění načte a zpracuje Tomcat.

[5] http://sourceforge.net/projects/sesame/files/Sesame%202/
[6] http://tomcat.apache.org/download-70.cgi

5.3.2 Spuštění

Nyní zbývá nastavit příslušná práva na Bash skripty v `CATALINA_HOME/bin/`, viz výpis 5.13. Tomcat je možné spustit skriptem `catalina.sh` a příslušným parametrem. Parametr `start` spustí Tomcat na pozadí. Ke spuštění v daném okně konzole slouží parametr `run`. Pro vypnutí běžícího serveru je určen parametr `stop` (případně s dalším parametrem `-force` k vynucení vypnutí).

```
$ chmod ug+x $CATALINA_HOME/bin/*.sh
$ $CATALINA_HOME/bin/catalina.sh start
```

Výpis 5.13: Spuštění Tomcat serveru

Tomcat standardně využívá pro komunikaci port 8080. Nastavit jiný port je možné změnou v `CATALINA_HOME/conf/server.xml`. Jde o hodnotu u parametru `Connector port`. Řádka obsahující tuto hodnotu je ve výpisu 5.14 a v nezměněném souboru jde o 70. řádku.

```
<Connector port="8080" protocol="HTTP/1.1"
   connectionTimeout="20000" redirectPort="8443" />
```

Výpis 5.14: Nastavení čísla portu v konfiguračním souboru

Pokud Tomcat server běží, je možné se připojit k Sesame workbench na adrese `http://localhost:8080/openrdf-workbench/`. Workbench nabízí webový interface pro ovládání serveru. K možnosti práce s daty je potřeba vytvořit alespoň jednu repository. Sesame má také konzolovou utilitu k ovládání. Spouštěcí skript `console.sh` se nachází ve složce `/bin` domovského adresáře Sesame.

5.3.3 Změny v konfiguraci

Skript `catalina.sh`, kterým se spouští Apache Tomcat, čte proměnnou prostředí `CATALINA_OPTS` a její hodnotu předává Javě jako spouštěcí argument. Lze tak tedy nastavit parametry ovlivňující velikost heap-space Javy.

Při vytváření repository jsem zvolil indexy spoc, posc, opsc. Heap-space parametry byly stejné jako v případě Mulgary. Tedy `-Xms512m -Xmx1200m` pro první stroj a `-Xms8g -Xmx15g` pro druhý stroj.

5.4 Apache Jena TDB

Apache Jena[7] je Java framework sloužící k budování aplikací sémantického webu licencovaný pod Apache License v2. Mimo jiné nabízí API rozhraní pro práci s RDF daty a úložiště pro jejich uchování. TDB je komponentou Jeny, která se používá pro ukládání RDF dat. Jedná se o netransakční databázi. Pro vytvoření HTTP SPARQL serveru nad tímto úložištěm je použitý Jena Fuseki.

Pro testování používám Jena Fuseki verze 0.2.5 a Jena TDB verze 2.7.4.

5.4.1 Instalace

Aktuální Apache Jena release, stejně jako Jena Fuseki, je k dispozici z oficiálního Apache repository[8]. Adresář, do kterého byl nakopírován obsah archívu `apache-jena-x.x.x.tar.gz`, je vhodné přidat do proměnné prostředí `JENA_HOME`. Adresář s Fuseki do `FUSEKI_HOME`.

Fuseki obsahuje sadu SPARQL Over HTTP (SOH) skriptů, díky nimž lze z příkazové řádky pracovat se SPARQL endpointem. U těchto skriptů a dále skriptu `fuseki-server` je nutné přidat oprávnění ke spuštění jako je tomu ve výpisu 5.15.

```
$ echo export JENA_HOME="/cesta/k/jena" >> ~/.bashrc
$ echo export FUSEKI_HOME="/cesta/k/fuseki" >> ~/.bashrc
$ chmod ug+x $FUSEKI_HOME/fuseki-server $FUSEKI_HOME/s-*
```

Výpis 5.15: Nastavení proměnných prostředí a oprávnění

5.4.2 Spuštění

```
$ cd $FUSEKI_HOME
$ java -Xms512m -Xmx1200m -jar fuseki-server.jar --TYP /NAZEV
```

Výpis 5.16: Spuštění Fuseki serveru

[7] https://jena.apache.org/index.html
[8] http://archive.apache.org/dist/jena/binaries/

Fuseki se spouští jako Java aplikace pomocí `fuseki-server.jar`, viz výpis 5.16. Je potřeba ho spouštět ze složky, ve kterém se aplikace nachází.

Parametr `--TYP` definuje, jaký typ úložiště je požadován. In-memory dataset, který není nikterak uchován na disku a po vypnutí aplikace jsou tedy data ztracena, je vytvořen pomocí `--mem`. Typ `--file=SOUBOR` vytvoří in-memory dataset a načte do něj data ze `SOUBOR`. V případě `--loc=DIR` se použije existující TDB databáze v adresáři `DIR`, nebo se tam čistá databáze vytvoří. Při použití `--desc=SOUBOR` se založí databáze podle popisu v assembler souboru. Poslední variantou je `--config=SOUBOR`, která vytvoří několik datasetů podle definice v konfiguračním souboru.

Parametr `/NAZEV` deklaruje jméno, pod kterým bude dataset dostupný přes HTTP. Jde o povinný údaj. Jedinou výjimkou je případ, kdy se deklaruje z konfiguračního souboru. Autor Fuseki dokumentace varuje před použitím `/dataset` jako parametru pro pojmenování. V takovém případě by mohla být narušena funkčnost webového rozhraní [4].

Fuseki při nezměněném běhu poslouchá na portu 3030. To lze změnit přidáním `--port=CISLO` do argumentů. Pokud mají být zpracovávány i update SPARQL příkazy je nutné spouštět s `--update` parametrem.

5.4.3 Změny v konfiguraci

Fuseki spouštím s Java heap-space parametry `-Xms512m -Xmx1200m`, resp. `-Xms8g -Xmx15g`. Tyto parametry jsou také změněné ve skriptu pro naplnění TDB úložiště, `JENA_HOME/bin/tdbloader` na řádku 24.

V konfiguračním souboru `FUSEKI_HOME/log4j.properties` je snížená úroveň logování jako je tomu ve výpisu 5.17. Při zachování původní hodnoty `INFO` se vypisuje každý příchozí požadavek a odchozí odpověď do konzole, což by ve výsledku mohlo zkreslovat naměřená data.

```
log4j.logger.org.apache.jena.fuseki.Fuseki=WARN
```

Výpis 5.17: Změna úrovně logování Fuseki serveru

5.5 Apache Jena SDB

Jena SDB je komponenta Jeny sloužící k ukládání RDF dat do relační databáze. Je licencována pod Apache License v2. Pro vytvoření SPARQL webového endpointu je opět použit Jena Fuseki. Ve verzi Fuseki 0.2.5 se mi nepodařilo SDB zprovoznit, proto jsem použil verzi 0.2.4, pod kterou SDB bez problémů běží.

K testování tedy používám Jena Fuseki 0.2.4 a Jena SDB 1.3.5 nad MySQL 5.1.66 v případě prvního stroje. U druhého stroje jsou použity relační databáze MySQL 5.5.30 a PostgreSQL 9.1.9-1.

5.5.1 Instalace

Nejprve je potřeba nainstalovat vhodnou relační databázi, nad kterou SDB poběží. V mém případě se jedná o MySQL a PostgreSQL. Podle dokumentace na oficiálních stránkách [6] je vyžadována instalace MySQL alespoň verze 5.0.22 a PostgreSQL v8.2.

SDB lze stáhnout z oficiální Apache repository[9]. Složku, kam se archiv rozbalí, je vhodné zapsat do proměnné prostředí `SDBROOT`. Po rozbalení je potřeba přidat spouštěcí práva na skripty v `SDBROOT/bin/`. Potřebné příkazy jsou zapsánvy ve výpisu 5.18.

```
$ echo export SDBROOT="/cesta/k/sdb" >> ~/.bashrc
$ chmod ug+x $SDBROOT/bin/*
```

Výpis 5.18: Změna spouštěcích práv a proměnné prostředí

Instalace nižší verze Fuseki probíhá stejně, jako verze vyšší. Ke stažení je opět v repositáři. Pro připojení MySQL a PostgreSQL databází je zapotřebí stáhnout příslušné Java connectory z oficiální stránky Oraclu[10] a stránky PostgreSQL vývojové skupiny[11]. Po stažení a rozbalení stačí překopírovat jar soubory do `SDBROOT/lib/`.

Aby bylo Fuseki schopné se k databázím připojit, je potřeba poskytnout mu informace o spojení. K tomu slouží konfigurační soubory s příponou ttl.

[9] http://archive.apache.org/dist/jena/binaries/
[10] http://dev.mysql.com/downloads/connector/j/
[11] http://jdbc.postgresql.org/download.html

Konfigurační soubory pro připojení k oběma databázím jsou v elektronické příloze práce v `config/mysql.ttl` a `config/pgsql.ttl`. Podle [5] je v případě PostreSQL lepší použít layout2/index než layout2/hash.

5.5.2 Spuštění

Ke spuštění je nutné mít zapnuté MySQL, případně PostreSQL, a založenou databázi, která je definována v konfiguračním souboru. Poté se následujícím příkazem z výpisu 5.19 spustí SDB a Fuseki server.

```
$ cd $FUSEKI_HOME
$ java -Xms512m -Xmx1200m -cp $SDBROOT/lib/*:fuseki-server.jar
    org.apache.jena.fuseki.FusekiCmd --config=mysql.ttl
```

Výpis 5.19: Spuštění Fuseki serveru pro SDB nad MySQL

5.6 Oracle

Oracle databáze je proprietární software společnosti Oracle Corporation. Od verze 11 nabízí podporu pro sémantické technologie pod názvem Oracle Spatial and Graph a RDF Semantic Graph.

Pro plnění databáze a dotazování se nad daty v ní využívám nástroj MetaMed [16] [17], který je vyvíjen zde na Katedře informatiky a výpočetní techniky. Podobně jako u ostatních Java aplikací, i v případě Metamedu nastavuji heap-space parametry na `-Xms8g -Xmx15g`.

6 Výsledky

Data jsou zpracována v souboru `vysledky.ods` na přiloženém CD. Na listu `load` jsou výsledky týkající se načítání dat do úložišť. List `sparql` zobrazuje metriky BSBM a mého benchmarku. Na `BSBM Dataset` je popsán obsah vygenerovaných dat. List `BSBM ověření odpovědí` porovnává počty odpovědí na jednotlivé dotazy. Zbylé listy obsahují naměřená data, která byla zpracována do výstupů na předcházejících listech.

6.1 Nahrávání dat do úložišť

Na prvním testovaném stroji se malé množství paměti RAM ukázalo jako značně limitující faktor. 2 GB RAM nepostačovala k tomu, aby se do in-memory úložišť vešel dataset s 50 miliony trojic. Bylo nicméně zajímavé sledovat, jak si s nedostatkem paměti poradí persistentní úložiště.

Z tabulky 6.1, případně grafu 2 v příloze, je vidět, že nejlépe na prvním stroji dopadla Jena TDB, která pomocí programu `tdbloader` naimportovala a zaindexovala BSBM data za necelé 3 hodiny. V řádech hodin se pohyboval i čas importu do Virtuosa. Zbylá úložiště v tomto ohledu propadla. Sesame se svojí nativní databází se podobně jako Jena SDB nad MySQL plnily několik dnů. Mulgara import dat nezvládla, když při načítání 37. milionu trojic vyhodila chybu a ukončila se. Pokus o pokračování po spuštění úložiště byl neúspěšný.

Ze získaných dat je možné určit trvání importu jednotlivých souborů, tedy milionu trojic. Lze tak sledovat rozdíl mezi dobou načítání prvních trojic, kdy je ještě paměť dostačující k držení dat, a doby načítání posledních trojic, kdy paměť již zcela jistě nestačí. Graf je k nalezení v souboru `vysledky.ods`, případně v příloze jako graf 1. Z grafu je patrné, že zatímco Sesame a Jena nad MySQL se s nedostatkem paměti postupně vyrovnávaly a křivky zpomalily růst, křivka Mulgary rostla stále stejně strmě.

Na druhém stroji probíhalo importování o poznání lépe. Paměť již byla dostatečná k tomu, aby šlo použít Sesame in-memory úložiště, které ve výsledku dokázalo data zpracovat nejrychleji. Pod hodinu se kromě toho dostala i Jena TDB.

	Stroj 1 BSBM	Stroj 2 BSBM	Stroj 2 MRE
Jena TDB	02:55:22	00:56:48	00:23:42
Jena in-memory	error	error	00:09:29
Jena SDB mysql	114:26:01	14:10:28	14:44:52
Jena SDB Postgres	n/a	09:59:39	05:20:09
Mulgara	error	error	02:59:58
Sesame native	122:23:38	129:43:01	110:21:40
Sesame in-memory	error	00:48:17	00:12:47
Virtuoso	05:34:47	01:33:54	01:50:43
Metamed Oracle	n/a	02:58:38	03:33:59

Tabulka 6.1: Doba plnění jednotlivých úložišť, HH:MM:SS

In-memory úložiště od Jeny spuštěné pod Fuseki, pravděpodobně v důsledku špatné práce s pamětí, nedokázalo všechna data zpracovat. Nejvíce se podařilo načíst 16 milionů řádek při použití přepínače Java Garbage Collectoru `-XX:+UseConcMarkSweepGC` a necháním půlminutové prodlevy po načtení každého souboru. Další soubor už načten nebyl a po více než dvou hodinové nečinnosti byl proces ukončen. Při zkoušení jiných přepínačů proces spadl sám s chybou „500 GC overhead limit exceeded“ nebo „500 Java heap space“.

Jediný systém, který si na druhém stroji nepolepšil oproti prvnímu, je Sesame nativní úložiště. Na výkonnějším stroji mu načítání trvalo o dalších 7 hodin více. Částečně to může být důsledkem toho, že načítání bylo pozastaveno a pokračovalo se v něm až po několika dnech.

Za zmínku stojí, že Mulgara opět po 37. milionu načtených trojic vyhodila chybu „AbstractXAResource - Failed to rollback resource in transaction“. Následně se databáze stala nekonzistentní a nešla nikterak obnovit. Pokusy o přeskočení souboru, na kterém se načítání zaseklo, nikam nevedly. Mulgara se zasekla na libovolném následujícím souboru. Dalo by se spekulovat o tom, že jde o nějaký softwarový limit, vzhledem k tomu, že tak činí na dvou různých konfiguracích strojů. To nicméně vyvrací úspěšný pokus o import medicínských dat, kdy se něco přes 49 milionů naimportovalo do úložiště bez chyby.

Při načítání medicínských dat jsem provedl dvě změny. První z nich bylo použití Metamedu k vytvoření in-memory úložiště namísto Fuseki. Data se do tohoto úložiště podařilo načíst a tato varianta se stala nejrychlejší metodou.

Poměrně rychlé bylo načítání dat v podání Sesame in-memory a Jena TDB.

Druhou změnou bylo přepnutí layout2/hash na layout2/index u PostgreSQL. Tato změna se výrazně projevila, když snížila dobu importu medicínských dat na méně než polovinu (z 13 hodin na 5 hodin 20 minut).

Celkově si z pohledu rychlosti načítání dat nejlépe vedla Jena TDB v případě perzistentních úložišť. V In-memory úložištích si nejlépe vedl memory model Metamedu. Jako průměrná úložiště lze považovat Virtuoso a Oracle.

6.2 SPARQL dotazování

V tabulce 6.2 jsou naměřené hodnoty obecné metriky BSBM querymix/hodina (qmph). Na prvním stroji si nejlépe vedlo Virtuoso, které zpracovalo 732 sad dotazů za hodinu (18300 dotazů). Přibližně poloviční rychlostí pracovala Jena TDB (345 qmph). Třetí v pořadí bylo Sesame nativní (125 qmph). Na výkonějším stroji se však již Jena TDB (1890 qmph) dostala před Virtuoso (1733 qmph). Jena SDB stejně jako Sesame nativní a in-memory se nemohou s nejlepšími měřit. Oracle nemohl v této metrice být testován, neboť program Metamed nepodporuje `DESCRIBE` a `CONSTRUCT` dotazy, které jsou součástí mixu dotazů.

	Stroj 1	Stroj 2
Jena TDB	345,2	1892,2
Jena SDB mysql	0,1	0,7
Jena SDB Postgres	n/a	0,7
Sesame native	125,6	23,4
Sesame in-memory	n/a	185,5
Virtuoso	732,5	1733,9

Tabulka 6.2: Rychlost zpracovávání SPARQL dotazů v qmph

Při porovnávání podle jednotlivých dotazů nelze určit jednoznačného vítěze. V tabulce 6.3 je uvedeno, v kolika různých typech dotazů bylo úložiště nejlepší. Na prvním stroji je v součtu o jeden dotaz méně, protože byl použit mix dotazů, ze kterého byl vyjmut dotaz 6.

Z takového srovnání vychází nejlépe Virtuoso v dotazech nad medicínskými daty a Sesame in-memory v dotazech nad BSBM daty. Za zmínku

stojí, že Jena TDB jako vítěz v qmps metrice je nejrychlejší pouze v jednom typu BSBM dotazů. Příčinou toho je fakt, že jak Virtuoso, tak Sesame in-memory mají značné problémy s dotazem číslo 6. Sesame se dále velmi zdržuje na dotazu 5.

	Stroj 1 BSBM	Stroj 2 BSBM	Stroj 2 MRE
Jena TDB	0	1	2
Jena in-memory	n/a	n/a	1
Jena SDB mysql	1	0	0
Jena SDB Postgres	n/a	0	0
Mulgara	n/a	n/a	0
Sesame native	1	0	1
Sesame in-memory	n/a	6	2
Virtuoso	9	2	5
Metamed Oracle	n/a	3	1

Tabulka 6.3: Počet nejlepších časů při porovnávání jednotlivých typů dotazů

Relační databáze pod Jena SDB v této části testování propadly. Ani v jednom typu dotazů z obou testování se nepřiblížily nejlepším časům.

Aby bylo možné porovnat výsledky s Oraclem, sestavil jsem sady dotazů neobsahující nic, s čím by neuměl pracovat. Výsledky jsou zachyceny v tabulce 6.4. Hodnoty jsou porovnatelné pouze v rámci svého sloupce. Oracle (11073 qmph) je jasně nejlepší v prvním z ukazatelů, když pracuje asi šestkrát rychleji než Jena TDB (1948 qmph). Třetí Virtuoso má 1774 qmph. V druhé metrice je nejrychlejší Virtuoso s 6955 qmph a těsně za ním Sesame in-memory. Oracle je až třetí s rychlostí 4722 qmph.

V druhé metrice se vyskytuje řada zajímavých výsledků. Jena TDB skončila předposlední a SDB nad relačními databázemi se výrazně zrychlily (1/6 resp. 1/12 rychlosti nejlepšího). Tyto výsledky mohou být způsobené malým množstvím dotazů, které na úložiště směřovaly. V takových podmínkách se úložiště nemusela stihnout zahřát.

Obecně největší problémy dělaly dotazy číslo 17 a 22 z mé sady dotazů a číslo 5 z BSBM. V případě dotazů 5 a 17 je pravděpodobně problém v kombinací velkého množství kandidátů na výběr a matematického filtru, pomocí kterého se má redukovat počet kandidátů. V mém případě se navíc jedná o přetypování stringu na integer. Poměrně dobře si v dotazech 5, 17 a 22 vedla Jena TDB.

	BSBM bez 9, 12	MRE bez 17, 21, 24
Jena TDB	1948	571
Jena in-memory	n/a	4534
Jena SDB mysql	1	1110
Jena SDB Postgres	1	442
Mulgara	n/a	2193
Sesame native	24	3966
Sesame in-memory	188	6947
Virtuoso	1774	6955
Metamed Oracle	11073	4722

Tabulka 6.4: Metrika qmph pro speciální sady dotazů

Z pohledu práce ze SPARQL dotazy je obtížné určit nejlepšího. Virtuoso, Oracle, Jena TDB i Sesame in-memory mají určité slabé a silné stránky. Bylo by potřeba provést detailnější analýzu k určení vítěze.

6.2.1 Problémy s Mulgarou

V této části testování se vyskytlo hned několik problémů a většina z nich se týkala úložiště Mulgara. Prvním z nich byl problém v dotazu 10, po kterém Mulgara vrátila chybovou hlášku „Unable to support arbitrarily complex ORDER BY clauses“. Syntaxe `ORDER BY xsd:double(str(?price))`, která chtěla, aby byla hodnota ceny interpretována jako číslo a nikoliv jako string, dělala problém. Zjednodušení podmínky na `ORDER BY ?price` problém vyřešilo.

Vývojáři na wiki stránkách projektu [12] kromě problémů s ORDER BY zmiňují i problém s tím, že Mulgara vrací vždy jen DISTINCT výsledky, i když by SPARQL mohl vracet duplicitní řádky. S tímto problémem jsem se ale nesetkal.

Další problémem byl dotaz 5 z BSBM sady. Po odeslání tohoto dotazu chvíli nebyla vidět žádná aktivita databáze. Po asi 30 vteřinách Mulgara zablokovala connector, přes který byl dotaz poslán. Nepodařilo se mi najít řešení ani vhodnou úpravu dotazu.

Mulgara také potřebuje přímo v dotazu mít položku FROM specifikující RMI grafu, ze kterého má získávat data. Proto byl do každého dotazu přidán řádek `FROM <rmi://localhost/server1#test>`.

Posledním problémem Mulgary bylo, že po určitém počtu dotazů (v průběhu 127. sady, tj. asi 2900 dotazů) vyhodila chybu `java.io.IOException: Too many open files` a ukončila se.

Tyto chyby byly objeveny při testování nad redukovaným BSBM datasetem. Výsledky z něj však neuvádím, protože nemají pro účel srovnávání vypovídající hodnotu.

Posledním problémem Mulgary, resp. `querylang` Java aplikace sloužící jako konzole, je nefunkčnost operátoru `set time on;`. Po nastavení by konzole měla časovat jednotlivé příkazy a informovat o době trvání uživatele. Po zadání příkazu se objeví potvrzovací zpráva, ale časování chybí. K měření doby odpovědi bylo tedy využito webové rozhraní, které časovací informaci poskytuje.

6.2.2 Ostatní problémy

Dalším úložištěm, které mělo problémy s některými z dotazů, byla Jena nad MySQL. Konkrétně se jedná o dotazy 1, 3, 4 a 5 BSBM sady. Odpovědi, i když správné, byly vráceny po extrémně dlouhé době. Řádově se jedné o hodiny. Proto byly tyto dotazy vypuštěny ze sady dotazů pro MySQL.

Problém měl také Metamed, který zatím nemá implementovanou funkčnost pro práci s `DESCRIBE` a `CONSTRUCT` dotazy. Dále si neumí poradit s dotazem 29 (prototyp 17 v příloze) z mé sady dotazů. Po dlouhé době zpracovávání požadavku vrátí Java chybu `java.lang.OutOfMemoryError: Java heap space`.

6.2.3 Ověření správnosti výsledků

Sada BSBM Tools obsahuje program `qualification`, který by měl být schopen porovnat obdržené odpovědi na dotazy s očekávanými odpověďmi. Soubor se správnými odpověďmi nebo program na jeho generaci se mi nepovedl najít a tak jsem přistoupil k vlastnímu způsobu porovnání kvality výsledků.

K ověření správnosti byl použit ukazatel průměrného počtu výsledků. Jelikož dataset i sady dotazů byly generovány deterministicky, je nutnou podmínkou k správnosti to, aby se hodnoty nad stejně velkými datasety a ze stejného množství dotazů rovnaly. Tento způsob kontroly nemůže odhalit drobnou chybu, která se vyskytne v jednom z několika stovek dotazů. Na detekci takových chyb bohužel nemám spolehlivý nástroj a tak nelze s jistotou tvrdit, že všechny odpovědi jsou správné.

Za povšimnutí stojí řádek s dotazy 9 a 12. Nejedná se zde o `SELECT` dotazy, ale o `DESCRIBE` resp. `CONSTRUCT`. BSBM Testdriver u nich měří počet obdržených bytů. Je evidentní, že každé úložiště používá odlišný formát odpovědi. Dále je nutné upozornit, že je jeden rozdíl v sadě dotazů pro první a druhý stroj. Ve druhém byl jeden dotaz 7 nahrazen dotazem 6. To znamená, že v šestém řádku nebudou nulové počty odpovědí a zároveň na sedmém řádku se bude lišit počet odpovědí.

6.3 Porovnání s ostatními výsledky

O problému s úložištěm Sesame a jeho neškálovatelností se již delší dobu ví. Již v [13] se na problém poukázalo. Autoři viděli příčinu především ve špatné správě přidělování ID jednotlivým resource a také ve způsobu jakým Sesame vyvozuje nové trojice.

Podíváme-li se na novější výsledky, vidíme většinou vybrané Virtuoso, BigOwlim a Jena TDB jako úložiště na testování. Poslední dva BSBM experimenty nejsou výjimkou. V roce 2009 [7] se testovala tato tři úložiště vůči 100M a 200M datasetu na stroji s 8GB RAM. V takových podmínkách by jistě nešlo použít in-memory model. BigOwlim tehdy zazářil v rychlosti načítání dat - za 33 minut načetl 100M dataset a za 78 minut i 200M. Pokud porovnáme tento čas se mnou naměřenými výsledky (nejrychleji 48 minut u Sesame in-memory), pak jasně předčí všechna testovaná úložiště, a to i přesto, že importoval dvojnásobný objem dat. Situaci nicméně mohl zkreslovat fakt, že se používala data v TURTLE formátu. V dotazovací fázi nebylo Bigowlim tak dominantní. S 834 qmph by se řadil pravděpodobně mezi průměrné nebo slabší systémy. V mém testování nad polovičním datasetem byla nejvyšší rychlost 1890 qmph u Jena TDB.

O dva roky později se test opakoval se stejnými datasety, ale na výkonějším stroji [8]. BigOwlim tentokrát zvládl za 17 minut 100M a za 39 minut

200M dataset. Byl ale vidět velký pokrok Virtuosa. Čas importu dokázal snížit na 1:50, resp 4 hodiny. To přibližně odpovídá při uvažování rozdílu v HW mnou naměřeným hodnotám 1:33 nad 50M. Při testování dosahovalo Virtuoso dvakrát většího průtoku než BigOwlim a Jena TDB. Hodnota 7352 qmph se může zdát jako velká oproti výsledkům této práce (1733 qmph Virtuoso), je ale třeba si všimnout, že se nepoužívá 6. dotaz, který je časově velmi náročný (fulltextové vyhledávání).

DBPSB nabízí na stránkách [2] výsledky měření z roku 2012, kdy je porovnáváno množství, kolik dotazů jsou úložiště schopna zpracovat. Porovnávané jsou Virtuoso, Jena TDB, Sesame a BigOwlim. Na grafech je vidět, že s rostoucím datasetem si Virtuoso drží prvenství v této metrice. Opět se zde projevuje špatná škálovatelnost Sesame, které při malých datastech drží krok s Virtuosem, ale s rostoucím objemem dat jde rychle dolů.

7 Závěr

Během práce jsem se seznámil s teorií kolem RDF metadatového modelu. Vyhledal jsem v současné době používané způsoby testování RDF úložišť a z nich vybral Berlin SPARQL Benchmark pro provedení testování na svém a propůjčeném výkonnějším stroji. Testoval jsem řadu různých úložišť.

Prozkoumal jsem ontologie využívané v medicínských datech a na jejich základě jsem vytvořil sadu SPARQL dotazů nad těmito daty. Při tvorbě jsem vycházel ze struktury BSBM dotazů. S novou sadou dotazů jsem pracoval v druhé fázi testování. Několikrát jsem se setkal s nestandardním chováním úložišť a vše jsem se snažil popsat v dokumentu.

Při práci jsem vytvořil několik Bash skriptů k zautomatizování práce s úložišti a zpracování dat z logů. Skripty jsou na přiloženém CD. Naměřené hodnoty jsou zaznamenány v OpenOffice Calc sešitu.

Ve výsledku byla Jena TDB nejrychlejším persistentním RDF úložištěm při načítání dat. Nejlepší úložiště podle výkonu při vyhledávání nebylo ale možné určit. Každé z trojice Virtuoso, Jena TDB a Oracle mělo nějaké slabé a silné stránky.

Benchmark také odhalil, že Jena SDB nad relačními databázemi je nevhodná k použití k práci s velkým objemem dat. Podobně tomu je v případě Sesame, které má problémy se škálovatelností.

Na konci práce jsem porovnal výsledky s několika vybranými benchmarky.

Přehled zkratek

BSBM Berlin SPARQL Benchmark - benchmark používaný v první fázi testování.

DASTA Datový standard pro lékařské a laboratorní zprávy - národní standard.

DBPSB DBpedia SPARQL Benchmark

LUBM Lehigh University Benchmark

OWL Web Ontology Language - jazyk pro tvorbu ontologií.

qmph Querymix per hour - metrika BSBM.

RDF Resource Description Framework - metadatový model, který spravuje World Wide Web Consortium.

RDFS RDF Schema - základní slovník pro tvorbu ontologií.

SOH SPARQL Over HTTP - soubor skriptů k spouštění SPARQL dotazů na HTTP endpoint

SPARQL SPARQL Protocol and RDF Query Language - dotazovací jazyk pro RDF data

URI Uniform Resource Identifier - textový řetězec sloužící k přesnému určení zdroje informace.

W3C World Wide Web Consortium - mezinárodní organizace, která vyvíjí a udržuje standardy pro World Wide Web.

Literatura

[1] *wiki.dbpedia.org : Downloads 38:*, 2012a. Dostupné z: http://wiki.dbpedia.org/Downloads38.

[2] *AKSW : Projects / DBPSB / Results 2012*, 2012b. Dostupné z: http://wiki.aksw.org/Projects/DBPSB/Results2012?v=uss.

[3] *User Guide for Sesame - Chapter 6. Server software installation.* Aduna, 2012. Dostupné z: http://www.openrdf.org/doc/sesame2/users/ch06.html.

[4] *Apache Jena – Fuseki: serving RDF data over HTTP.* The Apache Software Foundation, 2012a. Dostupné z: https://jena.apache.org/documentation/serving_data/#server-uri-scheme.

[5] *Apache Jena - SDB Database Notes:.* The Apache Software Foundation, 2013. Dostupné z: http://jena.apache.org/documentation/sdb/db_notes.html.

[6] *Apache Jena - SDB Databases Supported.* The Apache Software Foundation, 2012b. Dostupné z: http://jena.apache.org/documentation/sdb/databases_supported.html.

[7] BIZER, C. – SCHULTZ, A. *Berlin SPARQL Benchmark Results - 11/30/2009*, 2009a. Dostupné z: http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/results/V5/index.html.

[8] BIZER, C. – SCHULTZ, A. *Berlin SPARQL Benchmark Results - 11/30/2009*, 2011a. Dostupné z: http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/results/V6/index.html.

[9] BIZER, C. – SCHULTZ, A. The berlin sparql benchmark. *International Journal on Semantic Web and Information Systems (IJSWIS)*. 2009b, 5, 2, s. 1–24.

[10] BIZER, C. – SCHULTZ, A. *Berlin SPARQL Benchmark*, 2011b. Dostupné z: `http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/spec/20110607/`.

[11] GEARON, P. *Mulgara - Args - Mulgara Project*, 2008. Dostupné z: `http://code.mulgara.org/projects/mulgara/wiki/Args`.

[12] GEARON, P. *Mulgara – SPARQLIssues*, 2011. Dostupné z: `http://code.mulgara.org/projects/mulgara/wiki/SPARQLIssues`.

[13] GUO, Y. – PAN, Z. – HEFLIN, J. LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*. 2005, 3, 2, s. 158–182.

[14] MORSEY, M. et al. Usage-Centric Benchmarking of RDF Triple Stores. In *Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, s. 2134–2140, 2012.

[15] *Virtuoso Open-Source Wiki: RDF Performance Tuning*. OpenLink Software, 2009. Dostupné z: `http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VirtRDFPerformanceTuning`.

[16] VCELAK, P. – KLECKOVA, J. Extrakce metadat z medicínských dat (Medical Meta Data Extraction and Manipulation Project), October 2012.

[17] VCELAK, P. et al. MetaMed – Medical Meta Data Extraction and Manipulation Tool Used in the Semantically Interoperable Research Information System. In *Biomedical Engineering and Informatics (BMEI), 2012 5rd International Conference on*, s. 1281–1285, October 2012. ISBN 978-1-4673-1182-3, IEEE Catalog Number: CFP1293D-CDR.

[18] *RDF Primer*. The World Wide Web Consortium, 2004. Dostupné z: `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`.

[19] *SPARQL Query Language for RDF*. The World Wide Web Consortium, 2007. Dostupné z: `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`.

Přílohy

BSBM SPARQL dotazy

Dotaz 1: Hledání produktů s obecnými charakteristikami

```
PREFIX bsbm−inst: <http://www4.wiwiss.fu−berlin.de/bizer/bsbm/v01/instances/>
PREFIX bsbm: <http://www4.wiwiss.fu−berlin.de/bizer/bsbm/v01/vocabulary/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf−schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#>

SELECT DISTINCT ?product ?label
WHERE {
    ?product rdfs:label ?label .
    ?product a %ProductType% .
    ?product bsbm:productFeature %ProductFeature1% .
    ?product bsbm:productFeature %ProductFeature2% .
    ?product bsbm:productPropertyNumeric1 ?value1 .
        FILTER (?value1 > %x%)
        }
ORDER BY ?label
LIMIT 10
```

Dotaz 2: Zjišťování všech informací o konkrétním produktu

```
PREFIX bsbm−inst: <http://www4.wiwiss.fu−berlin.de/bizer/bsbm/v01/instances/>
PREFIX bsbm: <http://www4.wiwiss.fu−berlin.de/bizer/bsbm/v01/vocabulary/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf−schema#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?label ?comment ?producer ?productFeature ?propertyTextual1
  ?propertyTextual2 ?propertyTextual3 ?propertyNumeric1 ?propertyNumeric2
  ?propertyTextual4 ?propertyTextual5 ?propertyNumeric4
WHERE {
    %ProductXYZ% rdfs:label ?label .
    %ProductXYZ% rdfs:comment ?comment .
    %ProductXYZ% bsbm:producer ?p .
    ?p rdfs:label ?producer .
    %ProductXYZ% dc:publisher ?p .
    %ProductXYZ% bsbm:productFeature ?f .
    ?f rdfs:label ?productFeature .
    %ProductXYZ% bsbm:productPropertyTextual1 ?propertyTextual1 .
    %ProductXYZ% bsbm:productPropertyTextual2 ?propertyTextual2 .
    %ProductXYZ% bsbm:productPropertyTextual3 ?propertyTextual3 .
    %ProductXYZ% bsbm:productPropertyNumeric1 ?propertyNumeric1 .
    %ProductXYZ% bsbm:productPropertyNumeric2 ?propertyNumeric2 .
```

```
    OPTIONAL { %ProductXYZ% bsbm:productPropertyTextual4
      ?propertyTextual4 }
    OPTIONAL { %ProductXYZ% bsbm:productPropertyTextual5
      ?propertyTextual5 }
    OPTIONAL { %ProductXYZ% bsbm:productPropertyNumeric4
      ?propertyNumeric4 }
}
```

Dotaz 3: Hledání produktů s určitou vlastností a nemající jinou vlastnost

```
PREFIX bsbm−inst: <http://www4.wiwiss.fu−berlin.de/bizer/bsbm/v01/instances/>
PREFIX bsbm: <http://www4.wiwiss.fu−berlin.de/bizer/bsbm/v01/vocabulary/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf−schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#>

SELECT ?product ?label
WHERE {
    ?product rdfs:label ?label .
    ?product a %ProductType% .
        ?product bsbm:productFeature %ProductFeature1% .
        ?product bsbm:productPropertyNumeric1 ?p1 .
        FILTER ( ?p1 > %x% )
        ?product bsbm:productPropertyNumeric3 ?p3 .
        FILTER (?p3 < %y% )
    OPTIONAL {
        ?product bsbm:productFeature %ProductFeature2% .
        ?product rdfs:label ?testVar }
    FILTER (!bound(?testVar))
}
ORDER BY ?label
LIMIT 10
```

Dotaz 4: Hledání produktů mající jednu ze sad vlastností

```
PREFIX bsbm−inst: <http://www4.wiwiss.fu−berlin.de/bizer/bsbm/v01/instances/>
PREFIX bsbm: <http://www4.wiwiss.fu−berlin.de/bizer/bsbm/v01/vocabulary/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf−schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#>

SELECT DISTINCT ?product ?label ?propertyTextual
WHERE {
    {
       ?product rdfs:label ?label .
       ?product rdf:type %ProductType% .
```

```
        ?product bsbm:productFeature %ProductFeature1% .
            ?product bsbm:productFeature %ProductFeature2% .
        ?product bsbm:productPropertyTextual1 ?propertyTextual .
            ?product bsbm:productPropertyNumeric1 ?p1 .
            FILTER ( ?p1 > %x% )
    } UNION {
        ?product rdfs:label ?label .
        ?product rdf:type %ProductType% .
        ?product bsbm:productFeature %ProductFeature1% .
            ?product bsbm:productFeature %ProductFeature3% .
        ?product bsbm:productPropertyTextual1 ?propertyTextual .
            ?product bsbm:productPropertyNumeric2 ?p2 .
            FILTER ( ?p2> %y% )
    }
}
ORDER BY ?label
OFFSET 5
LIMIT 10
```

Dotaz 5: Hledání produktů podobných jinému konkrétnámu produktu

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bsbm: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/vocabulary/>

SELECT DISTINCT ?product ?productLabel
WHERE {
        ?product rdfs:label ?productLabel .
    FILTER (%ProductXYZ% != ?product)
        %ProductXYZ% bsbm:productFeature ?prodFeature .
        ?product bsbm:productFeature ?prodFeature .
        %ProductXYZ% bsbm:productPropertyNumeric1 ?origProperty1 .
        ?product bsbm:productPropertyNumeric1 ?simProperty1 .
        FILTER (?simProperty1 < (?origProperty1 + 120) &&
          ?simProperty1 > (?origProperty1 - 120))
        %ProductXYZ% bsbm:productPropertyNumeric2 ?origProperty2 .
        ?product bsbm:productPropertyNumeric2 ?simProperty2 .
        FILTER (?simProperty2 < (?origProperty2 + 170) &&
            ?simProperty2 > (?origProperty2 - 170))
}
ORDER BY ?productLabel
LIMIT 5
```

Dotaz 6: Hledání produktu podle slova v jeho popisku

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bsbm: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/vocabulary/>

SELECT ?product ?label
WHERE {
        ?product rdfs:label ?label .
    ?product rdf:type bsbm:Product .
        FILTER regex(?label, "%word1%")
}
```

Dotaz 7: Hledání obchodníků mající konkrétní produkt a recenzí na něj

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rev: <http://purl.org/stuff/rev#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX bsbm: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/vocabulary/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?productLabel ?offer ?price ?vendor ?vendorTitle ?review ?revTitle
       ?reviewer ?revName ?rating1 ?rating2
WHERE {
        %ProductXYZ% rdfs:label ?productLabel .
    OPTIONAL {
        ?offer bsbm:product %ProductXYZ% .
                ?offer bsbm:price ?price .
                ?offer bsbm:vendor ?vendor .
                ?vendor rdfs:label ?vendorTitle .
        ?vendor bsbm:country <http://downlode.org/rdf/iso-3166/countries#DE> .
        ?offer dc:publisher ?vendor .
        ?offer bsbm:validTo ?date .
        FILTER (?date > %currentDate% )
    }
    OPTIONAL {
        ?review bsbm:reviewFor %ProductXYZ% .
        ?review rev:reviewer ?reviewer .
        ?reviewer foaf:name ?revName .
        ?review dc:title ?revTitle .
    OPTIONAL { ?review bsbm:rating1 ?rating1 . }
    OPTIONAL { ?review bsbm:rating2 ?rating2 . }
    }
}
```

Dotaz 8: Hledání pouze anglických recenzí

```
PREFIX bsbm: <http://www4.wiwiss.fu−berlin.de/bizer/bsbm/v01/vocabulary/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX rev: <http://purl.org/stuff/rev#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?title ?text ?reviewDate ?reviewer ?reviewerName ?rating1 ?rating2
  ?rating3 ?rating4
WHERE {
        ?review bsbm:reviewFor %ProductXYZ% .
        ?review dc:title ?title .
        ?review rev:text ?text .
        FILTER langMatches( lang(?text), "EN" )
        ?review bsbm:reviewDate ?reviewDate .
        ?review rev:reviewer ?reviewer .
        ?reviewer foaf:name ?reviewerName .
        OPTIONAL { ?review bsbm:rating1 ?rating1 . }
        OPTIONAL { ?review bsbm:rating2 ?rating2 . }
        OPTIONAL { ?review bsbm:rating3 ?rating3 . }
        OPTIONAL { ?review bsbm:rating4 ?rating4 . }
}
ORDER BY DESC(?reviewDate)
LIMIT 20
```

Dotaz 9: Získání veškerých informací o konkrétním recenzentovi

```
PREFIX rev: <http://purl.org/stuff/rev#>

DESCRIBE ?x
WHERE { %ReviewXYZ% rev:reviewer ?x }
```

Dotaz 10: Hledání stále platné nabídky z USA dodatelné do 3 dnů

```
PREFIX bsbm: <http://www4.wiwiss.fu−berlin.de/bizer/bsbm/v01/vocabulary/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT DISTINCT ?offer ?price
WHERE {
        ?offer bsbm:product %ProductXYZ% .
        ?offer bsbm:vendor ?vendor .
    ?offer dc:publisher ?vendor .
        ?vendor bsbm:country <http://downlode.org/rdf/iso−3166/countries#US> .
        ?offer bsbm:deliveryDays ?deliveryDays .
```

```
        FILTER (?deliveryDays <= 3)
        ?offer bsbm:price ?price .
    ?offer bsbm:validTo ?date .
    FILTER (?date > %currentDate% )
}
ORDER BY ?price
LIMIT 10
```

Dotaz 11: Nalezení veškerých informací o konkrétní nabídce

```
SELECT ?property ?hasValue ?isValueOf
WHERE {
  { %OfferXYZ% ?property ?hasValue }
  UNION
  { ?isValueOf ?property %OfferXYZ% }
}
```

Dotaz 12: Export informací o nabídce do jiného RDF schématu

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rev: <http://purl.org/stuff/rev#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX bsbm: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/vocabulary/>
PREFIX bsbm-export: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/
  vocabulary/export/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

CONSTRUCT { %OfferXYZ% bsbm-export:product ?productURI .
            %OfferXYZ% bsbm-export:productlabel ?productlabel .
            %OfferXYZ% bsbm-export:vendor ?vendorname .
            %OfferXYZ% bsbm-export:vendorhomepage ?vendorhomepage .
            %OfferXYZ% bsbm-export:offerURL ?offerURL .
            %OfferXYZ% bsbm-export:price ?price .
            %OfferXYZ% bsbm-export:deliveryDays ?deliveryDays .
            %OfferXYZ% bsbm-export:validuntil ?validTo }
WHERE { %OfferXYZ% bsbm:product ?productURI .
        ?productURI rdfs:label ?productlabel .
        %OfferXYZ% bsbm:vendor ?vendorURI .
        ?vendorURI rdfs:label ?vendorname .
        ?vendorURI foaf:homepage ?vendorhomepage .
        %OfferXYZ% bsbm:offerWebpage ?offerURL .
        %OfferXYZ% bsbm:price ?price .
        %OfferXYZ% bsbm:deliveryDays ?deliveryDays .
        %OfferXYZ% bsbm:validTo ?validTo }
```

MRE SPARQL dotazy

Dotaz 13: Hledání pacientů s dvěma konkrétními diagnózami

```
PREFIX rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#>
PREFIX ds: <http://mre.kiv.zcu.cz/ontology/2012/01/dasta.owl#>


SELECT DISTINCT ?id ?patient
WHERE {
    ?patient rdf:type ds:Patient .
    ?patient ds:id_pac ?id .

    ?patient ds:hasDiagnosis ?diag1 .
    ?diag1 ds:diag %Diagnoza1% .

    ?patient ds:hasDiagnosis ?diag2 .
    ?diag2 ds:diag %Diagnoza2% .

    ?patient ds:hasDiagnosis ?diag3 .
    ?diag3 ds:poradi ?pocet .
        FILTER (?pocet > %Diagnoz%)
        }
ORDER BY ?id
LIMIT 10
```

Dotaz 14: Zjišťování základních informací o konkrétním pacientu

```
PREFIX ds: <http://mre.kiv.zcu.cz/ontology/2012/01/dasta.owl#>
PREFIX mre: <http://mre.kiv.zcu.cz/>
PREFIX sits: <http://mre.kiv.zcu.cz/ontology/2012/01/sits.owl#>

SELECT ?id ?prijmeni ?sex ?treatmentFile ?id2 ?birth ?death ?sits
WHERE {
  %Patient% ds:id_pac ?id .
  OPTIONAL {%Patient% ds:jine_idu ?id2 .}
  %Patient% ds:prijmeni ?prijmeni .
  %Patient% ds:sex ?sex .
  %Patient% ds:dat_dn ?birth .
  OPTIONAL {%Patient% ds:dat_de ?death .}
  OPTIONAL {
    %Patient% mre:hasSITS ?sits .
    ?sits sits:id ?treatmentFile .
  }}
```

Dotaz 15: Hledání pacientů s dvěma diagnózami nemající jinou konkrétní diagnózu

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ds: <http://mre.kiv.zcu.cz/ontology/2012/01/dasta.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>


SELECT DISTINCT ?id ?patient
WHERE {
    ?patient rdf:type ds:Patient .
    ?patient ds:id_pac ?id .

    ?patient ds:hasDiagnosis ?diag1 .
    ?diag1 ds:diag %Diagnoza1% .
    ?diag1 ds:dat_du ?date1 .
        FILTER ( ?date1 > %Datum1% )

    ?patient ds:hasDiagnosis ?diag2 .
    ?diag2 ds:diag %Diagnoza2% .
    ?diag2 ds:dat_du ?date2 .
        FILTER ( ?date2 > %Datum2% )

    OPTIONAL {
        ?patient ds:hasDiagnosis ?diag3 .
        ?diag3 ds:diag %Diagnoza3% .
        ?patient ds:id_pac ?testVar }
            FILTER (!bound(?testVar))
}


ORDER BY ?id
LIMIT 10
```

Dotaz 16: Hledání pacientů mající jednu ze dvojic diagnóz

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ds: <http://mre.kiv.zcu.cz/ontology/2012/01/dasta.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT DISTINCT ?id ?patient
WHERE {
    {
    ?patient rdf:type ds:Patient .
    ?patient ds:id_pac ?id .
```

```
    ?patient ds:hasDiagnosis ?diag1 .
    ?diag1 ds:diag %Diagnoza1% .
    ?diag1 ds:dat_du ?date1 .
        FILTER ( ?date1 > %Datum1% )

    ?patient ds:hasDiagnosis ?diag2 .
    ?diag2 ds:diag %Diagnoza2% .
    ?diag2 ds:dat_du ?date2 .
        FILTER ( ?date2 > %Datum2% )
    } UNION {
    ?patient rdf:type ds:Patient .
    ?patient ds:id_pac ?id .

    ?patient ds:hasDiagnosis ?diag3 .
    ?diag3 ds:diag %Diagnoza1% .
    ?diag3 ds:dat_du ?date1 .
        FILTER ( ?date3 > %Datum1% )

    ?patient ds:hasDiagnosis ?diag4 .
    ?diag4 ds:diag %Diagnoza3% .
    ?diag4 ds:dat_du ?date4 .
        FILTER ( ?date4 > %Datum3% )
    }
}
ORDER BY ?id
OFFSET 1
LIMIT 10
```

Dotaz 17: Hledání CT snímku podobného zadanému snímku podle velikosti souboru a exposure

```
PREFIX nfo: <http://www.semanticdesktop.org/ontologies/2007/03/22/nfo/
  nfo_data.rdfs#>
PREFIX ds: <http://mre.kiv.zcu.cz/ontology/2012/01/dasta.owl#>
PREFIX do: <http://mre.kiv.zcu.cz/ontology/2012/01/dicom.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>


SELECT DISTINCT ?image ?hashValue
WHERE {
        ?image ds:hasHash ?hash .
        ?hash nfo:hashValue ?hashValue
    FILTER (%ImageURI% != ?image)
```

```
        %ImageURI% do:Image_Type ?typ .
        ?image do:Image_Type ?typ .
        %ImageURI% nfo:fileLength ?origValue1 .
        ?image nfo:fileLength ?simValue1 .
        FILTER (xsd:integer(?simValue1) < (xsd:integer(?origValue1) + 500)
          && xsd:integer(?simValue1) > (xsd:integer(?origValue1) − 500))
        %ImageURI% do:Exposure ?origValue2 .
        ?image do:Exposure ?simValue2 .
        FILTER (xsd:integer(?simValue2) < (xsd:integer(?origValue2) + 50)
          && xsd:integer(?simValue2) > (xsd:integer(?origValue2) − 50))
}
ORDER BY ?image
LIMIT 5
```

Dotaz 18: Vyhledávání v textu lékařských zpráv

```
PREFIX rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#>
PREFIX ds: <http://mre.kiv.zcu.cz/ontology/2012/01/dasta.owl#>

SELECT ?id ?text
WHERE {
    ?patient rdf:type ds:Patient .
    ?patient ds:id_pac ?id .
    ?patient ds:hasClinicalEvent ?event .
    ?event ds:ptext ?text .
    FILTER regex(?text, "%word%")
}
```

Dotaz 19: Hledání vyšetřeních podstoupených pacientem od určitého data

```
PREFIX ds: <http://mre.kiv.zcu.cz/ontology/2012/01/dasta.owl#>
PREFIX do: <http://mre.kiv.zcu.cz/ontology/2012/01/dicom.owl#>

SELECT ?oznaceni ?nazev ?nazevOddeleni ?datum ?vek ?vaha ?popis ?provedl
  ?kodDiag ?text
WHERE {
        %Patient% ds:hasClinicalEvent ?event .
        ?event ds:oznaceni_o ?oznaceni .
        ?event ds:nazev ?nazev .
        ?event ds:hasOriginatorDepartment ?oddeleni .
    OPTIONAL {
        ?oddeleni ds:jmeno ?nazevOddeleni .
    }
        ?event ds:ptext ?text .
```

```
    OPTIONAL {
        ?event ds:has_study ?studie .
        ?studie do:Patient_s_Age ?vek .
        OPTIONAL {
            ?studie do:Patient_s_Weight ?vaha .
        }
        ?studie do:Study_Description ?popis .
        ?studie do:Referring_Physician_s_Name ?provedl .
        ?studie do:Admitting_Diagnoses_Description ?kodDiag .
        ?studie do:Study_Date ?datum .
        FILTER (?datum > %Date% )
}}
```

Dotaz 20: Hledání sérií týkajících se pacienta

```
PREFIX do: <http://mre.kiv.zcu.cz/ontology/2012/01/dicom.owl#>
PREFIX ds: <http://mre.kiv.zcu.cz/ontology/2012/01/dasta.owl#>

SELECT ?datum ?serie ?pozice ?cast_tela ?cislo_serie ?equipment ?frame ?popis
  ?protokol
WHERE {
    %Patient% do:has_study ?study .
    ?study do:contains_series ?serie .
    ?serie do:Patient_Position ?pozice .
    ?serie do:Body_Part_Examined ?cast_tela .
    ?serie do:Series_Date ?datum .
    ?serie do:is_created_by ?equipment .
    ?serie do:Series_Number ?cislo_serie .
    ?serie do:is_spatialy_defined_by ?frame .
    ?serie do:Series_Description ?popis .
    ?serie do:Protocol_Name ?protokol .
}
ORDER BY DESC(?datum)
LIMIT 20
```

Dotaz 21: Získání veškerých informací o vybavení na kterém byla vytvořena série

```
PREFIX do: <http://mre.kiv.zcu.cz/ontology/2012/01/dicom.owl#>

DESCRIBE ?x
WHERE { %Serie% do:is_created_by ?x }
```

Dotaz 22: Hledání snímků týkajících se pacienta. Filtr podle data poslední změny souboru

```
PREFIX nfo: <http://www.semanticdesktop.org/ontologies/2007/03/22/nfo/
  nfo_data.rdfs#>
PREFIX ds: <http://mre.kiv.zcu.cz/ontology/2012/01/dasta.owl#>
PREFIX do: <http://mre.kiv.zcu.cz/ontology/2012/01/dicom.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>


SELECT DISTINCT ?fileName ?modified ?location
WHERE {
        %Patient% ds:hasClinicalEvent ?event .
        ?event ds:hasAttachment ?file .
        ?file nfo:fileName ?fileName .
        ?file do:Exposure ?exposure .
        FILTER (xsd:integer(?exposure) <= 150)
        ?file nfo:fileLastModified ?modified .
        FILTER (?modified > %Date%)
        ?file do:Slice_Location ?location .
}
ORDER BY xsd:double(?location)
LIMIT 10
```

Dotaz 23: Nalezení veškerých informací o pacientovi

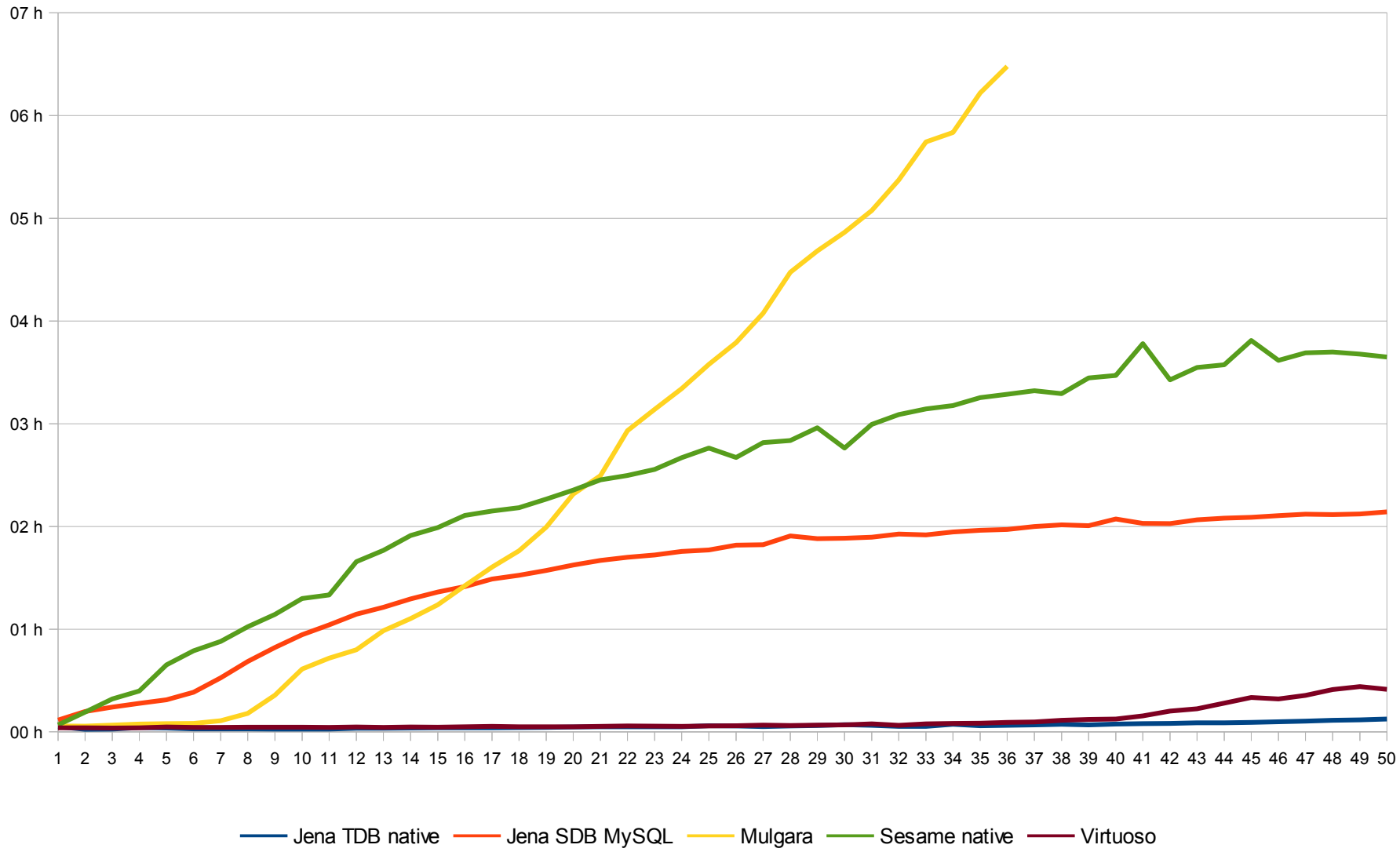
```
SELECT ?property ?hasValue ?isValueOf
WHERE {
  { %Patient% ?property ?hasValue }
  UNION
  { ?isValueOf ?property %Patient% }
}
```

Dotaz 24: Export informací o konkrétní studii do jiného RDF schématu

```
PREFIX ds: <http://mre.kiv.zcu.cz/ontology/2012/01/dasta.owl#>
PREFIX do: <http://mre.kiv.zcu.cz/ontology/2012/01/dicom.owl#>
PREFIX new: <http://new.rdf/schema/export.owl#>

CONSTRUCT{%Studie% new:pactient ?patientURI .
%Studie% new:event ?eventURI .
%Studie% new:nazev ?nazev .
%Studie% new:oddeleni ?oddeleni .
%Studie% new:text ?text .
%Studie% new:provedl ?provedl .
```

```
%Studie% new:datum ?datum .
}

WHERE {
        ?eventURI ds:has_study %Studie% .
        ?patientURI ds:hasClinicalEvent ?eventURI .
        ?eventURI ds:nazev ?nazev .
        ?eventURI ds:hasOriginatorDepartment ?oddeleniURI .
        ?oddeleniURI ds:jmeno ?oddeleni .
        ?event ds:ptext ?text .
        %Studie% do:Referring_Physician_s_Name ?provedl .
        %Studie% do:Study_Date ?datum .

}
```

Graf 1 - doba načítání x-tého milionu trojic včetně indexování na prvním stroji



Graf 2 - Doba importování dat do úložišť