Reprezentácia a spracovanie používateľských preferencií v RDF

Veronika Vaneková

Ústav informatiky, Prírodovedecká fakulta *
Univerzita Pavla Jozefa Šafárika v Košiciach
Jesenná 9, 040 01 Košice, Slovakia
veronika.vanekova@upjs.sk

Abstrakt. Jazyk RDF sa používa na reprezentáciu dát aj metadát rôzneho druhu. Predstavujeme tu OWL schému, ktorá umožňuje ukladať v RDF informácie o používateľoch a ich preferenciách. Preferencie sú vyjadrené pomocou fuzzy množín. Na základe týchto údajov je možné vyhľadať najlepšie objekty z domény pracovných ponúk pre daného používateľa. Popisujeme tiež spôsob, ako doplniť chýbajúce informácie o používateľských preferenciách na základe zhody iných údajov.

Kľúčové slová: model používateľa, fuzzy množiny, preferencie, RDF, ontológie

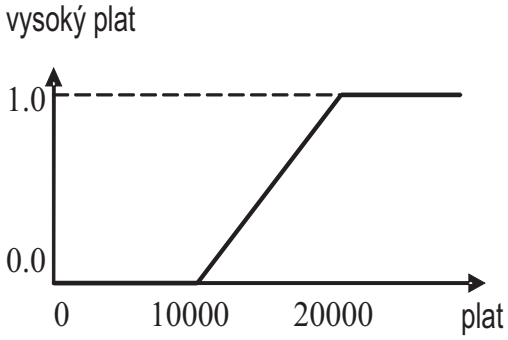
1 Úvod

Mnohé znalostné systémy, ktoré komunikujú priamo s používateľmi, zhromažďujú o nich informácie v určitej forme. Väčšinou ide o záznamy akcií, ktoré používateľ v systéme vykonal, alebo kritériá vyhľadávania, ktoré zadal. Tieto informácie tvoria jeho profil v rámci daného systému.

Pri vyhľadávaní nejakých objektov (napríklad pracovných ponúk na internetovom portáli) používateľ špecifikuje podmienky, ktoré musia výsledné objekty splniť. Napríklad zadá obmedzenie plat $>$ 20000 Sk/mesiac. Podmienky tohto typu však stanovia presnú hranicu a často tak prídeme o výsledky, ktoré sú tesne "pod čiarou" a iba nepodstatne sa odlišujú od výsledkov tesne "nad čiarou". Vhodnejšie je použiť ako kritérium pre vyhľadávanie fuzzy množiny.

Pri "klasických" množinách známych z teórie množín vieme vždy jednoznačne určiť, či do nich nejaký prvok patrí alebo nepatrí. V prípade fuzzy množín však poznáme aj čiastočné členstvo. Ak má charakteristická funkcia fuzzy množiny hodnotu 1, znamená to úplné členstvo prvku v množine, 0 znamená, že prvok do množiny nepatrí a hodnoty medzi 0 a 1 označujú čiastočné členstvo. Ľahko sa vidí, že fuzzy množiny sú zovšeobecnením klasických množín, kde charakteristická funkcia môže nadobúdať iba hodnoty 0 alebo 1, ale nie hodnoty medzi nimi.

* Táto práca je čiastočne podporovaná projektom 'Štátna úloha výskumu a vývoja "Nástroje pre získavanie, organizovanie a udržovanie znalostí v prostredí heterogénnych informačných zdrojov" prierezového štátneho programu "Budovanie informačnej spoločnosti".'



Obrázok 1. Fuzzy množina vysoky_plat

Obrázok 1 znázorňuje fuzzy množinu vysoky_plat. Prvkami množiny sú pracovné ponuky; čím lepšiu hodnotu pre daného používateľa má daný atribút (v tomto prípade plat), tým vyššiu hodnotu bude mať charakteristická funkcia. Táto fuzzy množina definuje používateľské preferencie vzhľadom na ponúkaný plat. Pre každú ponuku tak vieme určiť, nakoľko bude používateľovi vyhovovať.

Niektorí používatelia preferujú vysoké hodnoty (napríklad čím vyšší plat, tým lepšie), v niektorých prípadoch preferujú nízke hodnoty (napríklad absolventi hľadajú takú prácu, kde nepožadujú veľa rokov praxe), prípadne hodnoty zo stredu možného intervalu (napríklad dosiahnuté vzdelanie). Fuzzy množina vždy popisuje iba jednu hľadanú vlastnosť.

Ak chceme pri výbere ponuky zohľadniť viac vlastností, skombinujeme jednotlivé fuzzy hodnoty pomocou agregačnej funkcie a tak získame výslednú hodnotu z intervalu [0,1], ktorá vyjadruje celkovú relevanciu ponuky. Čím bližšie k jednotke je výsledná hodnota, tým vhodnejšia je ponuka. Typickou agregačnou funkciou je napríklad vážený priemer. Iná možnosť je monotónna klasifikácia, ktorá sa dá získať z dát a zapísať vo forme pravidiel.

Ak teda pre konkrétneho používateľa uložíme fuzzy množiny a agregačnú funkciu, dostaneme tak záznam jeho preferencií vzhľadom na atribúty pracovných ponúk. Neskôr už používateľ nemusí zadávať nijaké kritériá a automaticky pre neho dostaneme relevantné výsledky. Samozrejme, tento prístup má význam vtedy, ak pracovné ponuky neostávajú také isté, ale postupne pribúdajú nové a odstraňujú sa staré.

Tento článok popisuje spôsob, akým ukladáme používateľské preferencie a ako s nimi následne pracujeme. Používame tu dáta o pracovných ponukách, ale systém popísaný v tomto článku sa dá analogicky použiť pre ľubovoľné iné dáta, pokiaľ určíme atribúty, ktoré môžu byť pre používateľa zaujímavé. V kapitole 2 sa venujeme štruktúre dát, kapitola 3 je zameraná na prácu s používateľskými preferenciami, ich získanie a spracovanie. Uvádzame tiež testy rýchlosti použitých metód a vlastnosti databázy.

2 Reprezentácia

Pracovné ponuky, ktoré používame, sú uložené v RDF databáze, je teda výhodné rovnakým spôsobom ukladať aj používateľské preferencie. V tejto kapitole stručne uvedieme jazyk RDF a jeho vlastnosti. Potom popíšeme schému, v akej ukladáme pracovné ponuky aj informácie o používateľoch.

RDF (Resource Description Framework) je štruktúra určená na reprezentáciu informácií a metadát na webe (Klyne, Carroll 2004). Je to výrokový jazyk, v ktorom vyjadrujeme logické tvrdenia a výroky pomocou presnej formálnej abecedy. Všetky tvrdenia sú vo forme trojíc:

<subjekt, predikát, objekt>

Takéto tvrdenie nám hovorí, že subjekt má vlastnosť označenú ako predikát a hodnota tejto vlastnosti je objekt. Trojicu môžeme považovať za binárny predikát v logike prvého rádu, teda:

predikát(subjekt,objekt)

Príklad <Osoba1, vzdelanie, "PhD."> Uvedená RDF trojica hovorí o tom, že Osoba1 má vlastnosť vzdelanie s hodnotou "PhD.". Môžeme ju zapísať ako binárny predikát vzdelanie(Osoba1, "PhD.").

Ľubovoľnú znalosť z konkrétnej domény vieme rozložiť na takéto trojice (binárne predikáty). Pomocou RDF teda vieme popísať rôzne znalosti od metadát článkov a www stránok až po rozsiahle dáta v doménových ontológiách. Termín "ontológia" popíšeme v nasledujúcej podkapitole.

2.1 Ontológie

Ak chceme modelovať znalosti z ľubovoľnej domény, oblasti nášho záujmu, musíme najskôr určiť triedy (entity), ktoré doména obsahuje. Podobne ako pri objektovom programovaní, triedy sú usporiadané v stromovej štruktúre vzťahom dedičnosti. Okrem toho triedy majú rôzne vlastnosti a vzťahy ku iným triedam. Toto všetko tvorí štruktúru našich znalostí o doméne. Keď túto štruktúru špecifikujeme formálne, dostaneme doménovú ontológiu.

Ontológia sa dá graficky znázorniť ako graf. Triedy a inštancie, prípadne aj hodnoty jednoduchých dátových typov (literály) sú uzlami grafu. každá trojica potom tvorí jednu hranu v grafe. Napríklad trojica <subjekt, predikát, objekt> znamená, že uzly subjekt a objekt sú spojené hranou označenou názvom predikátu. Hrany sú orientované a smerujú od subjektu k objektu.

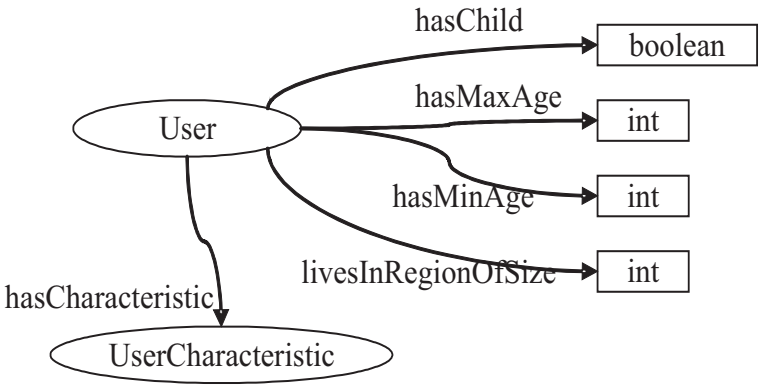
Obrázok 2. Grafické znázornenie jednej RDF trojice.

Trojice môžeme uložiť do databázy a spracovať ich niektorým z dopytovacích jazykov navrhnutých pre RDF. Ako sme uviedli v článku (Vaneková,

Bella, Gurský, Horváth 2005), väčšina RDF dopytovacích jazykov má nedostatky, ktoré bránia ich priamemu použitiu na získanie fuzzy hodnôt. Chýbajú operácie s číslami, agregačné funkcie a triedenie. Tieto funkcie implementujeme pomocou klientského programu v Jave, ktorý komunikuje s RDF databázou a poskytuje pracovné ponuky používateľom.

2.2 Model používateľa

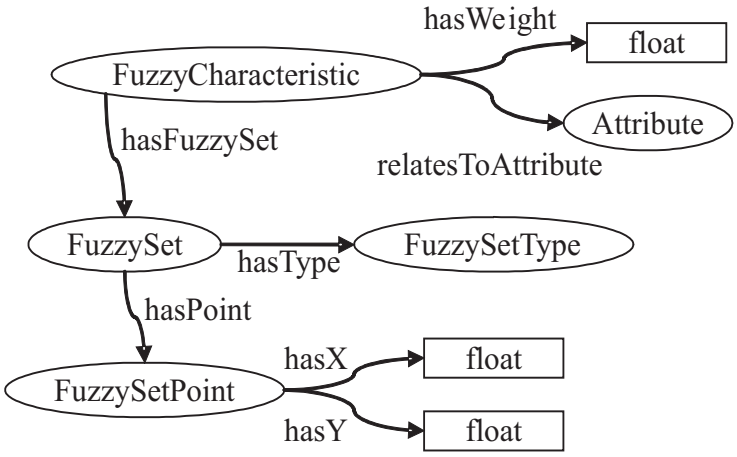
Aj používatelia tvoria samostatnú doménu, ktorú môžeme modelovať. Hlavnou triedou je samozrejme User. Obrázok 3 znázorňuje schému vlastností, ktoré má trieda User. Sú to "hasMaxAge" a "hasMinAge", teda interval, do ktorého patrí vek používateľa; ďalej vlastnosť "livesInRegionOfSize" udáva počet obyvateľov mesta, z ktorého pochádza používateľ; hasChild nadobúda hodnoty true alebo false podľa toho, či používateľ má deti. Okrem týchto vlastností chceme ešte evidovať stupeň vzdelania, rodinný stav, pohlavie, a samozrejme preferencie k jednotlivým atribútom pracovných ponúk. Tieto informácie už nie sú vo forme niektorého z jednoduchých dátových typov (int, boolean, float). Sú to inštancie tried z ontológie a preto ich prepojíme s používateľom cez ďalší uzol typu "UserCharacteristic".



Obrázok 3. Časť schémy ontológie používateľa

Trieda UserCharacteristic má podtriedy FuzzyCharacteristic a GenericUserCharacteristic. Vlastnosť relatesTo spája inštanciu GenericUserCharacteristic s ľubovoľnou inou inštanciou, ktorá sa už nachádza v doménovej ontológii, napríklad so stupňom dosiahnutého vzdelania. Trieda FuzzyCharakteristic slúži na ukladanie fuzzy množín.

Fuzzy množinu charakterizujú jej body, ktoré majú súradnice x a y. Typ fuzzy množiny je jedna zo štyroch možností: rastúca, klesajúca, kopec, dolina. Tieto možnosti popisujú tvar charakteristickej funkcie. Napríklad typ "rastúca"



Obrázok 4. Schéma pre fuzzy množiny

je fuzzy množina, v ktorej majú väčšie prvky hodnotu charakteristickej funkcie bližšiu k 1.

Každá fuzzy charakteristika z ontológie má vlastnosť relatesToAttribute. Vyjadruje to, s akým atribútom pracovnej ponuky súvisí fuzzy množina, napríklad pracovná pozícia, plat.

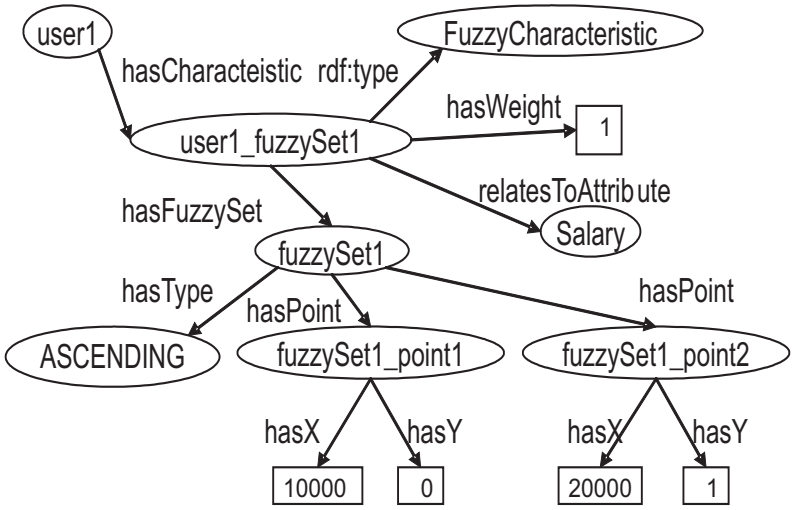
Vzorovú inštanciu používateľa ukazuje obrázok 5. Tento používateľ má definované preferencie vzhľadom na plat pomocou fuzzy množiny fuzzySet1. Fuzzy množinu charakterizujú dva body fuzzySet1_point1 a fuzzySet1_point2 so svojimi x-ovými a y-ovými súradnicami. Medzi týmito bodmi má charakteristická funkcia lineárny priebeh. Zo súradníc bodov je vidieť, že funkcia je rastúca a to isté vyjadruje aj vlastnosť hasType s hodnotou ASCENDING.

3 Spracovanie preferencií z ontológie

Ako sme uviedli v kapitole 2.1, fuzzy množiny nie je možné aplikovať na dáta z domény cez jednoduchý dopyt. Preto ich potrebujeme z RDF databázy načítať do objektov v Jave. Tento postup nezaberie príliš veľa pamäti, lebo na rozdiel od tisícok pracovných ponúk máme v databáze menej ako 10 fuzzy množín pre konkrétneho používateľa.

3.1 Mapovanie na objekty v Jave

Štruktúra objektov v Jave je podobná tej štruktúre, ktorú poznáme z ontológie. Trieda UserProfile má vlastnosti gender, maritalStatus, educationLevel, ktoré majú niekoľko možných hodnôt (napríklad vlastnosť gender môže mať hodnotu MALE alebo FEMALE). Tieto vlastnosti sú typu enum. Podobne to platí pre



Obrázok 5. Používateľ user1 s fuzzy množinou pre plat

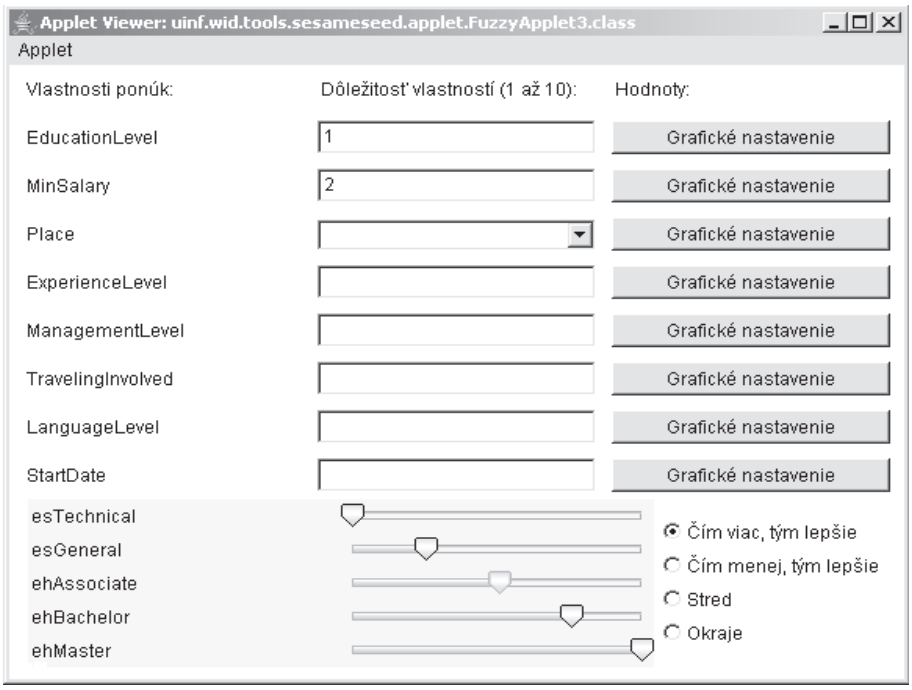
vlastnosť skills, kde však môžeme vybrať ľubovoľnú podmnožinu zručností, ktoré používateľ má. Vlastnosti age a placeOfLiving vyjadrujú vek používateľa a počet obyvateľov jeho bydliska. Číselné hodnoty sme rozdelili do niekoľkých kategórií, a teda sme tiež dostali typ enum s konečným počtom možností. Je tu ešte vlastnosť hasChild typu boolean.

Ako je vidieť, všetky vlastnosti majú konečný počet možných hodnôt a teda v našom modeli sme aj používateľov rozdelili do skupín, napríklad "muži do 25 rokov, ktorí majú vysokoškolské vzdelanie a vodičský preukaz". Predpokladáme, že tieto skupiny používateľov môžu mať rôzne preferencie, ale v rámci jednej skupiny budú ich preferencie podobné.

Databáza Sesame, ktorú používame pre našu ontológiu, obsahuje vlastný dopytovací jazyk SeRQL. Pomocou Sesame API podporuje aj komunikáciu databázového servera s klientským programom. Vytvorili sme teda triedy, ktoré komunikujú s databázou a starajú sa o správu používateľských profilov.

3.2 Získavanie preferencií: priamo alebo z indexu

Od nového používateľa požadujeme dve veci: vyplnenie dotazníka so všetkými vlastnosťami uvedenými v 3.1, a zadanie preferencií. Pre fuzzy množiny máme grafické rozhranie, kde môže používateľ ťahaním posuvníka označiť hodnoty atribútov ako lepšie alebo horšie.



Obrázok 6. Používateľské rozhranie

Napriek tomu, že ovládanie tohto rozhrania je do istej miery intuitívne, môže byť pre niektorých používateľov náročné alebo zdĺhavé. Aj tento problém má riešenie. Pokiaľ používateľ nezadá ručne svoje preferencie, ale vyplní dotazník, vieme určiť, do ktorej skupiny používateľov patrí. V tejto skupine pre každý atribút pracovnej ponuky zistíme, aký typ fuzzy množiny prevláda a rovnaký typ použijeme aj pre nového používateľa. Týmto spôsobom môže ušetriť čas a námahu, ale možno na úkor menšej presnosti výsledkov.

Na takéto priradenie preferencií používame index uložený v súbore. Tento index pre každú skupinu používateľov a každý atribút pracovnej ponuky obsahuje počty fuzzy množín pre 4 základné typy (rastúca, klesajúca, kopec a dolina). Tak vieme určiť, ktorý typ prevláda a použiť ho.

Index je riešený ako hashová tabuľka. Kľúčom je hash vypočítaný z vlastností používateľa, pričom berieme do úvahy iba osobné údaje a nie preferencie. Používatelia rovnakého typu teda budú mať aj rovnaký hash. Pre konkrétneho používateľa a konkrétny atribút dostaneme štyri čísla, ktoré označujú počet fuzzy množín štyroch základných typov. Vieme napríklad zistiť, že používatelia rovnakého typu majú pre atribút "vzdelanie" väčšinou klesajúci typ množiny a aj novému používateľovi priradíme v tomto atribúte klesajúcu fuzzy množinu.

Kedykoľvek nový používateľ zadá svoje preferencie, uložíme ich aj do indexu. Ak však preferencie nezadá ale dostane ich z indexu, neukladáme ich, pretože pre nás neobsahujú nové informácie.

3.3 Testy rýchlosti dopytov

RDF databázy majú iné vlastnosti než klasické relačné databázy, hoci môžu byť na nich založené. V prípade RDF databáz ide o ukladanie a spracovanie trojíc. Na to postačujú tri stĺpce v tabuľke, ale zníži sa rýchlosť zložitejších dopytov, pretože to vyžaduje mnoho operácií typu join.

V našom systéme potrebujeme často komunikovať s databázou a aktualizovať informácie o používateľoch. Dopyty "select" použijeme vtedy, keď chceme z informácií v ontológii zostrojiť objekt (typu User) v Jave a pracovať s ním. Pri registrácii nového používateľa máme k dispozícii objekt a uložíme ho do ontológie pomocou operácie "insert". Pri zmene preferencií zase potrebujeme analógiu "update", ale databáza Sesame takúto operáciu zatiaľ nepodporuje. Zmenu hodnôt v databáze dosiahneme tak, že vymažeme staré hodnoty a vložíme nové.

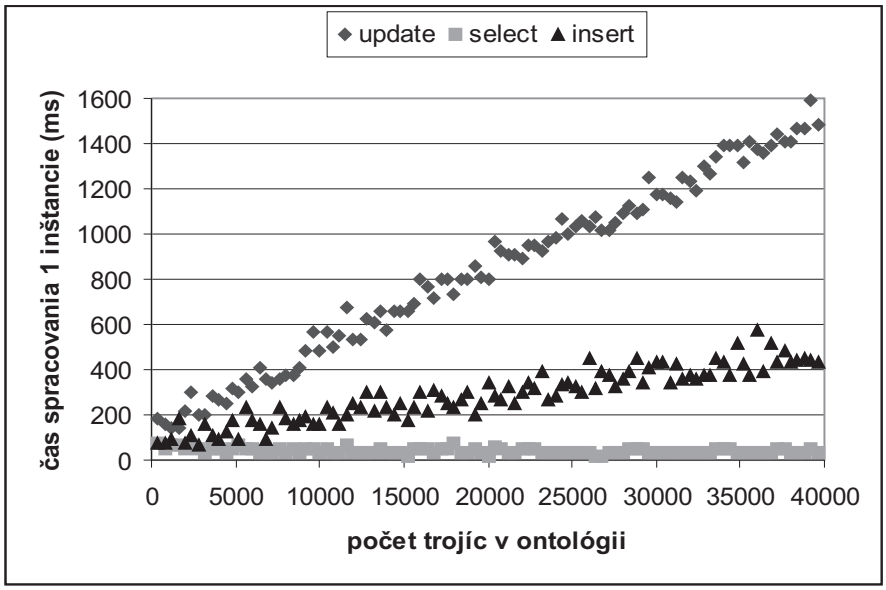
V systéme môže pracovať viac používateľov súčasne. Po uložení objektu používateľa do ontológie dostaneme $1 + a \cdot (7 + 5 \cdot b)$ trojíc, kde a je počet atribútov, b je počet bodov danej fuzzy množiny. V typickom prípade je $a = 7$, $b \in \{2, \ldots, 10\}$, teda inštancia používateľa sa môže skladať až zo 400 trojíc. V tejto kapitole sa pozrieme na to, ako rýchlo databáza Sesame spracuje dopyty, ktoré vyžadujeme.

Sesame podporuje komunikáciu s klientskými programami cez Java API. Máme k dispozícii nasledujúce metódy:

- `performTableQuery(QueryLanguage lenguage, String query);`
  Táto metóda vráti výsledky dopytu vo forme tabuľky. Tabuľkové hodnoty potom spracujeme na objekt v Jave. Pri testovaní programu sa čas dopytu a vytvárania objektu pohyboval okolo 40 milisekúnd a nepresiahol 100 milisekúnd.
- `addData(String data, String prefix, RDFFormat format, boolean checkData, AdminListener listener);`
  Metóda pridá do databázy RDF dáta zakódované v jednom z povolených formátov. V prvej verzii sme pridávali jednotlivé trojice. Takýto prístup je neefektívny, pretože každý prístup do databázy zaberá čas a preto je vhodnejšie obmedziť počet prístupov na minimum. Z každého objektu používateľa sme vytvorili jeden RDF dokument a pridali sme ho do databázy naraz. To znížilo čas operácie približne 5-8 krát.
  Pri testovaní sme ďalej zistili, že rýchlosť insertu lineárne závisí od počtu trojíc, ktoré sa predtým nachádzali v ontológii. Táto vlastnosť negatívne ovplyvní prácu s každou rozsiahlou ontológiou. Je spôsobená tým, že pri vkladaní trojice sa musia aktualizovať vnútorné indexy v RDF databáze.
- `addGraph(QueryLanguage language, String query);`
  `removeGraph(QueryLanguage language, String query);`

Najviac času pri testovaní zabrali operácie "update". Ako sme uviedli vyššie, Sesame takúto operáciu nemá implementovanú a simulovali sme ju pomocou vymazania starých trojíc a následného vloženia nových. Čas vkladania aj mazania trojíc závisí od celkového počtu trojíc v ontológii. Výsledné časy updatu boli dvoj- až štvornásobné oproti časom samostatného insertu za tých istých podmienok.

Obrázok 7 názorne ukazuje výsledky testov. Na osi x je počet trojíc v ontológii, na osi y čas príslušnej operácie s jedným používateľom (čo znamená asi 400 trojíc). Najrýchlejšou operáciou je select, ktorý nezávisí od počtu trojíc v ontológii, alebo závisí len minimálne. Najviac času zaberá update, ktorý zahŕňa aj operáciu insert.



Obrázok 7. Graf času databázových operácií

4 Záver

V tomto článku sme popísali spôsob reprezentácie a spracovania používateľských preferencií pre systém vyhľadávania. Preferencie k jednotlivým atribútom pracovných ponúk ako plat, vzdelanie, pozícia a podobne, reprezentujeme ako fuzzy množiny štyroch základných typov: rastúca, klesajúca, kopec, dolina. Každú

ponuku môžeme ohodnotiť podľa fuzzy množín číslom z intervalu [0,1]. Pri vyhľadávaní podľa viacerých atribútov dostaneme výsledné ohodnotenie po aplikovaní agregačnej funkcie, čo je obyčajne vážený priemer.

Preferencie získame od používateľov cez špeciálne rozhranie. Pokiaľ používateľ nezadá svoje preferencie manuálne, vieme mu priradiť preferencie podľa toho, aký typ množiny prevláda v skupine, do ktorej používateľ patrí. Všetky ručne zadané preferencie sa preto ukladajú do indexu vo forme hashovej tabuľky.

Pri ukladaní používateľských profilov a ich zmien a pri mapovaní ontológie na objekty v Jave využívame databázové operácie ako select, insert a delete. Najrýchlejšou operáciou je select. Čas vykonania insertu a deletu závisí od toho, koľko trojíc ontológia obsahuje. Najpomalšou operáciou je delete a od neho odvodená operácia update.

V budúcnosti chceme okrem rýchlosti dopytov analyzovať aj to, nakoľko vhodné sú defaultne dosadené hodnoty a či sú správne zvolené skupiny používateľov. Tieto testy si vyžadujú množstvo reálnych dát, nasadenie systému do skúšobnej prevádzky a spätnú väzbu od používateľov.

References

1. Gurský, P.: Algoritmy na vyhľadávanie najlepších k objektov bez priameho prístupu. *Proceedings of Znalosti 2006, s. 95-105.* ISBN 80-248-1001-8. `http://klud.ics.upjs.sk/~gursky/papers/2006znalosti.pdf`
2. Klyne, G. – Carroll, J. J.: Resource Description Framework (RDF): Concepts and Abstract Syntax. *W3C Recommendation.* 10.2.2004. `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`
3. Vaneková, V. – Bella, J. - Gurský, P. – Horváth, T.: Fuzzy RDF in the Semantic Web: Deduction and Induction. *Proceedings of the WDA 2005, s. 16-29.* Elfa Academic Press, Košice. Jún 2005. ISBN 80-8086-015-7.