UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA
PRÍRODOVEDECKÁ FAKULTA

UNIFIKÁCIA PRODUKTOV INTERNETOVÝCH OBCHODOV

2013 **Bc. Peter ŠINAĽ**

UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA
PRÍRODOVEDECKÁ FAKULTA

UNIFIKÁCIA PRODUKTOV INTERNETOVÝCH OBCHODOV

DIPLOMOVÁ PRÁCA

Študijný program:	Informatika
Pracovisko (katedra/ústav):	Ústav informatiky
Vedúci diplomovej práce:	RNDr. Peter Gurský, PhD.

Košice 2013

Bc. Peter ŠINAĽ

Univerzita P. J. Šafárika v Košiciach
Prírodovedecká fakulta

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Peter Šinaľ
Študijný program: Informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: Diplomová práca
Jazyk záverečnej práce: slovenský

Názov: Unifikácia produktov internetových obchodov

Názov EN: Unification of e-shops products

Cieľ: 1, Navrhnúť spôsob prípravy dát a uloženia produktov z viacerých produktových katalógov s možnosťou unifikácie rovnakých druhov produktov.
2, Navrhnúť algoritmus unifikácie produktov rovnakého druhu z viacerých zdrojov
3, Navrhnuté riešenie otestovať nad reálnou doménou.

Literatúra: 1, Jiří Dokulil, Jakub Yaghob, Filip Zavoral: Evoluce replikačních algoritmů v stohově orientovaných systémech, report
2, D. Bednarek, D. Obdrzalek, J. Yaghob, F. Zavoral: Data Integration Using DataPile Structure. In proc. of the 9th East-European Conference on Advances in Database and Information Systems, Tallinn, Estonia, 2005
3, Peter Christen: Data Matching, Concepts and Techniques for Record Linkage, Entity resolution, and Duplicate Detection, 2012

Vedúci: RNDr. Peter Gurský, PhD.
Ústav: ÚINF - Ústav informatiky

Dátum zadania: 24.10.2011

Dátum schválenia: 19.04.2013

prof. RNDr. Viliam Geffert, DrSc.
riaditeľ ústavu

Čestné vyhlásenie

Vyhlasujem, že som celú diplomovú prácu vypracoval samostatne s použitím uvedenej literatúry.

Košice, 19. Apríl 2013 .....

vlastnoručný podpis

Poďakovanie

Rád by som sa poďakoval vedúcemu diplomovej práce RNDr. Petrovi Gurskému, PhD. za cenné rady a vedenie tejto práce.

Abstrakt v štátnom jazyku

Táto práca sa zaoberá návrhom algoritmu unifikácie produktov rovnakého druhu z viacerých zdrojov. Práca začína návrhom spôsobu prípravy dát a uloženia produktov z viacerých produktových katalógov s možnosťou unifikácie rovnakých druhov produktov. Pre optimalizáciu rýchlosti je v algoritme implementovaná relevantnosť atribútov a zdrojová závislosť. Navrhnutý algoritmus unifikácie produktov je otestovaný nad reálnou doménou produktov.

Abstrakt v cudzom jazyku

This thesis deals with the design of unification algorithm products of the same type from several sources. The first step is to propose a method for data preparation and storage products from several product catalogs with the possibility of unification same kinds of products. To optimize the speed of the algorithm are implemented the relevance of the attributes and resource dependence. The proposed algorithm unification of products is tested with the domain of real products.

Obsah

Úvod

Táto diplomová práca sa zaoberá problematikou identifikovania zhodnosti produktov z rôznych zdrojov. Zdrojom dát sú internetové obchody. Jeden obchod ponúka produkty spravidla z viacerých domén. Produkty každej domény sú charakterizované prostredníctvom hodnôt atribútov. Atribúty produktov sú špecifické pre každú doménu.

Implementačná časť práce je realizovaná ako modul v rámci projektu Kapsa. Projekt Kapsa má za cieľ sťahovať celé internetové obchody a ukladať ich v štruktúrovanej kánonickej reprezentácii do jedného úložiska. Štruktúra úložiská umožní efektívne vyhľadávanie produktov. Výsledky celého procesu sú prezentované prostredníctvom webového rozhrania pre používateľa.

Dôležitou súčasťou tohto procesu je identifikovanie zhodných produktov z rôznych internetových obchodov. Proces identifikovania zhodnosti produktov nazývame unifikácia. V tejto práci sme navrhli a implementovali vlastný algoritmus procesu unifikácie založený na porovnávaní produktov podľa ich atribútov. V návrhu algoritmu sme sa inšpirovali všeobecným modelom procesu unifikácie[3].

V práci v prvom rade navrhujeme spôsob prípravy dát a uloženia produktov z viacerých produktových katalógov. Je to základný predpoklad pre unifikáciu rovnakých druhov produktov. Tento proces zodpovedá počiatočnému kroku všeobecného modelu unifikácie – predpríprava dát a je popísaný v tretej kapitole. Jednotlivé kroky všeobecného modelu procesu unifikácie sú charakterizované v druhej kapitole.

Ďalšiu časť práce predstavuje návrh algoritmu unifikácie produktov rovnakého druhu z viacerých zdrojov podľa atribútov. V rámci uvedeného návrhu algoritmu sa realizujú ďalšie dva kroky všeobecného modelu procesu unifikácie, konkrétne porovnanie záznamov a klasifikácia. Porovnanie záznamov sa vykonáva pomocou balíka pomocných funkcií. Ich význam a funkcionalita sú vystihnuté v štvrtej kapitole s názvom Porovnávacie funkcie. Charakteristika a analýza nami navrhnutého algoritmu unifikácie podľa atribútov je prezentovaná v piatej kapitole.

Záverečná časť práce je venovaná testovaniu navrhnutého riešenia algoritmu unifikácie, pre ukladanie produktov z viacerých produktových katalógov, nad reálnou doménou produktov. V tejto poslednej časti je zapracovaný finálny krok všeobecného

modelu procesu unifikácie – vyhodnotenie. Výsledky testov, ako aj možnosti ďalšieho využitia navrhnutého algoritmu unifikácie podľa atribútov sú jadrom šiestej kapitoly.

1 Charakteristika problému a motivácia

V súčasnej dobe používatelia internetu čoraz viackrát využívajú služby rôznych internetových obchodov pre zakúpenie ľubovoľných produktov. Každý internetový obchod je tvorený jedným produktovým katalógom. Produktový katalóg obsahuje informácie o jednej alebo viacerých doménach produktov. Hlavná požiadavka používateľov je mať maximálne množstvo informácií o produkte a porovnanie týchto informácií skrz viaceré internetové obchody. Základným cieľom projektu Kapsa je práve naplnenie uvedenej požiadavky.

Počiatočným krokom v projekte Kapsa je získanie informácií o produktoch z viacerých internetových obchodov. Proces získavania dát o produktoch sa realizuje pomocou programu nazvaného crawler. Crawler je program, ktorý dokáže so zvoleného internetového obchodu získať produktové stránky konkrétnej domény. Produktová stránka je webová stránka, ktorá obsahuje informácie o konkrétnom produkte. Pre získanie korektných dát je dôležitá správna konfigurácia crawlera.

Po získaní súborov, ktoré predstavujú produktové stránky konkrétnej domény, sa vykoná anotácia dát získaných crawlerom. Anotácia určí, aké atribúty popisujú produkt danej domény. Niektoré pomenovania atribútov vystupujú vo viacerých doménach. Nie vždy popisujú tú istú vlastnosť produktu. K správnemu určeniu atribútov nám môže pomôcť aj vedomosť o tom, aké hodnoty nadobúda konkrétny atribút v danej doméne. Hodnoty atribútov sú naplnené pomocou extrakcie. Počas extrahovania dát môžu nastať rôzne chyby, ktorých dôsledkom je nekorektné naplnenie hodnôt atribútov. Tiež môže nastať prípad, že hodnota sa nedá vyextrahovať, pretože v zdrojovom súbore príslušná hodnota atribútu chýba, alebo sa nedá spracovať.

Atribúty produktov môžeme rozdeliť do dvoch hlavných skupín, na zdrojovo závislé a zdrojovo nezávislé. Zdrojovo nezávislé atribúty sú vlastnosťami produktov bez ohľadu na to kde sa predávajú, napríklad uhlopriečka pri televízore. Ako príklad zdrojovo závislého atribútu môžeme uviesť atribút cena. Keďže hodnota atribútu cena pre ten istý produkt, je rôzna v jednotlivých internetových obchodoch.

Po získaní reprezentácie produktov prostredníctvom hodnôt svojich atribútov sa vykonáva unifikácia skrz jednotlivé produktové katalógy, teda identifikovanie rovnakých produktov v rôznych produktových katalógoch. V tejto práci sme navrhli algoritmus unifikácie podľa atribútov. Pri návrhu algoritmu sme sa čiastočne inšpirovali riešením problému unifikácie, ktoré bolo použité v rámci stohového systému (kapitola

5.1 Existujúce riešenie v stohovom systéme), avšak prioritným zdrojom pre návrh nášho algoritmu unifikácie bol všeobecný model unifikácie[3]. S predpokladom získania presnejších výsledkov sme rozšírili všeobecný model unifikácie o vlastnosti atribútov - uvedenú zdrojovú závislosť a relevantnosť atribútov (kapitola 5.2.1 Relevantnosť atribútov).

Výsledky unifikácie sú jednotne prezentované prostredníctvom webového rozhrania pre používateľa. Ten následne môže prehľadávať produkty na základe zvolených kritérií.

V súčasnosti sa proces unifikácie uplatňuje v rôznych oblastiach manipulácie s dátami, v ktorých sa vyskytuje požiadavka na zjednocovanie dát. Najčastejšie v spojitosti s osobnými údajmi napríklad pri spájaní databáz ľudských zdrojov. Ďalej napríklad v oblasti biológie, genetiky, kde sa hľadajú zhody v dátach. V oblasti produktových katalógov, ktorej sa venujeme, sme nenašli žiadny algoritmus, ktorý by presne spĺňal požiadavky nášho zadania.

2 Unifikácia vo všeobecnosti

V tejto kapitole opíšeme proces unifikácie podľa všeobecného modelu [3]. Tento model bol navrhnutý na základe niekoľkých doposiaľ existujúcich riešení procesu unifikácie, ktoré boli použité v rôznych oblastiach.

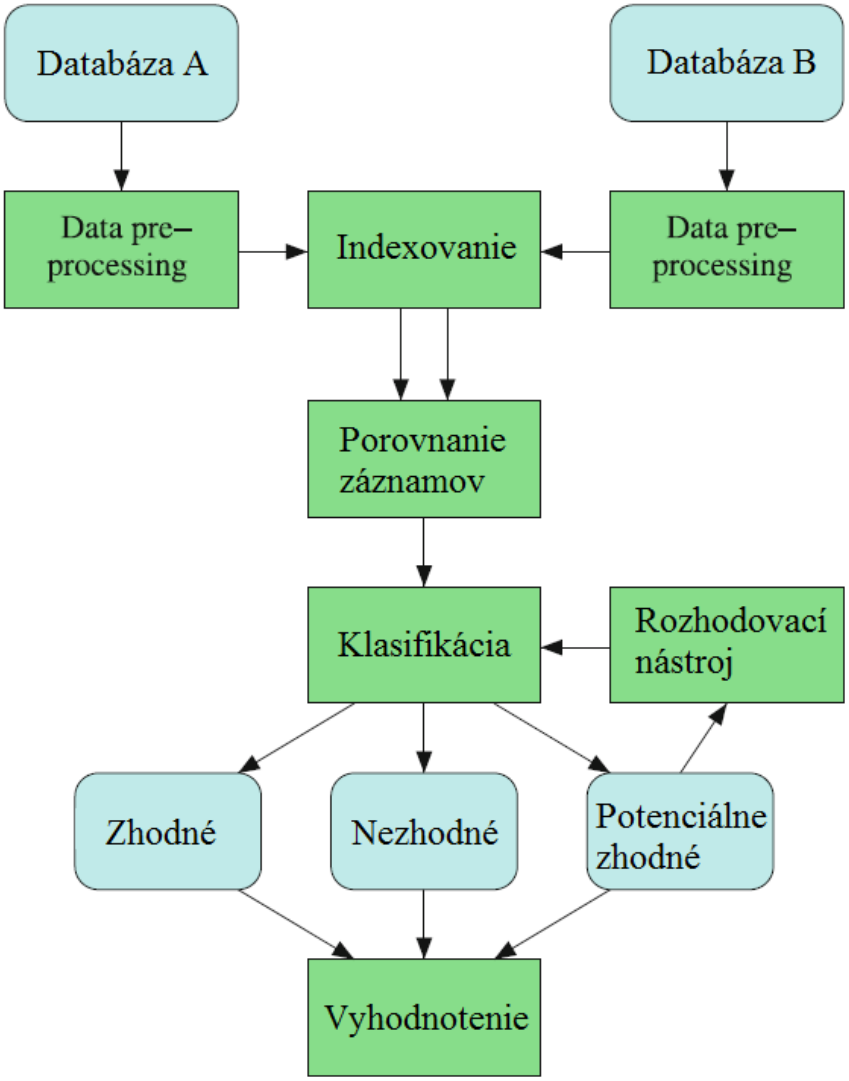
Na základe všeobecného modelu je unifikácia proces, ktorý pozostáva s piatich krokov. Konkrétne :

- Predpríprava dát (data preprocessing)
- Indexovanie
- Porovnanie záznamov
- Klasifikácia
- Vyhodnotenie

Najskôr uvedený model unifikácie popíšeme všeobecne. Potom sa vyjadríme k problémom, ktorými je potrebné sa zaoberať v rámci jednotlivých krokov, kvôli naplneniu našich cieľov.

Prvý krok sa nazýva predpríprava dát (data preprocessing). Uvedený krok unifikácie zabezpečuje, aby dáta z rôznych zdrojov boli v rovnakom formáte pre ďalší postup. Druhý krok sa nazýva indexovanie. Jeho hlavným cieľom je redukovať kvadratickú zložitosť procesu unifikácie pomocou vhodných dátových štruktúr, ktoré umožňujú účinné a efektívne vytváranie dvojíc záznamov pre porovnanie. V treťom kroku nastáva porovnanie vytvorených dvojíc záznamov, ktoré predstavujú výstup predošlého kroku, pomocou porovnávacích funkcií. V nasledujúcom kroku, ktorý je označený ako klasifikácia, sú dvojice záznamov zatriedené na základe vypočítaných hodnôt podobnosti z kroku porovnania do troch skupín a to zhodné, nezhodné a potenciálne zhodné. O zhode, či nezhode záznamov zatriedených do skupiny potenciálne zhodných rozhodne iný rozhodovací nástroj. Posledným krokom je vyhodnotenie úspešnosti unifikácie.

Na obrázku 1 je znázornený príklad všeobecného modelu unifikácie dát, ktorá je tvorená piatimi vyššie prezentovanými krokmi. Ako vstup celého procesu uvažujeme dve databázy.



Obr. 1 Všeobecný model unifikácie

Veľká časť databáz reálneho sveta obsahuje nekonzistentné a neúplné dáta, a to v dôsledku viacerých faktorov. Je všeobecne známe, že dáta nízkej kvality spôsobujú finančné straty firmám, podnikom alebo vládam. Pre všetky typy analýzy, spracovania alebo manažovania dát platí „garbage in - garbage out“, teda výsledkom nekvalitných dát na vstupe sú nekvalitné dáta na výstupe. Existuje niekoľko dimenzií kvality dát. Niektoré relevantné pre proces unifikácie sú :

- Presnosť – vykonanie kontroly pri zadávaní dát, kvôli overeniu správnosti dát.
- Úplnosť – ako úplné sú dáta v databázach, koľko hodnôt atribútov, ktoré sú použité v unifikácií, je / nie je vyplnených.
- Konzistentnosť – konzistentné hodnoty atribútov vo vstupných databázach, kvôli zabezpečeniu integrity dát.

- Informácia o čase – ako staré sú údaje, ktoré sú k dispozícií pre prezentovaný proces unifikácie, časové zladenie údajov dvoch databáz. Uvedený faktor môže byť dôležitý pre atribúty, ktorých hodnoty sa menia v čase.
- Dostupnosť – či sú všetky požadované údaje v databáze, či je v databázach potrebný počet atribútov, ktoré dokážu reprezentovať daný subjekt v procese unifikácie.
- Vierohodnosť – môžeme hodnoty v databázach považovať za dôveryhodné, pravdivé, alebo je možné, že niektoré hodnoty sú nesprávne, klamlivé.

Pravdepodobne najdôležitejšími vyššie popísanými dimenziami sú práve presnosť a konzistentnosť, pretože ďalšie kroky procesu unifikácie indexovanie, porovnávanie záznamov a klasifikácia (ktoré popíšeme nižšie), vyžadujú na vstupe presné a konzistentné dáta, aby unifikácia poskytla čo najkvalitnejší výsledok.

2.1 Predpríprava dát pre proces unifikácie

Ako sme vyššie uviedli, databázové tabuľky alebo iné štruktúry, ktoré so použité ako zdroje dát na vstupe pre proces unifikácie môžu byť rôzneho formátu, štruktúry alebo obsahu. Proces unifikácie vyžaduje, aby dáta z rôznych zdrojov boli štandardizované. Hlavným cieľom kroku predprípravy dát je zabezpečiť, aby atribúty používané v procese unifikácie mali rovnakú štruktúru a ich obsah bol v rovnakom formáte.

Aby sme dokázali realizovať definované ciele práce, taktiež sme potrebovali zakomponovať uvedený krok predprípravy dát do nášho postupu. Keďže dáta o produktoch sme získavali s rôznych zdrojov, bolo potrebné zabezpečiť korektný vstupný formát dát (kapitola 3 Získavanie a príprava dát).

Tri hlavné prvky, ktoré tvoria základ v uvedenom kroku predprípravy dát a ktoré sme uvažovali aj v našom návrhu vstupných štruktúr sú:

- **Odstránenie nežiaducich znakov a slov**

  Niektoré hodnoty atribútov vo vstupnej databáze môžu obsahovať znaky, slová, termíny alebo skratky, ktoré obsahujú neužitočné informácie pre proces unifikácie a môžu resp. mali by byť odstránené.
- **Extrakcia skratiek a oprava pravopisných chyb**

  Prostredníctvom detekcie sú hodnoty, ktoré obsahujú typografické chyby alebo variácie opravené a rozšírené skratky sú nahradené štandardnými formami.

- **Segmentácia atribútov do dobre definovaných foriem.**

  Najťažší krok, pretože často existuje niekoľko možných priradení hodnôt atribútov do dobre definovaných vstupných foriem pre proces unifikácie. Napríklad zoberme adresu. Prvý vstupný formát : ulica, lokalita. Druhý : číslo, názov ulice, typ, názov lokality, poštové smerovacie číslo. Úlohou je určiť, ktoré priradenie je a najvhodnejšie s pohľadu korektnosti, a to následne vykonať.

Je dôležité poznamenať, že proces predspracovania dát nemôže modifikovať pôvodné vstupné dáta. Ako náhle by došlo k modifikácií originálnych vstupných dát, mohli by sme predprípravou stratiť nejaké informácie použiteľné inými nástrojmi. Aj túto skutočnosť sme uvažovali v našom postupe, a preto sme dáta preklápali cez niekoľko tabuliek, aby sme v prípade potreby mali k dispozícií zálohu pôvodných dát.

2.2 Indexovanie

Databázové tabuľky so štandardizovanými a vhodne uloženými dátami sú pripravené na realizáciu ďalšieho kroku procesu unifikácie, ktorým je indexovanie. Hlavným cieľom indexovania v procese unifikácie je znížiť počet dvojíc záznamov, ktoré je potrebné porovnať. Bez indexovania, proces unifikácie dvoch databáz vyžaduje to, že každý záznam z jednej databázy je potrebné v uvedenom procese porovnať s každým záznamom z druhej databázy. Celkový počet porovnaní je potom kvadratický vzhľadom na veľkosť porovnávaných databáz.

Najčastejší prístup, ktorý sa používa v procese indexovania je definícia tzv. blocking keys , teda blokových kľúčov na základe ktorých sú záznamy rozdelené do blokov. Následne v procese unifikácie v kroku porovnanie dvojíc záznamov sa porovnávajú iba záznamy z odpovedajúcich blokov, dôsledkom čoho je znížená výpočtová zložitosť procesu unifikácie.

Okrem uvedenej indexovacej techniky existuje aj niekoľko ďalších metód, ako riešiť krok indexovania v rámci procesu unifikácie.

V našom postupe realizácie návrhu unifikačného algoritmu sme indexovací krok procesu unifikácie vynechali. Počet porovnaní je znížený hlavne predpokladom, že porovnanie bude realizované iba medzi objektmi rovnakých domén (t.j. neporovnávame napr. televízory a mikrovlnky). Ďalšie zníženie časovej zložitosti porovnania umožnila relevantnosť atribútov (kapitole 5.2.1 Relevantnosť atribútov).

2.3 Porovnanie záznamov

Ďalším krokom procesu unifikácie na základe všeobecného modelu je krok porovnania záznamov. Skôr ako použiť rovnosť, ako porovnávaciu metódu pre porovnanie dvojíc hodnôt atribútov, je vhodnejšie pre proces porovnania dát aplikovať porovnávacie funkcie. Tie budú prezentované v kapitole 4. s rovnocenným názvom. Porovnávacia funkcia vracia nejaký indikátor toho, ako podobné sú dve hodnoty atribútov.

Najčastejšie sa uvažuje funkcia podobnosti $s = sim(ai, aj)$ (similarity - podobnosť), ktorá počíta numerickú podobnosť s medzi dvoma hodnotami atribútov a_i a a_j, kde a_i a a_j môžu byť reťazce, čísla, dátumy, časy, veky, geografické umiestnenia, alebo iné zložitejšie hodnoty ako napr. text, XML dokumenty. Je predpoklad, že uvedená funkcia podobnosti *sim* generuje normalizovanú hodnotu podobnosti v intervale $0<=s<=1$.

Všeobecné vlastnosti uvedenej funkcie sú:

- $sim(a_i , a_j)=1$: Výsledkom porovnania hodnoty so sebou samou je exaktná podobnosť
- $sim(a_i , a_j)=0$: Podobnosť hodnôt, ktoré sú kompletne odlišné je 0. (Čo je príčinou, že dáta sú úplne rôzne, závisí od typu porovnávaných dát)
- $0<=sim(a_i , a_j)<=1$: Približná podobnosť, ktorá je vypočítaná ak dve hodnoty atribútov sú trocha podobné jedna druhej. (Tiež závisí od typu porovnávaných dát).

Pri realizácií kroku porovnania záznamov v rámci nášho návrhu algoritmu unifikácie podľa atribútov, bolo najväčším problémov správne zvoliť príslušnú porovnávaciu funkciu pre porovnanie hodnôt príslušného atribútu. Riešením uvedeného problému bolo zatriedenie atribútov do niekoľkých skupín typov, kde každá skupina mala priradenú konkrétnu porovnávaciu funkciu (kapitola 3.3 Štruktúra tabuľky WRAPJOINEDDATA).

Výsledkom kroku porovnania záznamov v rámci procesu unifikácie sú teda číselné hodnoty, ktoré vyjadrujú podobnosť medzi dvojicami záznamov, ktoré boli určené na porovnanie.

2.4 Klasifikácia záznamov

V tomto kroku unifikácie prebieha vyhodnocovanie podobnosti dvojíc záznamov, ktoré sú reprezentované číselnými hodnotami a vypočítané v predchádzajúcom kroku porovnania záznamov. Hlavnou ideou popisovaného kroku klasifikácie je, že ak na základe podobnosti sú dvojice záznamov zhodné, potom odkazujú na tu istú entitu reálneho sveta, v našom prípade produkty. V opačnom prípade ide o dve rozdielne entity.

Existujú dva základné postupy pre proces klasifikácie záznamov.

- dvoj-triedová klasifikácia - každý porovnaný záznam je klasifikovaný ako zhodný (match) alebo rozdielny (non-match)
- troj-triedová klasifikácia - rozdeľuje porovnané záznamy do troch skupín, a to zhodné (match), rozdielne (non-match) a potenciálne zhodné (potential match).

V našom návrhu algoritmu unifikácie podľa atribútov sme sa rozhodli použiť v rámci kroku klasifikácie práve troj-triedovú klasifikáciu. Dôvodom je hlavne nekvalitnosť vstupných dát, získaných nepresnými metódami anotácie a extrakcie z webových stránok a teda nemožné plne automatické spracovanie unifikáciou.

Aby sme mohli klasifikovať jednotlivé záznamy podľa vyššie prezentovaných postupov, potrebovali sme určiť celkovú podobnosť porovnávaných párov záznamov pre jednotlivé atribúty. Zvolili sme si najpoužívanejšiu metódu na určenie celkovej podobnosti, ktorá zahŕňa výpočet tzv. sumovanej podobnosti (S*imSum*), cez všetky vypočítané hodnoty podobnosti pre jednotlivé atribúty.

Problémom uvedenej metódy, ktorý sme museli vyriešiť aj v našom návrhu algoritmu unifikácie je prípad, kedy výsledkom porovnania dvoch hodnôt daného atribútu je nula ako dôsledok toho, že niektorá hodnota z dvojice nie je vyplnená (čiže null). Potom zarátanie uvedenej nulovej hodnoty môže negatívne skresliť výsledok. Preto sme sa rozhodli použiť riešenie prezentovanej metódy pre výpočet sumovanej podobnosti, v ktorom sme uvažovali aj porovnávanie s tzv. „null“ hodnotami. Toto riešenie bude podrobnejšie opísané v kapitole 5.3.1 Výpočet podobnosti hodnôt atribútov.

Keďže sme sa rozhodli pre aplikáciu troj-triedovej klasifikácie bolo potrebné stanoviť dve prahové hodnoty, a to dolnú hranicu zhodných objektov $h_{dolná}$, a hornú hranicu rôznych objektov $h_{horná}$. Klasifikácia je potom reprezentovaná takto :

- $SimSum[r_i, r_j] \geq h_{dolná} \rightarrow [r_i, r_j]$ = zhodné
- $h_{horná} < SimSum[r_i, r_j] < h_{dolná} \rightarrow [r_i, r_j]$ = potenciálne zhodné
- $SimSum[r_i, r_j] < h_{horná} \rightarrow [r_i, r_j]$ = nezhodné

Okrem zvolenej troj-triedovej klasifikácie v kombinácií s výpočtom sumovanej podobnosti, existuje niekoľko ďalších techník ako realizovať krok klasifikácie v rámci procesu unifikácie.

Pravdepodobnostná klasifikácia je založená na výpočte pravdepodobnosti zhody dvojíc záznamov, na základe ktorej sa rozhodne o výsledku klasifikácie.

Ďalšia technika sa nazýva klasifikácia na základe pravidiel. Uvedená metóda klasifikuje dvojice záznamov na základe vopred definovaných pravidiel.

Potom sa používajú aj rôzne aktívne učiace sa prístupy pre proces klasifikácie. Menšou nevýhodou týchto metód, pre korektné klasifikovanie, je potreba mať k dispozícií tréningovú množinu dát.

Voľba správnej techniky kroku klasifikácie v procese unifikácie závisí od viacerých faktorov, ako napríklad : dostupnosť tréningových množín dát, distribúcia hodnôt atribútov, konkrétna doména, veľkosť domény.
Výsledkom kroku klasifikácie v rámci procesu unifikácie je zatriedenie jednotlivých entít na základe číselných výsledkov porovnania do dvoch alebo troch skupín v závislosti od použitej dvoj alebo troj – triedovej klasifikácie.

2.5 Vyhodnotenie

Posledným krokom procesu unifikácie na základe všeobecného modelu je vyhodnotenie úspešnosti uvedeného procesu. Podobne, ako pre realizáciu predošlých krokov procesu unifikácie, aj tu existuje viacero metód, ako vyhodnotiť úspešnosť prezentovaného procesu. Základný postup je založený na rozdelení klasifikovaných, predtým porovnaných, dvojíc záznamov do štyroch skupín :

1. True positives (kladne zhodné) – dvojice záznamov, ktoré boli klasifikované ako zhodné a skutočne sú zhodné. Teda dvojice záznamov, ktoré odkazujú na rovnakú entitu.

2. False positives (záporne zhodné) – dvojice záznamov, ktoré sú klasifikované ako zhodné, ale v skutočnosti nie sú zhodné. Uvedené dvojice záznamov odkazujú na rozdielne entity. Teda nastala chyba pri klasifikácií.
3. True negatives (kladne nezhodné) – dvojice záznamov, ktoré sú klasifikované ako nezhodné a skutočne sú nezhodné. Teda záznamy reprezentujú dve rozdielne entity reálneho sveta.
4. False negatives (záporne nezhodné) – dvojice záznamov, ktoré boli klasifikované ako nezhodné, ale v skutočnosti sú zhodné. Záznamy odkazujú na tu istú entitu reálneho sveta a teda nastala chyba pri klasifikácií.

Hlavným cieľom klasifikácie je identifikovať a korektne klasifikovať čo najviac kladne zhodných dvojíc záznamov a minimalizovať počet záporne zhodných a záporne nezhodných záznamov.

Na základe počtu kladne zhodných, záporne zhodných, kladne nezhodných, záporne nezhodných vieme vypočítať rôzne kvalitatívne hodnoty, ktoré charakterizujú úspešnosť unifikácie. Napríklad vieme vypočítať presnosť unifikácie pomocou vzorca :

$$p = \frac{TP + TN}{TP + FP + TN + FN}$$

Číselné hodnoty reprezentujúce presnosť procesu unifikácie rôznych produktov podľa atribútov, ktoré získame pomocou prezentovaného vzorca nám umožnia získať predstavu o presnosti vykonávaného procesu nad danými dátami.

Pri našom návrhu algoritmu sme sa nijako konkrétne nesnažili zapracovať prezentovaný krok vyhodnotenia priamo do algoritmu. Kroku vyhodnotenie v rámci nášho algoritmu unifikácie podľa atribútov sa chceme venovať v rámci realizácie posledného cieľa našej práce, a to otestovania navrhnutého algoritmu unifikácie nad reálnou doménou (kapitola 6. Testy).

3 Získavanie a príprava dát

Aby sme naplnili jednotlivé ciele diplomovej práce, potrebovali sme mať k dispozícií dáta. Získať dáta spôsobom, ktorý je založený na komunikácií s niektorým reálnym produktovým katalógom, je dosť zdĺhavý a vo väčšine prípadov aj neúspešný proces. Rozhodli sme sa preto, že dáta získame metódou, ktorá sa nazýva crawlovanie dát. Uvedená metóda v sebe zahŕňa výber vhodného webového crawlera, pomocou ktorého získavame dáta z nami zvoleného produktového katalógu. Následne je potrebné crawlované dáta wrapovať, teda vykonať extrakciu a anotáciu dát.

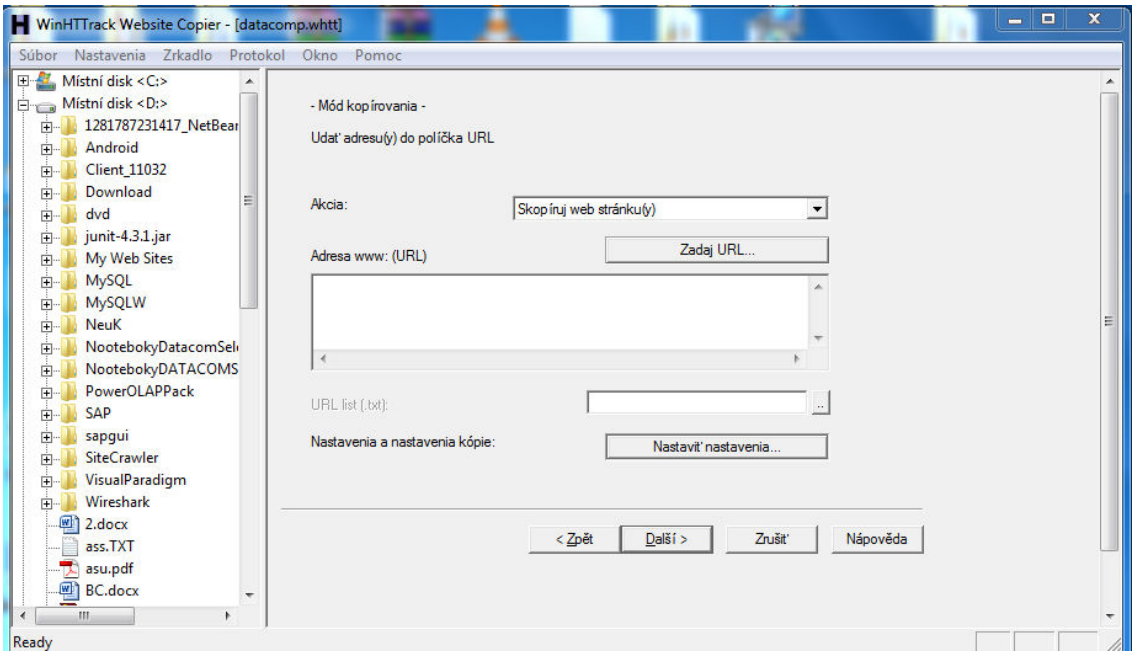
3.1 Crawlovanie a wrapovanie dát

V tejto podkapitole sa venujeme opisu už spomínaného procesu crawlovania a následného spracovania nacrawlovaných súborov pomocou wrapera.

Proces crawlovania používateľom zadanej web stránky začína web crawler analýzou zadanej web stránky. Uvedená analýza prebieha spôsobom hľadania hypertextových odkazov na zadanej webovej stránke na ďalšie webové stránky. Nájdené odkazy umožnia vytvorenie nových spojení, kde sa realizuje vyššie popísaná analýza rekurzívne. Webový crawler sa v skutočnosti nepohybuje po viacerých počítačov v internete, ako napríklad vírusy alebo inteligentní agenti, ale pracuje len na jednom počítači. V podstate posiela HTTP požiadavku na dokumenty iným počítačom v internete, podobným spôsobom, ako keď používateľ odklikáva hypertextové odkazy, ktoré sa nachádzajú na webových stránkach. Následne z odpovedí, ktoré získa po „internetovej“ komunikácií, vyselektuje požadované dokumenty, ktoré v ďalšom kroku uloží do používateľom vopred stanoveného priečinka.

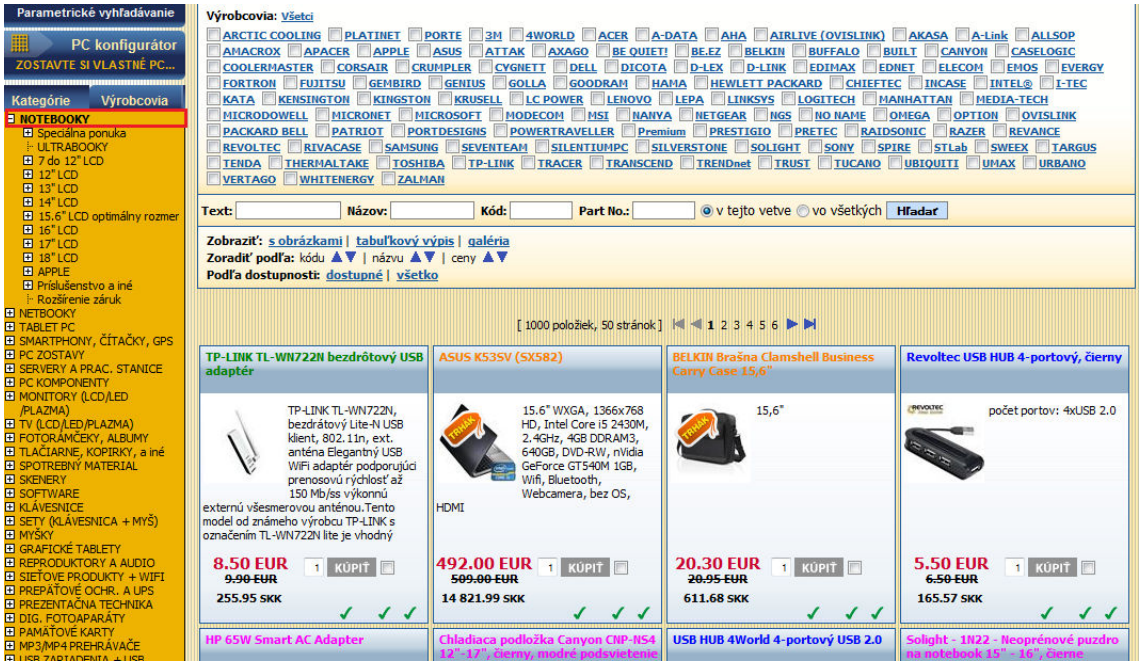
Existuje niekoľko dostupných implementácií webového crawlera. My sme sa počas hľadania informácií o týchto programoch stretli s niekoľkými, konkrétne ide o: WinWebCrawler, Mozenda, HttTrack Website Copier.

Následne sme sa rozhodli, že pri procese získavania dát pomocou vyššie spomenutej metódy crawlovania dát použijeme program WinHTTrack Website Copier (obr. 2).



Obr. 2 WinHTTrack Website Copier

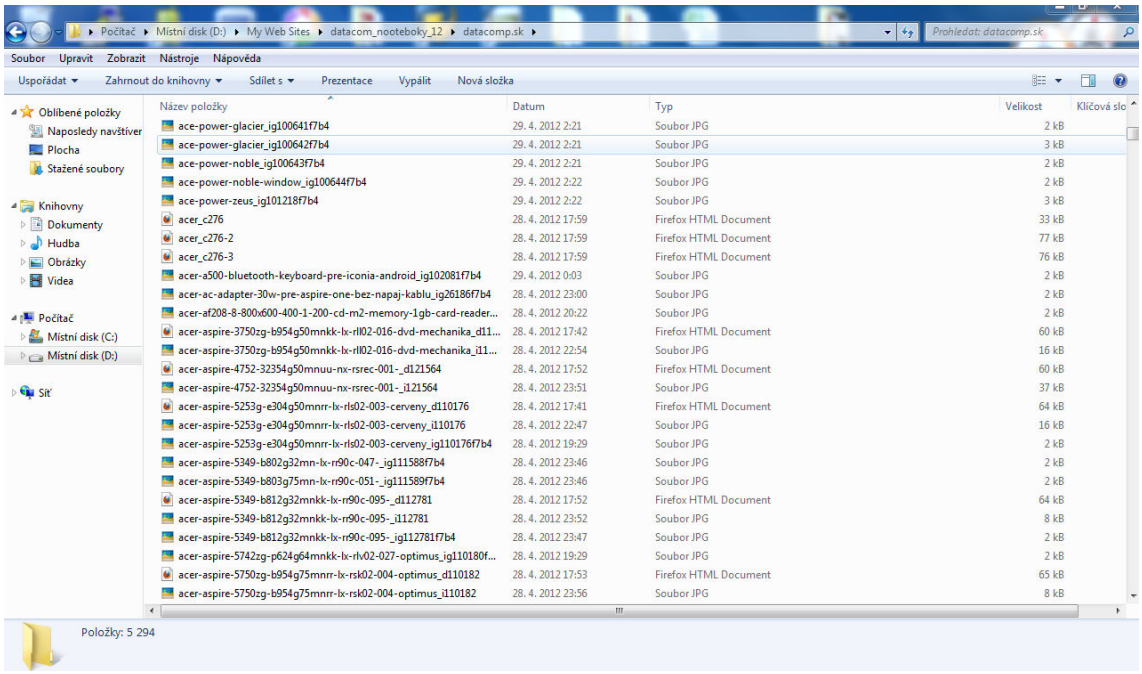
V ďalšej časti tejto podkapitoly opíšeme podrobnejšie postup crawlovania dát produktov konkrétnej domény. Pre začiatok sme si vybrali doménu notebooky s konkrétneho produktového katalógu datacomp.sk. Na obrázku 3 prezentujeme zobrazenú doménu notebooky.



Obr. 3 Doména notebooky

Výsledkom procesu crawlovania nami zvoleným webovým crawlerom získavame množinu súborov, ktorú uloží uvedený program do stanoveného priečinka na pevnom disku. Okrem html súborov spomínaná množina obsahuje aj obrázky, javascript

súbory a mnohé iné (obr. 4). Z tejto množiny súborov nás zaujímajú len súbory, ktoré reprezentujú produktové stránky reprezentujúce produkty patriace do zvolenej domény notebooky. Pre dosiahnutie uvedenej požiadavky na konkrétne súbory bolo potrebné vybrať určitú podmnožinu súborov, ktorá vyhovovala zadaným kritériám.



Obr. 4 Množina nacrawlovaných súborov

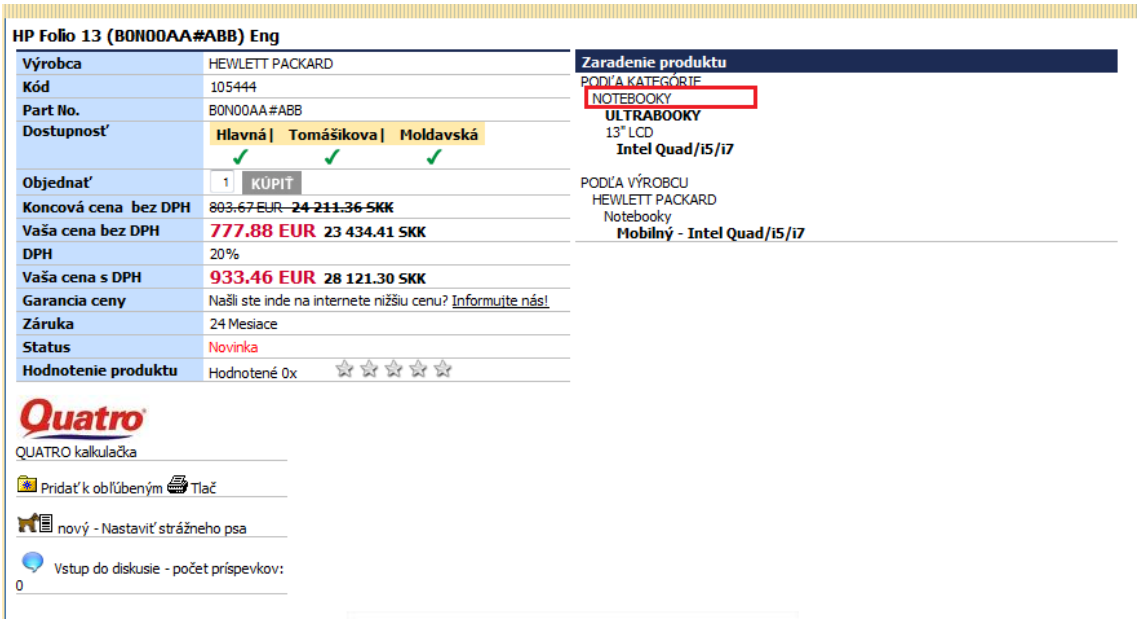
Preto sme následne realizovali proces spracovania crawlovaných dát, ktorý pozostáva z dvoch krokov:

- filtrovanie nacrawlovaných súborov
- wrappovanie dát.

Proces filtrovania kompletnej množiny súborov sme realizovali spôsobom, implementácie programu v Jave, ktorý prešiel všetky súbory danej množiny a na základe elementu v html súbore, charakterizujúceho príslušnosť do danej domény vytvoril podmnožinu, ktorá obsahovala len požadované súbory. Pre danú doménu notebooky príslušný element predstavoval reťazec, ktorý sa nachádza na obrázku 5 v červenom rámčeku.

Obr. 5 Element, charakterizujúci príslušnosť danej doméne

Uvedený element charakterizuje produktovú stránku danej domény notebooky. Na uvedenej produktovej stránke sa nachádza v časti, ktorá vyjadruje príslušnosť produktu do danej domény (obr. 6).



Obr. 6 Umiestnenie elementu na produktovej stránke

Po získaní množiny vybraných súborov, môžeme realizovať druhý krok procesu spracovania súborov, a to wrappovanie dát.

Uvedený krok sme realizovali pomocou wrappera implementovaného v Jave. Hlavnou ideou fungovania wrappera je fakt, že prechádza každý jeden zo súborov množiny, ktorú sme získali v predchádzajúcom procese a hľadá elementy v HTML kóde, ktoré určujú výskyt hodnôt atribútov, inými slovami priradzuje hodnoty jednotlivým atribútom pre daný produkt.

Pomocou uvedeného wrappera sme teda realizovali proces anotácie a extrakcie jednotlivých hodnôt atribútov danej domény. Získali sme množinu produktov patriacich do danej domény, reprezentovaných pomocou hodnôt svojich atribútov.

Uvedeným procesom sme získali dáta konkrétnej domény notebooky z konkrétneho produktového katalógu datacomp.sk.

Wrapovanie stránok má tú nevýhodu, že je funkčné len do momentu, pokiaľ sa nezmení dizajn cieľového webového portálu. Toto sa stalo aj počas riešenia tejto diplomovej práce na portáli datacomp.sk, a náš wrapper sa týmto stal nepoužiteľným.

Avšak z iného zdroja sa nám podarilo získať kvalitnejšie, obsahovo rozsiahlejšie a objemovo väčšie nawrapované dáta z viacerých zdrojov reprezentujúce doménu pneumatík. Preto sme v ďalšom postupe napĺňania cieľov pracovali práve s uvedenou doménou, ktorú budeme používať aj v príkladoch uvedených v práci.

3.2 Štruktúra tabuľky WRAPDATA

Ďalším krokom po vytvorení wrappera, je uloženie ním získaných dát. Jeden z krokov procesu unifikácie popísaných v kapitole 2 (Unifikácia vo všeobecnosti) je predpríprava dát. Základnou úlohou uvedeného kroku unifikácie je navrhnúť vhodné formy reprezentácie dát, ktoré vstupujú do ďalších krokov procesu. Výber správnych štruktúr pre uloženie nawrapovaných dát zaručuje efektívnosť a správnosť realizácie ďalších krokov prezentovaného procesu unifikácie produktov podľa atribútov. Našou snahou bolo navrhnúť univerzálnu štruktúru uloženia nawrapovaných dát pre ľubovoľnú doménu.

Navrhli sme, že nawrapované dáta uložíme do tabuľky s názvom WRAPDATA, ktorá má takúto stĺpcovú štruktúru :

Zdroj – Objekt – Vlastnosť – Hodnota – Časová pečiatka

Tabuľka WRAPDATA je tvorená s piatich stĺpcov. Následne podrobnejšie charakterizujeme obsah jednotlivých stĺpcov:

- zdroj predstavuje názov produktového katalógu, z ktorého pochádzajú informácie o danom produkte
- objekt reprezentuje konkrétny produkt
- vlastnosť vyjadruje, o ktorý atribút daného produktu ide
- hodnota obsahuje konkrétnu hodnotu atribútu vyjadreného položkou vlastnosť
- časová pečiatka, ktorá reprezentuje časový udaj pridania dát.

Príklad konkrétneho objektu uloženého v navrhnutej tabuľke WRAPDATA :

zdroj: pneumatiky.sk

objekt: Tire 1

vlastnosť: Šírka

hodnota: 210.000 mm

časová pečiatka: 2012-08-06.

Nezávislosť produktu od domény, ktorú poskytuje tabuľka WRAPDATA je hlavným dôvodom prečo sme si zvolili práve túto štruktúru. Ďalšou črtou uvedenej tabuľky je, že odpadá potreba riešiť možnosť pridávania atribútov, ak nejaké pribudli napríklad z nových zdrojov. Použitím tejto tabuľky však vzniká požiadavka unifikácie názvov atribútov danej domény medzi rôznymi zdrojmi dát. Hlavným cieľom prezentovanej tabuľky WRAPDATA je teda korektné zjednotenie nawrapovaných dát do zvoleného štandardného formátu.

Ďalší problém, ktorý rieši uvedená štruktúra tabuľky WRAPDATA je možnosť, kedy niektorá hodnota ľubovoľného atribútu je klasifikovaná nesprávnym dátovým typom. V nami navrhnutej štruktúre tabuľky WRAPDATA jednotlivé hodnoty atribútov ukladáme s dátovým typom varchar (reťazce). Pre použitie uvedeného dátového typu sme sa rozhodli preto, lebo tabuľka WRAPDATA môže takto uchovávať pôvodné dáta neovplyvnené konverziou, ktorá sa deje až následne v procese predprípravy dát.

Dáta, ktoré sú uložené v tabuľke WRAPDATA v navrhnutom formáte sa následne skonvertujú do ďalšej tabuľky, ktorú sme nazvali WRAPJOINEDDATA. V procese preklápania dát z tabuľky WRAPDATA do tabuľky WRAPJOINEDDATA sú jednotlivé hodnoty atribútov konvertované na príslušný dátový typ, ktorý je určený pre každý atribút. Konverzie sú zatiaľ realizované manuálne, avšak v budúcnosti predpokladáme, že budú realizované prostredníctvom nejakého nástroja, ktorý bude uvedené konverzie realizovať automaticky.

3.3 Štruktúra tabuľky WRAPJOINEDDATA

Dáta uložené v tabuľke WRAPDATA prezentovanej v predchádzajúcej podkapitole nie sú vo vhodnom formáte pre vstup unifikácie. Aby dáta boli uložené v korektnom vstupnom formáte, bolo potrebné vytvoriť ďalšiu tabuľku. Uvedenou tabuľkou je práve tabuľka WRAPJOINEDDATA.

Tabuľka WRAPDATA plní dve základné úlohy. Prvou je, že prostredníctvom svojej štruktúry zjednocuje dáta z rôznych zdrojov do spoločného formátu. Druhou úlohou spomínanej tabuľky je, že sa v nej uchovávajú pôvodné dáta. Teda v prípade akejkoľvek závažnejšej straty informácií (napr. nepodarí sa konverzia), ktorá môže nastať s veľmi malou pravdepodobnosťou v dátach uložených v tabuľke WRAPJOINEDDATA, poskytuje akúsi zálohu pre obnovenie dát.

Pre každú doménu vytvárame samostatnú tabuľku WRAPJOINEDDATA Máme k dispozícií niekoľko domén teda v databáze máme viac takýchto tabuliek. Tabuľka WRAPJOINEDDATA je tvorená stĺpcami, ktoré reprezentujú jednotlivé atribúty produktu. Okrem nich obsahuje aj ďalšie stĺpce, ako id_download (identifikátor stiahnutia), id_source (identifikátor zdroja). Stĺpec id_download obsahuje identifikátor stiahnutia dát z produktového katalógu. Stĺpec id_source reprezentuje označenie zdroja, z ktorého pochádzajú informácie o produktoch. Jednému identifikátoru stiahnutia odpovedá práve jeden identifikátor zdroja.

Dva základné rozdiely v porovnaní s tabuľkou WRAPDATA sú :

- hodnoty jednotlivých atribútov sú uložené v príslušných stĺpcoch pomenovaných podľa názvov atribútov danej domény
- hodnoty jednotlivých atribútov sú prekonvertované na príslušný dátový typ.

Jednotlivé atribúty sú na základe hodnôt, ktoré môžu nadobúdať klasifikované do niekoľkých základných typov. Presnejšie týchto :

- nominal – vymenovaná doména reťazcových hodnôt bez prirodzeného usporiadania, typicky typy, kategórie atď., napríklad farba
- ordinal numeric – nekonečná doména číselných hodnôt
- ordinal datetime – dátumy časy
- hierarchical – hodnoty organizované v stromovej hierarchií
- objectical – hodnoty odkazujúce na iné objekty ktoré sú hodnotou atribútu, napr. disk v notebooku
- discrete – vymenovaná doména číselných hodnôt
- string – nekonečná doména reťazcov, typický kódy produktov

Vymenované hodnoty atribútov označených typom nominal, hierarchical alebo discrete sú uložené v pomocných tabuľkách. Konkrétna hodnota atribútu typu nominal,

hierarchical alebo discrete je reprezentovaná cez identifikátor (primárny kľúč) danej pomocnej tabuľky. V tabuľke WRAPJOINEDDATA sú hodnoty atribútov označených typom nominal, hierarchical alebo discrete reprezentované práve prostredníctvom spomínaných identifikátorov.

Ako sme vyššie uviedli, jeden s rozdielov medzi tabuľkou WRAPDATA a tabuľkou WRAPJOINEDDATA je, že hodnoty jednotlivých atribútov sú prekonvertované na príslušný dátový typ. Pri napĺňaní jednotlivých stĺpcov sme preto vykonali konverzie z typu varchar (reťazec) na príslušný dátový typ. Uvedené konverzie sme realizovali manuálne, ale v budúcnosti sa dá uvažovať s nástrojom, ktorý bude konvertovať hodnoty atribútov automaticky.

Príklad konkrétneho produktu, teda pneumatiky uloženej v tabuľke WRAPJOINEDDATA :
ID : Tire 1
ID_DOWNLOAD : 3
ID_SOURCE : 1
EAN : 3528705090142
KOD : 02940
SIRKA : 235.0000
PROFIL : 40.0000
RADIAL : R
RAFEK : 18.000
LI : 91
SI : 300
VYROBCE : 22
NAZEV_VYROBCE : 23
DEZEN : PILOT SPORT CUP
TYP : S
IMGURL : http://www.treadtelpneu.cz/foto/obr__uniroyal-pilotsportcup.jpg
CENA_DOPORUCENA : 303.9100
CENA_VYSLEDNA : 239.6200
KUSU_24HOD : 0

KUSU_DATUM : 8

ARTICLE_GROUP : Null

CODE2 : Null

ARTICLE_NAME : Null

MIN7 : Null

MIN1: Null

WEIGHT : Null

PATTERN : Null

XLFR : Null

MO : Null

RUNFLAT : Null

ALLOYWHEEL_PITCH_CIRCLE : Null

ALLOYWHEEL_SIZE : Null

ALLOYWHEEL_ET : Null

CONSUMPTION : Null

BRAKING : Null

NOISE : Null

Niektoré vlastnosti atribútov danej domény, ako sú zdrojová závislosť alebo relevantnosť atribútov(bude vysvetlená nižšie) nie je možné získať procesom wrapovania dát. To znamená, že uvedené vlastnosti museli byť zadané ručne administrátorom pre každú doménu zvlášť. Je to preto, že produkty každej domény sú reprezentované špecifickými atribútmi a nie je možné vytvoriť univerzálny algoritmus, ktorý by dokázal získať a správne priradiť uvedené vlastnosti. Aj napriek uvedenému faktu sme sa rozhodli použiť daný prístup, pretože predpokladáme, že pomocou neho dosiahneme presnejšie výsledky.

4 Porovnávacie funkcie

Proces unifikácie je rozdelený do niekoľkých krokov. Jeden z nich predstavuje proces porovnania dvojíc hodnôt atribútov pomocou porovnávacích funkcií. V tejto kapitole si bližšie priblížime úlohu a spôsob fungovania porovnávacích funkcií. Porovnávacia funkcia vracia vypočítanú číselnú hodnotu podobnosti medzi dvomi hodnotami konkrétneho atribútu. Vypočítané hodnoty podobnosti sa spracovávajú v ďalšom kroku unifikácie, ktorým je klasifikácia.

Porovnávacie funkcie môžeme rozdeliť do niekoľkých skupín, na základe možnosti ich použitia. Teda niektoré z uvedených funkcií sú vhodnejšie pre výpočet podobnosti pre krátke reťazce, iné zase pre dlhšie reťazce, ďalšie sú vhodné, ak sa v reťazcoch nachádza veľa medzier atď..

Balík porovnávacích funkcií, ktorý je súčasťou nášho algoritmu sme neimplementovali my. Avšak po vzájomnej komunikácií s pracovníkmi univerzity, ktorá tento balík [4] používala vo svojom open source projekte, sme sa dohodli na možnosti jeho použitia a prípadnej modifikácie pre naše potreby. V nasledujúcej podkapitole si priblížime jednu konkrétnu porovnávaciu funkciu z uvedeného balíka porovnávacích funkcií. V ďalšej podkapitole popíšeme nami navrhnutú a implementovanú porovnávaciu funkciu.

4.1 Porovnávacia funkcia Levenshtein

Aproximačné porovnávacie funkcie založené na koncepte úpravy vzdialenosti počítajú najmenší počet úprav, ktoré je potrebné realizovať na prerobenie/prekonvertovanie jedného reťazca na iný. Rozličné implementácie uvedeného konceptu pracujú s rôznymi operáciami, ktoré si vyžadujú dané úpravy. Počet úprav medzi dvoma reťazcami predstavuje vzdialenosť, ktorá môže byť prepočítaná na podobnosť.

Úprava vzdialenosti funkcie Levenshtein je definovaná ako najmenší počet vložení, odstránení a nahradení jednoduchých znakov, ktoré sú potrebné na prekonvertovanie jedného reťazca na iný.

Použitím algoritmu implementovaného pomocou dynamického programovania, vzdialenosť medzi dvoma reťazcami s_1 a s_2 môže byť vypočítaná v čase $O(|s_1| \times |s_2|)$, použitím $O(min(|s_1|, |s_2|))$ pamäte.

Uvedený algoritmus začína vypĺňať prvý riadok a prvý stĺpec matice s korešpondujúcimi hodnotami stĺpca alebo riadka. Bunka *d[0,j]* v riadku *0* a stĺpci *j (0<=j<=|s1|)* je naplnená hodnotou *j*, a bunka *d[i,0]* v riadku *i* a stĺpci *0* je naplnená hodnotou *i (0<=i<=|s2|)*. Zvyšné bunky v matici sú naplnené hodnotami pomocou nasledujúceho rekurzívneho postupu:

- Ak $s_1[i] = s_2[j]$ potom d[i , j] = d[i-1, j-1].
- Ak $s_1[i] \neq s_2[j]$ potom

d[i , j] = minimum { d[i-1, j] + 1 zmazanie
d[i, j-1] + 1 vloženie
d[i-1, j-1] + 1 substitúcia. }

Príklad : úprava vzdialenosti funkcie Levenshtein medzi reťazcami ‘kitten’ a ‘sitting‘. Výsledok 3, je hodnota reprezentujúca počet potrebných úprav.

1. **k**itten → **s**itten (substitúcia 's' za 'k')
2. sitt**e**n → sitt**i**n (substitúcia 'i' za 'e')
3. sittin → sittin**g** (vloženie 'g').

Prezentovaný algoritmus Levenshtein úpravy vzdialenosti popisuje nasledujúci pseudokód :

```
int LevenshteinDistance(string s, string t){
int len_s = length(s), len_t = length(t)
        if      (len_s == 0) then  return len_t
        elseif  (len_t == 0) then  return len_s
        else    return minimum(LevenshteinDistance(s[1..len_s], t),
                    LevenshteinDistance(s, t[1..len_t]),
                    LevenshteinDistance(s[1..len_s], t[1..len_t])
}
```

Úprava vzdialenosti ukladá v matici len dva riadky v čase, a preto potrebuje len *O(min (|s1|, |s2|))* pamäte.

Výsledok úpravy vzdialenosti medzi dvoma reťazcami s_1 a s_2 je hodnota v pravom dolnom rohu matice, $dist_{levenshtein}(s_1,s_2) = d[|s_1|,|s_2|]$. Túto vzdialenosť môžeme prekonvertovať do podobnosti (v intervale 0 až 1) použitím vzťahu :

$$sim_{levenshtein}(s_1,s_2) = 1.0 - dist_{levenshtein}(s_1,s_2) / max(|s_1|,|s_2|)$$

Úprava vzdialenosti funkcie Levenshtein je symetrická, platí $0 <= dist_{levenshtein}(s_1,s_2) <= max(|s_1|,|s_2|)$. Ďalej platí vlastnosť, že absolútny rozdiel dĺžok

reťazcov s_1 a s_2 je menší ako $dist_{levenshtein}(s_1,s_2)$.Uvedená vlastnosť umožňuje rýchle filtrovanie tých párov reťazcov, ktoré majú veľký rozdiel v dĺžke, bez potreby výpočtu.

Iné rozšírenie základného algoritmu zahŕňa rozdielne ohodnotenie pre jednotlivé operácie. Tieto ohodnotenia môžu byť vhodné pre ďalšie výpočty.

V nami navrhnutom algoritme unifikácie produktov podľa atribútov sme sa rozhodli použiť popísanú funkciu Levenshtein. Táto funkcia je vhodná na porovnávanie kratších reťazcov. Jednotlivé hodnoty atribútov domény s ktorou pracujeme sú reprezentované práve kratšími reťazcami. Okrem spomenutej funkcie sú v prevzatom balíku implementované aj ďalšie porovnávacie funkcií, ktorých zameranie z hľadiska dĺžky reťazcov je rôzne. Napríklad: Monge Elkan – vhodná na porovnanie reťazcov, ktoré sú tvorené z viacerých slov; Qgrams Distance – vhodná na porovnanie dlhších reťazcov; Needleman Wunch – vhodná na sekvencie reťazcov.

4.2 Naša porovnávacia funkcia pre atribúty typu ordinal numeric

Spomínaný balík porovnávacích funkcií obsahuje niekoľko implementovaných porovnávacích funkcií, ktoré používame v našom algoritme. Okrem už vytvorených funkcií sme navrhli a implementovali aj vlastnú porovnávaciu funkciu. Uvedená funkcia je vytvorená pre vyhodnocovanie jedného z vyššie prezentovaných typov atribútov a to typ ordinal numeric. Atribút uvedeného typu môže nadobúdať číselné hodnoty z veľkého rozsahu a môže mať niekoľko číslic za desatinou čiarkou, pričom v databáze je uložená informácia o tom, ktorý atribút má koľko desatinných miest, označená ako presnosť. Pri implementácií prezentovanej funkcie sme využili informáciu, ktorá hovorí o presnosti hodnoty atribútu.

Pseudokód, ktorý charakterizuje implementáciu uvedenej funkcie pre ordinal numeric typ atribútu :

vráťPodobnosť(presnosť x, hodnota1, hodnota2) {

Zaokrúhlime vstupné hodnoty na x desatinných miest - dostaneme čísla A a B.

ak A=B, vrátime 1

ak (10^x)*|A-B|=1 vrátime 0.9

ak (10^x)*|A-B| < 10 vrátime 0.5

inak vrátime 0}

Základom porovnávacej funkcie je metóda, ktorá má na vstupe tri hodnoty. Prvá hodnota vyjadruje presnosť s akou atribút nadobúda jednotlivé hodnoty. Zvyšné dve

vstupné hodnoty reprezentujú dvojicu hodnôt atribútu pre porovnanie. Postup, ktorým metóda počíta podobnosť medzi dvoma hodnotami spočíva v overení troch podmienok.

V prvej podmienke sa overí či zaokrúhlené vstupné hodnoty sú zhodné. Ak áno podobnosť je jedna. V druhej podmienke sa overí, či zaokrúhlené hodnoty na základe danej presnosti sa líšia len o jednotku na príslušnom mieste za desatinou čiarkou. Ak je splnená uvedená podmienka tak podobnosť je 0.9. Posledná možnosť, kedy výsledkom bude nenulová podobnosť, je splnenie tretej podmienky. Tá overí, či vstupné hodnoty sú si veľmi blízke, alebo zhodné pre prípad zaokrúhlenia o jedno desatinné miesto menej. V prípade splnenia uvedenej podmienky je podobnosť určená číslom 0.5. Vo všetkých ostatných prípadoch je podobnosť rovná 0.

Predpokladáme, že implementácia prezentovanej porovnávacej funkcie pre ordinal numeric typ atribútov povedie k presnejším celkovým výsledkom procesu unifikácie. Hlavne preto, že v spomínanom balíku porovnávacích funkcií nebola žiadna funkcia, ktorá by pri vyhodnocovaní podobnosti atribútov typu ordinal numeric brala do úvahy aj spomínanú presnosť hodnôt atribútu.

5 Unifikačný algoritmus

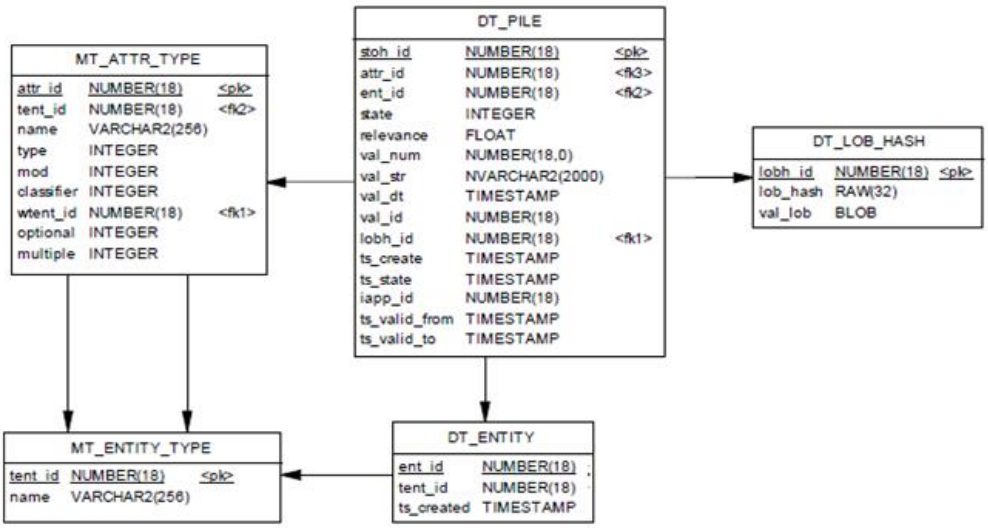
Hlavným cieľom našej diplomovej práce bolo navrhnúť algoritmus unifikácie produktov rovnakého druhu z viacerých zdrojov podľa atribútov. Pri návrhu sme sa snažili postupovať podľa jednotlivých krokov všeobecného modelu unifikácie, popísaných v kapitole 2 (Unifikácia vo všeobecnosti). Postup vykonávania niektorých krokov sme prispôsobili zadaniu za účelom dosiahnutia lepších výsledkov a efektívnosti algoritmu. Algoritmus unifikácie sme navrhli primárne pre unifikáciu produktov z produktových katalógov v projekte Kapsa, aj keď v prípade dodržania vlastností vstupu by mohol byť použitý aj pre unifikáciu iných entít. Tento algoritmus sme overili a otestovali nad doménou pneumatík.

5.1 Existujúce riešenie v stohovom systéme

Jedným z hlavných zdrojov bol článok [1,2], v ktorom autori riešia problematiku unifikácie nad stohovým systémom. Stohový systém je štruktúrovaný systém, ktorý umožňuje prepojenie medzi nesúvisiacimi aplikáciami a výmenu údajov medzi nimi. Prezentovaný stohový systém bol navrhnutý a použitý nad reálnou doménou, ktorá sa týkala personálnych informácií. Základnou ideou prezentovaného stohového systému je vertikalizácia dát. Uvedená metóda predstavuje inovatívnu metódu ukladania informácií o objektoch. Veľmi podobnú štruktúru máme v našej tabuľke WRAPDATA (kapitola 3.2 Štruktúra tabuľky WRAPDATA).

5.1.1 Vertikalizácia dát

Metóda vertikalizácie dát sa líši od klasických databázových princípov, ktoré sa využívajú na ukladanie dát. Hlavným rozdielom je skutočnosť, že sa nevyužíva tradičné horizontálne poňatie databázovej tabuľky, kde jeden riadok predstavuje nejakú množinu spolu súvisiacich atribútov nejakej entity. Namiesto toho je každý atribút reprezentovaný jedným riadkom dátového stohu a odpovedajúce atribúty sú následne spojené identifikátorom entity, ktorá je reprezentovaná uvedenými atribútmi. Celá dátová schéma je tvorená dvoma základnými tabuľkami : hodnoty atribútov (tabuľka DT_PILE na obrázku 7) a štruktúra entít (DT_ENTITY na obrázku 7). Okrem nich sa v štruktúre stohového systému nachádzajú aj menšie pomocné metatabuľky (napríklad MT_ATTR_TYPE, MT_ENTITY_TYPE, DT_LOB_HASH na obrázku 7), ktoré obsahujú potrebné metadáta.



Obr. 7 Základná schéma stohového systému podľa [1]

5.1.2 Unifikácia v stohovom systéme

Hlavným cieľom unifikácie je nájsť už existujúcu entitu alebo vytvoriť novú entitu v stohovom systéme.

Pomocou nasledujúceho príkladu autori článku [1] popísali, prečo je potrebná unifikácia a návrh algoritmu unifikácie v stohovom systéme. Popísali situáciu s ktorou sa môžeme stretnúť pri spracovaní rovnakých dát v rôznych aplikáciách. Nech aplikácia A1 má záznam o osobe s menom "Jana", priezvisko "Teskova" s rodným číslom napr. "806010/7000" a adresou : "Matkin dom č 10". Rovnaké informácie sú uložené v aplikácii A2 tým istým spôsobom. Po svadbe si Jana Teskova vzala manželovo priezvisko (ako je bežné). Takže jej priezvisko sa zmení na "Štanclová". Tiež odišla bývať so svojím novým manželom na adresu "Nový domov 20". Naša osoba upovedomí o svojom manželstve úrad, ktorý vykoná sprievodné zmeny iba pomocou kancelárskej aplikácie A1, a nevykonajú sa v inom úrade prostredníctvom aplikácie A2. Na prvý pohľad nemusí byť zrejmé, že A2 nezdieľa dáta s A1, pričom sú obidva používané na uchovanie dát o osobách v jednej organizácií. Druhá možnosť je, že A1 a A2 sú integrované dohromady, takže zmeny v A1 sa automaticky tiež rozdistribuujú na A2 (ale to je tzv. distribučný problém). Takže výsledok je, že A1 a A2 ukladajú rôzne údaje o jednej inštancii entity. Čo sa stane, keď sa snažíme o zlúčenie dát z A1 a A2 do spoločného úložiska dát?

Ako tento príklad ukazuje, takmer všetky atribúty sa zmenili. Ale niektoré z nich sú konštantné, a to najmä osobné identifikačné číslo, ktoré by malo byť unikátne. Bohužiaľ existujú prípady, keď rôzne osoby majú rovnaké rodné číslo. Na druhej strane, tieto prípady sú zriedkavé. V tomto príklade sa niektoré atribúty zmenili, ale kombinácia niektorých atribútov môže mať značný význam: napr. meno a priezvisko dohromady tvoria celý názov. Rovnosť týchto atribútov vyjadruje určitú pravdepodobnosť, že uvedené dva záznamy popisujú jednu osobu.

Prezentovaný príklad, ale aj ďalšie pozorovania viedli autorov tohto článku ku klasifikácii atribútov. Každý atribút bol klasifikovaný do jednej z troch nasledujúcich tried: určujúce, relevantné, ostatné.

- Určujúce - identifikuje inštanciu entity s veľmi vysokou pravdepodobnosťou (napr. rodné číslo, číslo pasu a podobne).
- Relevantné - významný atribút, ktorý pomáha identifikovať jednoznačne rovnosti subjektov (napr. typy atribútov "meno" a "priezvisko" pre entity typu "osoba").
- Ostatné - nemajú žiadny vplyv na zhodu entít.

Nasleduje idea algoritmu unifikácie entít pre dve inštancie v stohovom systéme založenom na relevantnosti atribútov :

1. Všetky určujúce a relevantné hodnoty atribútov sú si rovné - *úplne jasná zhoda s veľmi vysokou pravdepodobnosťou.*
2. Neprázdna podmnožina určujúcich a neprázdna podmnožina relevantných hodnôt atribútov sú rovnaké, zostávajúce určujúce a relevantné hodnoty atribútov nemajú ekvivalent v novej inštancií entity - *zhoda s pomerne vysokou pravdepodobnosťou*
3. Neprázdna podmnožina určujúcich hodnôt atribútov je rovná, zostávajúce určujúce hodnoty atribútovú nemajú ekvivalent v druhej inštancií entity, ale nejaká neprázdna podmnožina relevantných hodnôt atribútov sa líši - *pravdepodobnosť zhody určujúcich hodnôt atribútov prevažuje nad pravdepodobnosťou odlišných relevantných hodnôt atribútov, teda inštancie entity sú považované za potenciálne zhodné a správca systému je oboznámený.*
4. Neprázdna podmnožina určujúcich hodnôt atribútov sa líši, zvyšné určujúce hodnoty atribútov sú ekvivalentné, nejaká neprázdna podmnožina relevantných

hodnôt atribútov je rovnaká, ostatné relevantné hodnoty atribútov nemajú ekvivalent. Tento prípad je riešený ako vyššie – *inštancie entity sú považované za potenciálne zhodné a správca systému je oboznámený*.

5. Vo všetkých ostatných prípadoch - *sú inštancie entít rôzne s veľmi vysokou pravdepodobnosťou*.

Uvedená unifikácia na základe relevantnosti atribútov entity nie je postačujúca pre naše zadanie. V porovnaní s uvedeným algoritmom aplikovaným v stohu, sme v našom návrhu použili modifikovanú relevantnosť atribútov (kapitola 5.2.1 Relevantnosť atribútov) a zdrojovú závislosť.

5.2 Reprezentácia produktov v úložisku

V tejto podkapitole popíšeme vstupné dáta pre proces unifikácie. Základný vstup predstavujú v našom prípade dve tabuľky : WRAPJOINEDDATA a JOINEDDATA. Prvá z nich už bolo charakterizovaná v kapitole 3.2 Štruktúra tabuľky WRAPDATA, preto si ju len pripomenieme. Druhá uvedená tabuľka bude podrobnejšie opísaná v kapitole 5.2.2 Číselníky a JOINEDDATA tabuľka, najskôr však popíšeme relevantnosť atribútov, ktorú sme si jemne upravili oproti relevantnosti spomínanej v predchádzajúcej kapitole.

5.2.1 Relevantnosť atribútov

Pri návrhu algoritmu unifikácie produktov podľa atribútov sme sa inšpirovali vyššie prezentovaným riešením unifikácie v rámci stohového systému. Rozhodli sme sa zapracovať práve spomínanú relevantnosť atribútov aj do nášho riešenia v kombinácií s ďalšími komponentami. V porovnaní s personálnymi informáciami, z ktorých sa niektoré v čase menia, v doméne produktov je šanca na zmenu hodnoty nejakého určujúceho alebo relevantného atribútu oveľa menšia (napr. zmena grafickej karty v konkrétnom notebooku, alebo zmena dezénu pri konkrétnej pneumatike). Preto si môžeme dovoliť byť prísnejší pri nezhode atribútov.

Relevantnosť atribútov predstavuje akúsi dôležitosť atribútu, ktorá má vplyv na poradie vyhodnocovania atribútov v procese unifikácie. Rozhodli sme sa, že atribúty na základe relevantnosti budú rovnako začlenené do troch skupín, konkrétne na : *určujúce, relevantné* a *ostatné*.

- *Určujúce* - zhoda takto označených atribútov znamená s veľkou pravdepodobnosťou, že ide o rovnakú entitu. Naopak rozdielnosť hodnôt

nejakého určujúceho atribútu znamená s veľkou pravdepodobnosťou, že ide o rozdielne entity

- *Relevantné* – rozdielnosť hodnôt relevantných atribútov výrazne zvyšuje pravdepodobnosť rôznosti entít.
- *Ostatné* – zvyšujú pravdepodobnosť zhody. Užitočné sú vtedy, ak je málo porovnateľných relevantných a žiadne určujúce atribúty.

Ďalej sme sa rozhodli pre každý atribút, nezávisle od relevantnosti atribútu uviesť jeho *zdrojovú závislosť*. Táto vlastnosť plní dôležitú úlohu v kombinácií s predtým prezentovanou relevantnosťou, v procese unifikácie, hlavne čo sa týka určujúcich a relevantných atribútov. Platí, že zdrojovo závislé určujúce a relevantné atribúty má zmysel porovnávať, iba ak ide o porovnanie produktov z toho istého zdroja. Typicky môže ísť o inkrementálny identifikátor produktov pochádzajúcich z databázy zdroja, alebo o URL produktu.

Metadáta atribútu, ako sú relevantnosť a zdrojová závislosť sú uložené v tabuľke v databáze. Okrem týchto hodnôt spomenutá tabuľka obsahuje názvy a identifikátory typov atribútov (kapitola 3.3 Štruktúra tabuľky WRAPJOINEDDATA). Ďalej sa tam nachádza aj stĺpec, ktorého hodnoty reprezentujú priradenie konkrétnej porovnávacej funkcie pre jednotlivé atribúty. Na základe tejto informácie sa vie, akú porovnávaciu funkciu treba použiť v kroku porovnania v procese unifikácie pre hodnoty daného atribútu.

5.2.2 Číselníky a JOINEDDATA tabuľka

Pred samotným procesom unifikácie je potrebné pripraviť si vstupné dáta. Tento proces sa realizuje ako súčasť predprípravy dát v rámci procesu unifikácie. Časť dát, ktoré sú na vstupe unifikácie, sme už popísali v tretej kapitole. Informácie o produktoch získavame z tabuľky WRAPDATA, ktorá slúži na korektné zjednotenie nawrapovaných dát do zvoleného štandardného formátu. Dáta, uložené v tabuľke WRAPDATA v navrhnutom formáte sa skonvertujú do tabuľky WRAPJOINEDDATA. V procese preklápania dát z tabuľky WRAPDATA do tabuľky WRAPJOINEDDATA sú jednotlivé hodnoty atribútov konvertované z reťazcov na príslušný dátový typ, ktorý je určený pre každý atribút. Naplnená tabuľka WRAPJOINEDDATA je jedným zo vstupov pre proces unifikácie.

Ďalšiu časť vstupných dát pre unifikáciu predstavuje tabuľka JOINEDDATA. Pre každú doménu vytvárame samostatnú tabuľku JOINEDDATA. Máme k dispozícií

niekoľko domén teda v databáze je viac takýchto tabuliek. Štruktúra tejto tabuľky je veľmi podobná WRAPJOINEDDATA tabuľke. Stĺpce, ktoré ju tvoria reprezentujú jednotlivé atribúty produktu. Okrem nich obsahuje aj ďalší stĺpec, a to id_source (identifikátor zdroja).

Základný rozdiel medzi oboma tabuľkami je, že tabuľka JOINEDDATA obsahuje produkty uložené v úložisku, ktoré sú považované za jedinečné. Cieľom je na základe výsledkov unifikácie uvedenú tabuľku rozšíriť o nové produkty, ktoré sa nám nepodarí zunifikovať.

Medzi vstupné dáta patria aj číselníky. Číselník je tabuľka, ktorá ukladá hodnoty niektorých atribútov a umožňuje ich reprezentáciu vo vstupných tabuľkách (WRAPJOINEDDATA a JOINEDDATA) prostredníctvom číselných identifikátorov. Číselníky vytvárame pre typy atribútov nominal, hierarchical a discrete. Číselník obsahuje stĺpec s názvom temporary. Na základe hodnôt uvedeného stĺpca vieme povedať, ktorá hodnota je nová, teda nachádza sa len vo WRAPJOINEDDATA tabuľke, a ktorá hodnota je stará, teda môže sa nachádzať aj v JOINEDDATA tabuľke uloženej v databáze. Uvedená skutočnosť poskytuje informácie pomocou ktorých, vieme vytvoriť pomocné tabuľky.

5.2.3 Pomocné tabuľky

Tabuľky JOINEDDATA a WRAPJOINEDDATA obsahujú v každom z ich riadkov okrem iného, hlavne hodnoty atribútov produktov. Tieto hodnoty môžu byť reprezentované skutočnou hodnotou (napr. cena 12,53 eura ako číslo 12,53) alebo identifikátorom z číselníka vymenovaných hodnôt (napr. namiesto výrobcu „Michelin“ sa uvedie jeho identifikátor 22). Pomocné tabuľky sa používajú v procese unifikácie na porovnávanie hodnôt atribútov reprezentovaných identifikátormi z číselníkov. Keďže atribúty s vymenovanými hodnotami majú obvykle malú doménu, dochádza pri unifikácii k častému porovnaniu rovnakých dvojíc identifikátorov. Porovnanie cez porovnávacie funkcie je časovo náročné, preto sme sa rozhodli predpočítať pre každú zmysluplnú dvojicu identifikátorov podobnosť hodnôt, ktoré reprezentujú.

Pomocné tabuľky vytvárame pre tri typy atribútov, ktoré sú označené ako nominal, hierarchical alebo discrete na základe ich číselníkov. Postup vytvorenia pomocnej tabuľky pre uvedené tri typy atribútov je nasledovný.

Ako bolo už spomenuté v predchádzajúcej podkapitole, číselníky obsahujú stĺpec s názvom temporary. Prostredníctvom hodnôt uvedeného stĺpca máme informáciu o tom, ktorá hodnota je nová, a ktorá je stará. V pomocných tabuľkách ukladáme

podobnosť medzi novými a starými hodnotami atribútov. Staré atribúty navzájom neporovnávame, lebo predpokladáme, že sú rôzne a teda ich podobnosť pre účely unifikácie je nula.

Podobnosť vymenovaných hodnôt uvažujeme najmä kvôli možným preklepom v hodnotách nových zdrojov. Do pomocnej tabuľky sa ukladajú objekty v tvare starý identifikátor, nový identifikátor a podobnosť vypočítaná pomocou príslušnej porovnávacej funkcie z dvojice hodnôt na ktorú odkazujú uvedené identifikátory. Tabuľka č.1 znázorňuje príklad pomocnej tabuľky.

Tab. 1 Príklad pomocnej tabuľky

Id_value_old	Id_value_new	Similarity
1	7	0.5
2	7	0.7
3	7	0.25
4	7	0.4
1	8	0.3
2	8	0.62
3	8	0.5
4	8	0.36

Po ukončení procesu unifikácie je stĺpec temporary pre použité hodnoty nominal, hierarchical a discrete aktualizovaný. Konkrétne, hodnoty, ktoré sa vyskytli v produktoch klasifikovaných ako nové, a v stĺpci temporary sú označené ako dočasné/nové sa aktualizujú na trvalé/staré. Ostatné sú prepojené so starými hodnotami ako synonymá na základe predpokladanej zhody hodnôt atribútu zunifikovaných objektov. Pre doménovo závisle atribúty neexistuje predpoklad na vytváranie číselníkov a tým pádom ani nemá význam uvažovať o označovaní synoným..

5.3 Popis algoritmu

Na začiatku tejto kapitoly pomocou pseudokódu popíšeme, ako sa počíta podobnosť hodnôt atribútov dvoch produktov. V ďalšej časti podobne charakterizujeme zjednodušený algoritmus unifikácie produktov podľa atribútov. Zjednodušený v zmysle, že krok klasifikácie prebieha len na základe podobnosti hodnôt atribútov, určených sumovanou podobnosťou (SimSum). Potom popíšeme kompletný algoritmus unifikácie, a v závere tejto kapitoly uvedieme niekoľko príkladov unifikácie.

5.3.1 Výpočet podobnosti hodnôt atribútov

Pseudokód :

```
porovnaj(atribút A, produkt X, produkt Y) {
    ak A je typu string (reťazec) a obe hodnoty A produktov X a Y nie sú null
        vráť podobnosť vypočítanú určenou porovnávacou funkciou;
    ak A je typu nominal, hierarchical, discrete a identifikátory A produktov X a Y nie sú null
        vráť podobnosť určenú z pomocných tabuliek, alebo 0, ak taká dvojica identifikátorov
        v pomocných tabuľkách nie je;
    ak A je typu ordinal numeric a obe hodnoty A produktov X a Y nie sú null
        vráť podobnosť vypočítanú porovnávacou funkciou pre typ ordinal numeric;
    inak vráť podobnosť = 1;
}
```

5.3.2 Diskusia o výpočte podobnosti hodnôt atribútov

Proces unifikácie sa realizuje na základe hodnôt podobnosti dvoch objektov. V našom algoritme sa podobnosť hodnôt atribútov počíta na základe nami navrhnutej funkcie označenej *porovnaj(A,X,Y).* Na vstupe uvedenej funkcie sú dva produkty X a Y a atribút A, ktorého podobnosť hodnôt uvedená funkcia počíta.

V prvom prípade, ak atribút A je typu string(reťazec) podobnosť dvoch hodnôt atribútu A produktov X a Y je vypočítaná porovnávacou funkciou. V prípade uvedeného typu ako porovnávaciu funkciu sme zvolili funkciu Levenshtein (kapitola 4.1). Pre použitie uvedenej funkcie sme sa rozhodli preto, lebo hodnoty atribútov typu string v našej doméne sú spravidla krátke reťazce. A práve uvedená funkcia Levenshtein je vhodná na počítanie podobnosti medzi krátkymi reťazcami.

V ďalšom prípade sa využívajú predpočítané hodnoty z pomocných tabuliek(kapitola 5.2.3 Pomocné tabuľky). Ak sa v konkrétnej pomocnej tabuľke pre jeden z troch typov atribútu A nachádza dvojica identifikátorov hodnôt, tak sa vráti odpovedajúca predpočítaná podobnosť. Keďže skutočné hodnoty atribútov uvedených troch typov sú tiež spravidla krátke reťazce, na určenie predpočítanej podobnosti sme použili spomenutú funkciu Levenshtein. V prípade, že dvojica identifikátorov sa v tabuľke nenachádza, podobnosť je 0, lebo staré hodnoty považujeme za rôzne.

Napokon v prípade, keď atribút A je typu ordinal numeric, sa na výpočet podobnosti hodnôt atribútu A použije nami navrhnutá porovnávacia funkcia pre tento typ. Uvedená funkcia využíva vlastnosť atribútov, ktorá hovorí o presnosti, s akou nadobúda atribút A hodnoty.

Informácia o spomínanej presnosti, a tiež o tom, pre aký atribút A sa má použiť ktorá porovnávacia funkcia, sú zadané administrátorom systému v metadátach.

V prezentovanej funkcií *porovnaj(A,X,Y),* pri výpočte podobnosti hodnôt atribútu dvoch produktov je zahrnuté aj porovnanie s null hodnotami. Teda, ak aspoň jedna z hodnôt atribútu A je null, podobnosť je jedna, bez ohľadu na to, akého typu je daný atribút A. Myšlienkou tohto prístupu je to, že samotnú funkciu porovnania hodnôt berieme skôr ako penalizačnú, teda ak odhalí rozdielnosť, podobnosť sa znižuje, ale ak porovnanie nie je možné kvôli nevyplneným hodnotám, penalizácia nenastáva.

5.3.3 Zjednodušený algoritmus unifikácie

Pseudokód :

klasifikácia na základe SimSum (produkt X, produkt Y***)***
sum = 0;
pre každý atribút A domény {
    ak A je *zdrojovo závislý určujúci* alebo *zdrojovo závislý relevantný,* a zároveň X a Y nie sú z rovnakého zdroja
        sum = sum + 1;
    inak
        sum = sum + **porovnaj(**A, X, Y**);**
}
SimSum = sum / počet atribútov domény;
ak SimSum ≥ $h_{dolná}$
        vráť *zhodné*
ak SimSum < $h_{dolná}$ a SimSum > $h_{horná}$
        vráť *potenciálne zhodné*
inak vráť *rôzne*

5.3.4 Diskusia o zjednodušenom algoritme unifikácie

Základom vstupu zjednodušeného algoritmu unifikácie je produkt X z tabuľky JOINEDDATA uloženej v databáze, a druhý produkt Y uložený v tabuľke WRAPJOINEDDATA.

Algoritmus pracuje nasledovne: pre dvojicu produktov X a Y na vstupe sa postupne berie každý atribút A zo zoznamu atribútov. V ďalšom kroku na základe zdrojovej závislosti v spojení s relevantnosťou atribútu A sa počíta sumovaná podobnosť(SimSum).

Ak atribút A je *zdrojovo závislý určujúci* alebo A je *zdrojovo závislý relevantný*, a zároveň produkty X a Y nie sú z rovnakého zdroja, potom sa neporovnávajú. Predpokladáme, že hodnoty takých atribútov sú veľmi rozdielne.

V každom inom prípade sa k hodnote premennej *sum* pripočíta podobnosť hodnôt atribútu A, vypočítaná pomocou funkcie *porovnaj(A,X,Y)* (kapitola 5.3.1 Výpočet podobnosti hodnôt atribútov). V prípade *zdrojovo závislých ostatných* napr. atribút cena, prebieha porovnanie. Pre dva produkty z dvoch rôznych zdrojov, totiž cena nebude veľmi rozdielna v prípade, že ide o dvojicu totožných produktov. Preto má zmysel pripočítať podobnosť uvedených atribútov do sumovanej podobnosti.

V záverečnej časti algoritmu sa vykoná klasifikácia produktov X a Y na základe hodnoty SimSum. Uvedená SimSum je daná podielom hodnoty premennej *sum* a počtu atribútov danej domény. Samotná klasifikácia, potom spočíva v overení stanovených pravidiel. Teda ak hodnota SimSum je menšia ako dolná hranica ($h_{dolná}$ – vyjadruje dolné ohraničenie hodnoty SimSum pre klasifikáciu na zhodné produkty), potom klasifikačný status produktov X a Y je *zhodné*. Ak hodnota SimSum je menšia ako dolná hranica, a zároveň väčšia ako horná hranica ($h_{horná}$ - pod ňou sú len nové produkty), potom klasifikačný status produktov X a Y je *potenciálne zhodné*. Inak klasifikačný status produktov X a Y je *rôzne*. Hodnoty $h_{dolná}$ a $h_{horná}$ je potrebné určiť experimentálne, čo opisujeme v kapitole 6.1 Kvalitatívne testy.

5.3.5 Algoritmus unifikácie

Pseudokód :

```
1.  pre každý produkt X z objektov domény v úložisku v tabuľke JOINEDDATA {
2.   pre každý doteraz neklasifikovaný objekt Y v tabuľke WRAPJOINEDDATA {
3.    pre každý určujúci atribút A domény {
4.     ak produkty X alebo Y nemajú vyplnený atribút A
5.       choď na ďalší atribút (r. 3);
6.     ak A je zdrojovo závislý a objekty X a Y sú z rôzneho zdroja
7.       choď na ďalší atribút (r. 3);
8.     ak porovnaj(A, X, Y) == 1 {
9.        ZHODNE = ZHODNE ∪ {<X,Y>};
10.       POTENCIALNE = POTENCIALNE – {∀<a,Y,p>∈ POTENCIALNE, kde a,p sú ľubovoľné};
11.       choď na ďalší objekt X (r.1) ;
12.    } inak vezmi ďalší objekt Y (r.2) ;
13.   } // r.3
14.   pre každý relevantný atribút A {
15.    ak produkty X a Y nemajú vyplnený atribút A
16.      choď na ďalší atribút (r. 14);
17.    ak A je zdrojovo závislý a objekty X a Y sú z rôzneho zdroja
18.      choď na ďalší atribút (r.14);
19.    ak porovnaj(A, X, Y) < 1
20.      choď na ďalší objekt Y (r.2);
21.  } // r.14
22.  switch(klasifikácia na základe SimSum ){
23.     zhodné : ZHODNE = ZHODNE ∪ {<X,Y>};
```

24. POTENCIALNE = POTENCIALNE – {∀<a,Y,p>∈ POTENCIALNE, kde a,p sú ľubovoľné};
25. **choď** na ďalší objekt X (r.1);
26. ***potenciálne zhodné*** : POTENCIALNE = POTENCIALNE ∪ {<X,Y,Simsum>};
27. } // r.22
28. } // r. 2
29. } // r.1
30. **vlož** každý neklasifikovaný objekt z tabuľky WRAPJOINEDDATA do množiny NOVE;
31. **vráť** ZHODNE, POTENCIALNE, NOVE, zoznam dočasných a zoznam ekvivalentných identifikátorov.

5.3.6 Diskusia o algoritme unifikácie

Prvý cyklus postupne (po jednom) načítava jednotlivé produkty X z JOINEDDATA tabuľky, ktorá je uložená v databáze. Produkt X sa postupne porovnáva so všetkými spracovávanými objektmi Y z WRAPJOINEDDATA tabuľky. Algoritmus unifikácie na základe zatriedenia atribútov do skupín relevantnosti (kapitola 5.2.1 Relevantnosť atribútov) môžeme rozdeliť na tri moduly.

Základom prvého modulu je cyklus cez *určujúce* atribúty A. V prípade, že produkty X alebo Y nemajú vyplnený *určujúci* atribút A, vezme sa ďalší atribút A. V ďalšej časti sa overí, či A je zdrojovo závislý a produkty X a Y sú z rovnakého zdroja. Pokiaľ by nebola splnená uvedená podmienka, tak nemá zmysel tieto hodnoty porovnávať (napríklad ide o inkrementálny identifikátor produktu z databázy konkrétneho zdroja). Ak má zmysel porovnať hodnoty, vypočíta sa podobnosť hodnôt atribútu A pre X a Y. Výpočet podobnosti sa realizuje pomocou funkcie *porovnaj(A,X,Y)* (kapitola 5.3.1 Výpočet podobnosti hodnôt atribútov). Ak vypočítaná podobnosť uvedených hodnôt atribútu A je 1, produkty X a Y sú vložené do množiny **ZHODNE** a z množiny **POTENCIALNE** sa odstránia všetky dvojice produktov obsahujúce produkt Y. Pretože ak produkt Y bol označený ako zhodný s nejakým produktom X, teda ide o tie isté produkty, nemá význam uvažovať potenciálnu zhodu s inými produktmi z tabuľky JOINEDDATA, kde sú všetky produkty považované za navzájom rôzne. V opačnom prípade, teda ak vypočítaná podobnosť je menšia ako 1, považujeme produkty X a Y za rôzne a vezme sa ďalší produkt Y na porovnanie.

Druhý modul realizuje cyklus pre *relevantné* atribúty A. Rovnako ako v prvom module, aj tu sa najskôr overí, či produkty X a Y nemajú vyplnený *relevantný* atribút A, a druhá podmienka, ak A je *zdrojovo závislý* či objekty X a Y sú z rovnakého zdroja. V prípade splnenia uvedených podmienok sa vezme ďalší atribút A. Potom sa realizuje výpočet podobnosti hodnôt atribútu A pre X a Y pomocou vyššie popísanej funkcie

(kapitola 5.3.1 Výpočet podobnosti hodnôt atribútov). Ak vypočítaná podobnosť je menšia ako 1, považujeme produkty X a Y za rôzne a vezme sa ďalší produkt Y.

Napokon, ak predchádzajúce moduly neklasifikovali produkty X a Y za zhodné alebo rôzne, je realizovaný tretí modul, ktorého jadro tvorí *klasifikácia na základe SimSum* (kapitola 5.3.3 Zjednodušený algoritmus unifikácie). V prípade, že táto metóda vráti pre práve spracovávané produkty X a Y klasifikačný status *zhodné*, produkty X a Y sú vložené do množiny **ZHODNE**, z množiny **POTENCIALNE** sa odstránia všetky dvojice produktov obsahujúce Y a prejde sa na ďalší produkt X. Ak spomínaná metóda vráti klasifikačný status *potenciálne zhodné*, dvojica práve spracovávaných produktov X a Y je vložená do množiny **POTENCIALNE** spolu s hodnotou SimSum vypočítanou v klasifikácii na základe SimSum.

Po ukončení hlavného cyklu prechádzajúceho cez všetky produkty X z JOINEDDATA, sú všetky neklasifikované produkty Y z WRAPJOINEDDATA vložené do množiny **NOVE**.

Výsledkom algoritmu unifikácie sú : **ZHODNE**, **POTENCIALNE**, **NOVE**, zoznam dočasných a zoznam ekvivalentných identifikátorov.

Množina **ZHODNE** obsahuje dvojice totožných produktov, kde jeden produkt je z WRAPJOINEDDATA tabuľky a druhý z JOINEDDATA tabuľky. Teda ekvivalentné produkty už máme uložené v JOINEDDATA tabuľke. Množina **NOVE** obsahuje produkty, ktoré sa nám nepodarilo zunifikovať a budú vložené do tabuľky JOINEDDATA. Napokon množinu **POTENCIALNE** tvoria objekty v tvare: identifikátor produktu z WRAPJOINEDDATA tabuľky, identifikátor produktu z JOINEDDATA tabuľky a hodnotu, ktorá vyjadruje percentuálnu zhodu oboch produktov. Tento zoznam musí byť ešte spracovaný buď manuálne administrátorom systému, alebo prostredníctvom nejakého nástroja, ktorý dokáže jednoznačne rozhodnúť, či ide o nový produkt, alebo o dvojicu zunifikovaných produktov.

Tvorbu zoznamov dočasných a ekvivalentných identifikátorov sme pre jednoduchosť do pseudokódu neuvádzali. Zoznam ekvivalentných identifikátorov obsahuje niektoré identifikátory z číselníkov pre nominal, hierarchical a discrete typy atribútov. Do tohto zoznamu sa vkladajú identifikátory hodnôt atribútov z číselníkov pre produkt X a produkt Y ak sú splnené nasledujúce podmienky :

- podobnosť dvoch hodnôt atribútu A pre X a Y je určená z pomocných tabuliek,
- produkty X a Y sú klasifikované ako zhodné,
- skutočné hodnoty atribútu A sa nezhodujú.

Do zoznamu dočasných identifikátorov sú vložené identifikátory nominal, hierarchical a discrete hodnôt atribútov pre produkt Y pri splnení týchto podmienok: Uvedené identifikátory sú označené z hľadiska temporárnosti ako nové/dočasné a daný produkt Y je klasifikovaný ako nový.

Po skončení algoritmu unifikácie sa vykoná metóda, ktorej vstupom je zoznam ekvivalentných a dočasných identifikátorov. Hlavnou úlohou tejto metódy je vykonať update príslušných tabuliek na základe obsahu oboch zoznamov, a zabezpečiť tým aktuálnosť a presnosť dát v JOINEDDATA tabuľke. Všetky použité temporárne hodnoty sú na základe týchto tabuliek označené za netemporárne nové (ak sa ich identifikátory nachádzajú v zozname dočasných identifikátorov), alebo za netemporárne a ekvivalentné s inými (ak sa ich identifikátory nachádzajú v zozname ekvivalentných identifikátorov).

V navrhnutom a implementovanom algoritme unifikácie produktov podľa atribútov sme zakomponovali nové prvky, ako relevantnosť atribútov, zdrojovú závislosť, temporárnosť, ekvivalentnosť a pomocné tabuľky. Niektoré vstupy, ktoré sú súčasťou preprocesingu algoritmu unifikácie ešte nie sú plne automatizované. Napríklad atribúty jednotlivých domén sú klasifikované do jednej z troch skupín relevantnosti manuálne. Možnosť automatizácie tvorby týchto vstupov je otázna a môže byť súčasťou možnej ďalšej práce.

5.4 Konkrétne príklady unifikácie

Navrhli sme niekoľko príkladov unifikácie nad reálnou doménou produktov. K dispozícii sme mali dáta získané wrapovaním z dvoch nezávislých obchodov s pneumatikami. Jednotlivé príklady preto boli vytvorené nad touto doménou. Konkrétna pneumatika bola reprezentovaná tridsiatimi troma atribútmi. Dáta neboli kompletne vyplnené, inými slovami niektoré hodnoty atribútov boli null.

Nakoľko 33 atribútov je pomerne dosť, budeme kvôli prehľadnosti používať jednotnú formu príkladov. Každý nižšie uvedený príklad začína popisom. Za ním sa nachádza konkrétny príklad dvoch pneumatík. Na ľavej strane je vždy pneumatika z JOINEDDATA tabuľky na pravej z WRAPJOINEDDATA tabuľky. Keďže jedna pneumatika má veľa atribútov, rozhodli sme sa zobraziť len podstatné atribúty pre daný príklad. Jednotlivé atribúty sú na základe relevantnosti (kapitola 5.2) v každom príklade farebne rozlíšené. Určujúce atribúty sú znázornené červeným písmom, relevantné zeleným a ostatné čiernym písmom.

V prvom prípade sme vybrali pneumatiku (Pneu 1) z nawrapovaných dát. Tá mala vyplnený atribút Ean, ktorý je z hľadiska relevantnosti zatriedený do skupiny určujúcich zdrojovo závislých atribútov. V JOINEDDATA tabuľke bola uložená pneumatika s rovnakou hodnotou uvedeného atribútu. Algoritmus korektne zunifikoval obe pneumatiky bez potreby počítania sumovanej podobnosti (SimSum). Tento test poukázal na jednu z výhod navrhnutého algoritmu, a to, pokiaľ produkt má korektné vyplnené hodnoty určujúcich atribútov, tak odpadá potreba počítania sumovanej podobnosti v procese unifikácie a znamená výraznú úsporu času. Tento prípad by mal byť typický pri opätovnom stiahnutí stránok portálu za účelom aktualizácie dát (napr. ceny, dostupnosti a pod.)

EAN	3528705090142
KOD	02940
SIRKA	235.0000
PROFIL	70
...	...
RUNFLAT	208
NOISE	3.64
...	...

EAN	3528705090142
KOD	02940
SIRKA	235.0000
PROFIL	70
...	...
RUNFLAT	208
NOISE	3.64
...	...

V ďalšom prípade sme vybrali pneumatiku (Pneu 2) z nawrapovaných dát, ktorá nemala vyplnený určujúci atribút. Mala však vyplnené niektoré relevantné a ostatné atribúty. Z dôvodu absencie hodnoty určujúceho atribútu, a tiež preto, že algoritmus na základe vyplnených relevantných atribútov neklasifikoval Pneu 2 ako novú, sa realizoval výpočet sumovanej podobnosti. Na základe nej algoritmus označil uvedené pneumatiky za zunifikované. Pri výpočte unifikácie takejto pneumatiky môžu typicky nastať dva prípady. Keďže absentuje hodnota určujúceho atribútu tak je potrebné overiť hodnoty relevantných atribútov. Ak sú známe relevantné atribúty zhodné, produkt nie je klasifikovaný ako nový, a počíta sa sumovaná podobnosť. Ak je aspoň jedna hodnota relevantného atribútu rôzna, produkt je klasifikovaný ako nový a sumovaná podobnosť sa nepočíta.

EAN	3528705090142
KOD	02940
SIRKA	235.0000
PROFIL	70
...	...
RUNFLAT	208
NOISE	3.64
...	...

EAN	NULL
KOD	02940
SIRKA	235.0000
PROFIL	70
...	...
RUNFLAT	208
NOISE	3.64
...	...

V nasledujúcom prípade sme zvolili pneumatiku (Pneu 3) taktiež z nawrapovaných dát. Uvedená pneumatika mala vyplnené len ostatné atribúty. Aby algoritmus dokázal v tomto prípade vykonať unifikáciu a správne klasifikovať, musel sa vykonať výpočet sumovanej podobnosti. Výsledkom bola klasifikácia uvedených pneumatík do skupiny zhodných.

EAN	3528705090142
KOD	02940
SIRKA	235.0000
PROFIL	70
...	...
RUNFLAT	208
NOISE	3.64
...	...

EAN	NULL
KOD	NULL
SIRKA	NULL
PROFIL	NULL
...	...
RUNFLAT	208
NOISE	3.64
...	...

Na základe analýzy dát sme navrhli niekoľko okrajových prípadov, ktoré môžu nastať. Na začiatku sme uvažovali dve totožné pneumatiky, kde by jedna z nich (zo zdroja) mala nekorektne vyplnený určujúci atribút ako dôsledok chyby pri wrapovaní. V takom prípade algoritmus klasifikuje pneumatiku zo zdroja ako novú. Test poukázal na to, že pokiaľ by sa rátala ešte sumovaná podobnosť, tak by bola šanca že daná pneumatika bude klasifikovaná aspoň ako potenciálne zhodná, kde sa po následnom spracovaní môže správne rozhodnúť. Predpokladáme, že uvedený prípad môže nastať s veľmi malou pravdepodobnosťou.

EAN	3528705090142
KOD	02940
SIRKA	235.0000
PROFIL	70
...	...
RUNFLAT	208
NOISE	NULL
...	...

EAN	405064664577
KOD	02940
SIRKA	235.0000
PROFIL	70
...	...
RUNFLAT	208
NOISE	NULL
...	...

Nasledujúci príklad reprezentovaný zobrazenými pneumatikami poukazuje na porovnanie s null hodnotou. Teda v oboch pneumatikách je hodnota zvoleného atribútu nevyplnenú, teda null. V algoritme je zakomponované aj porovnanie s null hodnotami. Výsledky uvedeného porovnania sú akceptované aj do konečnej klasifikácie.

EAN	3528705090142
KOD	02940
SIRKA	235.0000
PROFIL	70
...	...
RUNFLAT	208
NOISE	NULL
...	...

EAN	405064664577
KOD	02940
SIRKA	NULL
PROFIL	70
...	...
RUNFLAT	208
NOISE	NULL
...	...

Na základe uvedených príkladov navrhnutým algoritmom unifikácie môžeme konštatovať niekoľko pozorovaní. Efektivita z pohľadu zložitosti výpočtu algoritmu je vysoká, pokiaľ sa v danej doméne nachádza jeden alebo viac určujúcich atribútov– po pozitívnej klasifikácii už nie je potrebné produkt z tabuľky WRAPJOINEDDATA tabuľky porovnávať s ďalšími produktmi z tabuľky JOINEDDATA , pri negatívnej klasifikácii, nie je potrebné skúmať podobnosť ďalších atribútov danej dvojice produktov. Ak máme v danej doméne vyplnené aspoň relevantné atribúty efektivita algoritmu je tiež vysoká, pretože opäť pri negatívnej klasifikácii, nie je potrebné skúmať podobnosť ďalších atribútov danej dvojice produktov. Týmto eliminujeme drahé počítanie podobností a hodnoty Simsum. Zrýchlenie takéhoto počítania prezentujeme aj v kapitole 6.2

Kvantitatívne testy. Tiež sa ukázalo, že okrajové prípady algoritmus vyhodnocuje korektne v rámci svojich možnosti.

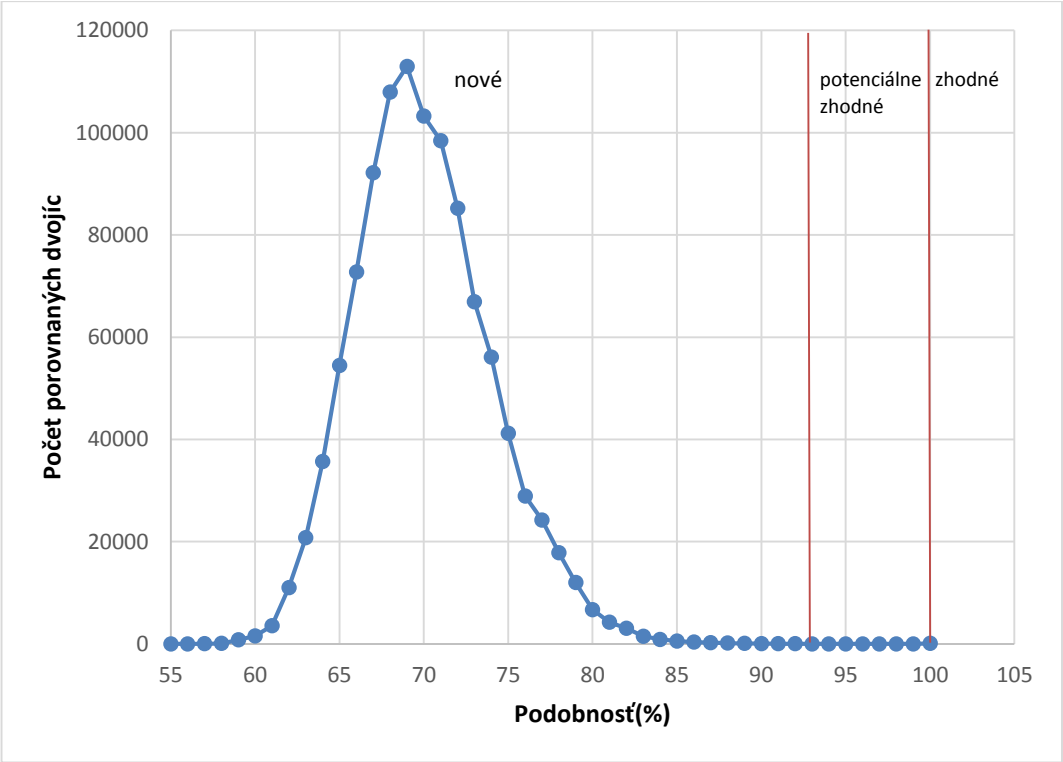
6 Testy

Na otestovanie navrhnutého algoritmu unifikácie sme použili získané/nawrapované dáta, reprezentujúce doménu pneumatík. Celkový počet pneumatík bol 15000. Konkrétna pneumatika bola reprezentovaná tridsiatimi troma atribútmi. Dáta o pneumatikách boli z dvoch rôznych zdrojov. Dáta neboli kompletne vyplnené, inými slovami niektoré hodnoty atribútov boli null. Takéto dáta poskytli prirodzené podmienky pre otestovanie nášho algoritmu unifikácie. Testy sme rozdelili do dvoch skupín. Prvú skupinu tvorili kvalitatívne testy. Ich hlavným cieľom bolo empiricky určiť hranice pre potenciálne zhodné produkty. Druhá skupina s názvom kvantitatívne testy, bola realizovaná s cieľom porovnať unifikačné algoritmy z hľadiska času a úspešnosti vyhodnocovania. Jednoduchý algoritmus unifikácie, ktorý klasifikoval len na základe SimSum s kompletným algoritmom unifikácie (kapitola 5.3.5 Algoritmus unifikácie) nad rôzne veľkou množinou dát.

6.1 Kvalitatívne testy

Hlavným zámerom vykonania kvalitatívnych testov bolo empirické určenie hraníc pre potenciálne zhodné produkty, teda vytvorenie klasifikátora, ktorý klasifikuje produkty do troch tried : zhodné, potenciálne zhodné a rôzne. Potrebujeme určiť hornú hranicu hodnoty SimSum, pod ktorou už s veľmi vysokou pravdepodobnosťou ide o rôzne produkty tak, aby nebola príliš nízka, a zároveň dolnú hranicu hodnoty SimSum, nad ktorou už s veľmi vysokou pravdepodobnosťou ide o zhodné produkty tak, aby nebola zbytočne príliš vysoká. Na určenie spomínaných hraníc sme počítali sumovanú podobnosť medzi všetkými dvojicami pneumatík z tabuliek JOINEDDATA a WRAPJOINEDDATA.

Tieto testy sme realizovali s tabuľkou JOINEDDATA, ktorá obsahovala 100 pneumatík a s WRAPJOINEDDATA, v ktorej bolo uložených 10658 pneumatík. Celkový počet porovnaní, ktoré sme vykonali bol 1065800. Na ďalšom grafe sú znázornené frekvencie výskytu hodnôt sumovanej podobnosti po aplikácií klasifikátora.



Obr. 8 Frekvencia hodnôt podobnosti

Na základe vykonaných testov sme určili hranice pre zatriedenie do skupiny potenciálne zhodných produktov. Manuálnym overením zhodnosti produktov, na základe vypočítanej sumovanej podobnosti sa ukázalo, že ak hodnota sumovanej podobnosti neprekročí 93% tak vieme povedať, že ide o nový produkt. Uvedieme príklad dvojice produktov (pneumatiky, vľavo z JOINEDDATA, vpravo z WRAPJOINEDDATA) tesne pod touto hranicou.

EAN	5452000390912
KOD	GY236017YEFF
SIRKA	235.0000
PROFIL	55
RADIAL	R
RAFEK	17.0000
VYROBCE	34
KUSU_DATUM	30
...	...

EAN	5452000379481
KOD	GY235517YEFFAU
SIRKA	235.0000
PROFIL	60
RADIAL	R
RAFEK	17.0000
VYROBCE	34
KUSU_DATUM	34
...	...

V tomto prípade ide o dve rôzne pneumatiky, keďže sa nezhodujú jednak v určujúcom atribúte Ean a v relevantnom atribúte Profil.

Ďalej uvedieme príklad dvojice produktov (pneumatiky, vľavo z JOINEDDATA, vpravo z WRAPJOINEDDATA) tesne nad uvedenou hranicou.

EAN	6953913100777
KOD	257015F201NEU
SIRKA	215.0000
PROFIL	70
RADIAL	R
RAFEK	15.0000
IMGURL	http://www.treadtelpneu.cz/foto/obr__hifly-hf201.jpg
KUSU_DATUM	50
...	...

EAN	6953913100777
KOD	257015F201NEU
SIRKA	215.0000
PROFIL	70
RADIAL	R
RAFEK	15.0000
IMGURL	g>http://www.treadtelpneu.cz/foto/obr__hifly-hf201.jpg,
KUSU_DATUM	50
...	...

V uvedenom prípade ich algoritmus na základe sumovanej podobnosti nemohol klasifikovať ako zhodné, a k zníženiu hodnoty podobnosti prispel atribút Imgurl.

V prípade že hodnota sumovanej podobnosti je 100%, tak predpokladáme, že produkty sú zhodné. Dolnú hranicu sme zvolili 100%, pretože ani človek by nenašiel argument proti zhodnosti. Teda ak hodnota sumovanej podobnosti je medzi hranicami 93% a 100% (znázornené na predošlých grafoch červenými čiarami), tak produkt je označený ako potenciálne zhodný.

Takto nastavené hranice sme skúsili použiť v obidvoch unifikačných algoritmoch. V nasledujúcej tabuľke (tab. 2) sú zobrazené počty produktov tabuľky WRAPJOINEDDATA s hľadiska klasifikácie do jednotlivých skupín (nové, potenciálne zhodné, zhodné) , pre oba porovnávané algoritmy unifikácie.

Tab. 2 Počet produktov v klasifikačných skupinách

	nové	potenciálne zhodné	zhodné
Kompletný algoritmus	10561	0	97
Jednoduchý algoritmus	10551	10	97

Ďalšia tabuľka prezentuje počet porovnaní produktu s WRAPJOINEDDATA, s produktami JOINEDDATA tabuľky (tab. 3) potrebných pre klasifikáciu do jednotlivých

klasifikačných skupín (nové, potenciálne zhodné, zhodné), pre oba porovnávané algoritmy unifikácie.

Tab. 3 Počet porovnaní produktov na základe klasifikácie

	nové	potenciálne zhodné	zhodné
Kompletný algoritmus	373253	0	5044
Jednoduchý algoritmus	1055100	1000	9700

Z uvedenej tabuľky vyplýva, že kompletný algoritmus unifikácie potrebuje na klasifikáciu produktov menej porovnaní ako jednoduchý algoritmus. Množina potenciálne zhodných produktov pri kompletnom algoritme bola prázdna.

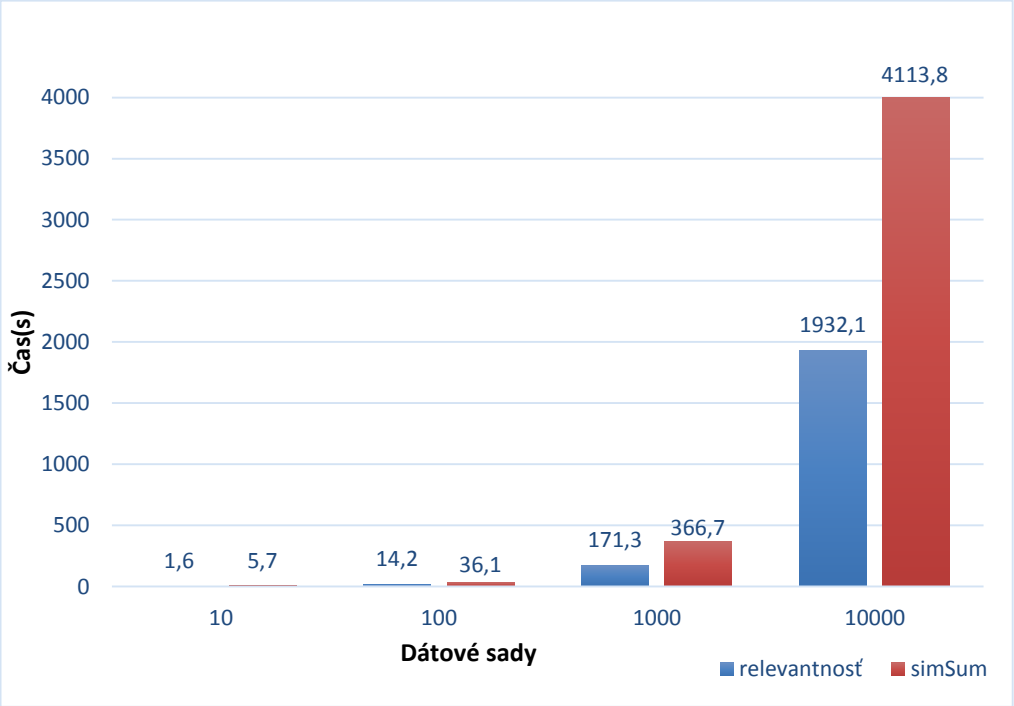
Manuálne sme overili množinu 10 potenciálne zhodných produktov, ktorú vytvoril jednoduchý algoritmus unifikácie. Z nich 8 bolo klasifikovaných kompletným algoritmom správne ako zhodné. To by na prvý pohľad mohlo vyzerať ako vylepšenie oproti jednoduchému algoritmu unifikácie. Zvyšné dva produkty však boli klasifikované kompletným algoritmom ako nové, pričom mali byť klasifikované ako zhodné. Jednoduchý algoritmus ich zatriedil aspoň do množiny potenciálne zhodných produktov. Ukázalo sa, že algoritmus nie je 100% presný, a administrátor systému sa nemôže spoliehať len na relevantnosť atribútov. Pre budúce nasadenie tohto unifikačného algoritmu bude potrebná hlbšia analýza a následná úprava pre zvýšenie presnosti.

6.2 Kvantitatívne testy

Kvantitatívne testy boli realizované s cieľom porovnať z hľadiska času a úspešnosti vyhodnocovania dva algoritmy. Najskôr sme vykonali sériu testov zameraných na porovnanie času vyhodnocovania procesu unifikácie kompletným nami navrhnutým algoritmom pre unifikáciu. Teda v procese unifikácie boli zapojené všetky prvky (relevantnosť atribútov, zdrojová závislosť...). Druhý prípad predstavoval proces unifikácie realizovaný výlučne metódou založenou na výpočte sumovanej podobnosti.

V oboch uvedených prípadoch sme testy vykonali postupne nad rôzne veľkou vstupnou tabuľkou JOINEDDATA. Tá obsahovala najskôr 10 potom 100 neskôr 1000 a napokon 10000 pneumatík. Pričom tabuľka WRAPJOINEDDATA obsahovala 15000 pneumatík z dvoch rôznych zdrojov. Výsledky testov ukázali, že v prvom prípade je algoritmus zjavne rýchlejší. Naopak v druhom prípade, keď sa proces unifikácie vykonáva len na základe výpočtu sumovanej podobnosti je algoritmus zjavne pomalší.

Nasledujúci graf (obr. 9) znázorňuje porovnanie časov (v sekundách) behu algoritmu unifikácie v uvedených dvoch prípadoch nad rôzne veľkými vstupnými množinami dát.



Obr. 9 Porovnanie časov

Z uvedeného grafu zreteľne vyplýva, že kompletný algoritmus unifikácie pracuje približne o polovicu rýchlejšie ako algoritmus unifikácie klasifikujúci len na základe sumovanej podobnosti.

Potom sme oba spomínané algoritmy porovnali z hľadiska úspešnosti vyhodnocovania unifikácie produktov. Ukázalo sa, že zjednodušený algoritmus je presnejší, a to v dôsledku počítania sumovanej podobnosti z hodnôt všetkých atribútov pre každú dvojicu porovnávaných produktov. V prípade kompletného algoritmu môže nastať možnosť, kedy vyhodnocuje len na základe porovnania *určujúcich* alebo *relevantných* atribútov, ktorých hodnoty môžu byť nesprávne vyplnené, ako dôsledok chyby pri predpríprave dát.

Prostredníctvom vykonaných kvantitatívnych testov nad danou doménou pneumatík, môžeme vyvodiť niekoľko pozorovaní. Empiricky sa potvrdilo, že sa v procese unifikácie môžeme spoliehať na relevantnosť atribútov. Inými slovami, v procese unifikácie nemusíme v niektorých prípadoch opierať o unifikáciu založenú len na výpočte sumovanej podobnosti. Čo výrazne môže ovplyvniť čas behu algoritmu nad danou doménou.

Ďalej sme na základe uvedených testov stanovili hranice pre klasifikáciu potenciálne zhodných produktov. Podmienkou pre efektívnu prácu algoritmu unifikácie je správna klasifikácia relevantnosti atribútov a korektné priradenie porovnávacích funkcií, pre jednotlivé atribúty danej domény.

Záver

V tejto práci sme navrhli algoritmus unifikácie produktov rovnakého druhu z viacerých zdrojov podľa atribútov. Pri návrhu a realizácií samotného algoritmu unifikácie sme vychádzali s informácií o všeobecnom modely unifikácie. Pričom návrh a realizáciu jednotlivých krokov prezentovaného algoritmu sme prispôsobili nášmu zadaniu.

Vytvoriť univerzálny unifikačný algoritmus pre akékoľvek vstupné dáta, v zmysle ľubovoľného vstupného formátu, či obsahu nie je možné. Niektoré algoritmy sa venujú procesu unifikácie, bohužiaľ nie sú natoľko univerzálne, aby sa dali aplikovať, respektíve prispôsobiť na naše zadanie. Aj preto sme sa rozhodli, že vytvoríme algoritmus unifikácie, ktorý dokáže pracovať so zložitejšími dátami na vstupe. Zložitejšie dáta predstavujú práve produktové katalógy tvorené doménami produktov.

V navrhnutom algoritme unifikácie je zapracovaných niekoľko prvkov. Niektoré z nich boli súčasťou iných podobných algoritmov, no mi sme ich modifikovali pre naše potreby. Ďalšie sme navrhli my ako inovatívne. Hlavným cieľom aplikácie uvedených prvkov v navrhnutom algoritme unifikácie bol predpoklad dosiahnutia kvalitnejších výsledkov.

Vykonané testy nad konkrétnou doménou produktov umožnili empiricky určiť hranice pre potenciálne zhodné produkty. Ďalej poukázali na prínos zapracovaných prvkov ako relevantnosť a zdrojová závislosť, z pohľadu času vykonávania procesu unifikácie.

V budúcnosti sa bude potrebné zamerať viac na hlbšiu analýzu algoritmu unifikácie na zvýšenie presnosti. Bude tiež potrebné sledovať správanie algoritmu pri väčších dátových sadách a otestovať ho nad viacerými doménami. Pre nasadenie do praxe je potrebné navrhnúť nástroj pre automatizáciu preprocesingu dát a nástroj na ručnú klasifikáciu potenciálne zhodných produktov.

Zoznam použitej literatúry

[1] Jiří Dokulil, Jakub Yaghob, Filip Zavoral: Evoluce replikačních algoritmů v stohově orientovaných systémech, report, 2006

[2] D. Bednarek, D. Obdrzalek, J. Yaghob, F. Zavoral: Data Integration Using DataPile Structure. In proc. of the 9th East-European Conference on Advances in Database and Information Systems, Tallinn, Estonia, 2005

[3] Peter Christen: Data Matching, Concepts and Techniques for Record Linkage, Entity resolution, and Duplicate Detection, 2012

[4] Balík porovnávacích funkcií dostupný: http://www.sheffield.ac.uk/dcs

Prílohy

Príloha A: CD médium – diplomová práca v elektronickej podobe, prílohy v elektronickej podobe.