VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

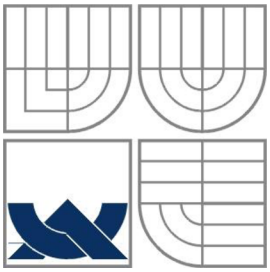
METODY PRO SHLUKOVÁNÍ DAT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

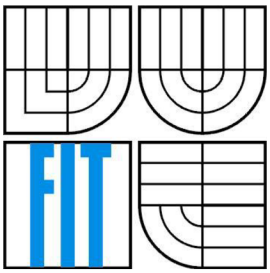
AUTOR PRÁCE
AUTHOR

ANTONÍN POHLÍDAL

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

METODY PRO SHLUKOVÁNÍ DAT

METHODS FOR CLUSTERING DATA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ANTONÍN POHLÍDAL

VEDOUCÍ PRÁCE
SUPERVISOR

ING. VLADIMÍR BARTÍK, PH.D.

BRNO 2009

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2008/2009

Zadání bakalářské práce

Řešitel: **Pohlídal Antonín**
Obor: Informační technologie
Téma: **Metody pro shlukování dat**
Kategorie: Databáze

Pokyny:

1. Seznamte se s problematikou shlukování a jednotlivými shlukovacími metodami.
2. Seznamte se s jazykem Java a databází MySQL.
3. Po dohodě s vedoucím zvolte jednu ze shlukovacích metod a navrhněte koncepci příslušné aplikace. Volbu metody a návrh konzultujte s vedoucím.
4. Implementuje danou metodu. Do vyvíjené aplikace zahrňte také dříve vytvořené implementace jiných shlukovacích metod. Ověřte funkčnost na vzorku dat a porovnejte výsledky s ostatními metodami.
5. Zhodnoťte dosažené výsledky a další možná pokračování tohoto projektu.

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, 2001.

Při obhajobě semestrální části projektu je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bartík Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2008
Datum odevzdání: 20. května 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zabývá hierarchickými metodami shlukování se zaměřením na implementaci shlukující metody zdola-nahoru a její porovnání s metodou DENCLUE. Nejdříve jsou popsány různé metody shlukování s důrazem na hierarchické metody. Následuje implementace vybrané metody za použití programovacího jazyka Java a databáze MySQL. Poslední část obsahuje porovnání zmíněné implementace s metodou DENCLUE, kterou implementoval pan Bc. Radim Kapavík.

Abstract

This bachelor's thesis deals with hierarchical clustering methods with a focus on implementation of agglomerative hierarchical clustering method and its comparison with the DENCLUE method. First of all, various methods are described with emphasis on hierarchical clustering methods. Further, there is an implementation of the selected method, using the Java programming language and MySQL database. The last part contains a comparison with the implementation of DENCLUE method, implemented by Mr. Bc. Radim Kapavík.

Klíčová slova

Získávání znalostí z databází, shluková analýza, dolování dat, shlukovací metody, hierarchické metody, MySQL, DENCLUE, porovnání metod.

Keywords

Knowledge discovery in databases, cluster analysis, data mining, clustering methods, hierarchical methods, MySQL, DENCLUE, methods comparison.

Citace

Pohlídal Antonín: Metody pro shlukování dat, bakalářská práce, Brno, FIT VUT v Brně, 2009

Metody pro shlukování dat

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vladimíra Bartíka, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Antonín Pohlídal
20.5.2009

Poděkování

Chtěl bych poděkovat panu Ing. Vladimíru Bartíkovi, Ph.D. za jeho odborné vedení, rady a pomoc při tvorbě této bakalářské práce. Poděkování patří také mé rodině za velkou podporu a neocenitelné rady.

Obsah

1 Úvod

Cílem této bakalářské práce je nastínit problematiku získávání znalostí z databází se zaměřením na její statistickou metodu – shlukovou analýzu. Po rozboru shlukové analýzy je následně provedena implementace její hierarchické shlukující metody (zdola-nahoru) za účelem předvedení její funkčnosti v praxi, pro lepší pochopení dané problematiky a také za účelem lepší názornosti. Tato implementace je potom porovnána s metodou DENCLUE, kterou implementoval pan Bc. Radim Kapavík. Cílem tohoto porovnání je názorná ukázka rozdílností metod a jejich výsledků. Při implementaci je také kladen důraz na návrh aplikace, která by měla zahrnovat vhodné uživatelské rozhraní pro obě metody, práci s daty a prezentaci výsledků.

Druhá kapitola se nejdříve zabývá stručným úvodem a historickým vývojem získávání znalostí z databází. Dále je rozebrána definice tohoto pojmu a proces, při kterém se provádí samotné získávání nových znalostí. Nakonec je zmíněna motivace a využití v praxi.

Ve třetí kapitole je popsána definice shlukové analýzy spolu s problémy, které musí tato statistická metoda řešit. Potom jsou nastíněny možnosti využití těchto metod v našem každodenním životě. Následuje stručný popis jednotlivých shlukovacích metod a požadavků, které jsou na tyto metody kladeny, stejně jako rozbor používaných datových struktur a různých typů dat.

Kapitola čtvrtá podrobněji rozebírá popis, výhody a nevýhody všech hierarchických metod. Dále zde jsou zmíněny shlukující (zdola-nahoru) a rozdělující (shora-dolů) hierarchické metody společně s popisem jejich práce a grafickou ukázkou postupu.

Obsahem páté kapitoly je implementace zvolené shlukovací metody (zdola-nahoru). Je zde rozebrán mnou implementovaný algoritmus, popsány zvolené ukončující podmínky a vysvětlen způsob zpracovávání zvolených typů proměnných. Následuje popis databázového systému MySQL, ve kterém je obsažena historie vývoje, jeho využití a také popis výhod a nevýhod. Dále je rozebráno uložení tabulek v databázi, provedení vkládání dat a jejich následné zobrazení. V závěru této kapitoly se věnuji návrhu a popisu uživatelského rozhraní a způsobu zobrazení výsledků shlukování.

Šestá kapitola vysvětluje důležitost testování výsledného programu, popisuje princip prováděného testování a demonstruje dva ukázkové testy. První zobrazený test byl proveden na uměle vytvořeném vzorku dat a u druhého je použit reálný vzorek dat.

V sedmé kapitole je provedeno porovnání metody DENCLUE se shlukující hierarchickou metodou. Nejdříve je zmíněna rozdílnost ve kvalitě shlukování, dále jsou rozebrány výhody a nevýhody obou metod a nakonec je popsána a na grafu znázorněna závislost času na velikosti dat.

Poslední kapitola obsahuje závěr, ve kterém je provedeno zhodnocení celé této práce a jsou navrženy různé možnosti vylepšení a rozšíření do budoucna.

2 Získávání znalostí z databází

S rozvojem techniky přichází různé možnosti ukládání a archivování dat. Elektronická podoba psaného slova zapříčinila revoluci v této oblasti. Zpočátku z toho plynuly jen samé výhody a ty zastínily pár nevýhod, které se vyskytly. A tak data v psané nebo tištěné podobě uložená v objemných archívech byla nahrazena elektronickou formou v daleko menších datových skladech. Úspora velkého množství místa, papíru a dalších prostředků vynaložených na zpracovávání dat přilákala zájemce ze všech oborů lidské činnosti. Největší výhoda ovšem byla v rychlejším a jednodušším zpracování různých informací. Po celém světě nastává rychlý rozvoj elektronické podoby dat, možností jejich uložení a následného zpracovávání. Zde ovšem začal v tichosti narůstat problém. Místa bylo najednou mnohonásobně více než doposud a proto se začalo ukládat stále větší množství dat. Nyní se již nemuselo provádět tak důkladné třídění, ale mohly se zpracovávat a ukládat i informace, které se dříve zdály méně podstatné anebo zbytečné.

2.1 Vývoj

Problém velkého množství dat začal být stále více patrný koncem 80. a začátkem 90. let minulého století. Nejenže neustále vzrůstá objem, ale především začíná být čím dál tím více obtížné provádět efektivní zpracovávání. Z velkých databází by se za použití jiných technik dalo získat hodně užitečných znalostí využitelných pro další rozhodování. Potřeba nové, lepší a hlavně automatizované práce s daty přichází ze všech oblastí, kde se zpracovává a uchovává velké množství údajů. Jedná se např. o oblasti průmyslové a zemědělské výroby, obchodu, státní správy, vědy, výzkumu ale i dalších. O těchto problémech se poprvé začíná debatovat na konferencích o umělé inteligenci v Americe. Až doposud se totiž oblast databází, statistiky a umělé inteligence vyvíjely nezávisle na sobě. Nyní bylo potřeba propojit znalosti ze všech těchto oblastí, aby mohl vzniknout nový obor získávání znalostí z databází (anglicky nazývaný Knowledge Discovery in Databases, KDD). V současnosti bývá také nazýván dolování dat (anglicky nazývaný Data Mining). Oba zmíněné názvy se často používají jako synonymum i když dolování dat je pouze jednou fází získávání znalostí z databází. Zájem o toto téma dále neustále vzrůstal po celém světě. Následují další konference nejen v Americe, ale i v Evropě či v Asii. Danou problematikou se také začínají zabývat různé odborné i více obecné časopisy. Získávání znalostí z databází se rychle stává velmi známým oborem po celém světě.

2.2 Definice a proces zpracování dat

„Můžeme říci, že *získávání znalostí z databází* je extrakce (neboli „dolování“) zajímavých (netriviálních, skrytých, dříve neznámých a potencionálně užitečných) modelů dat a vzorů z velkých objemů dat. Tyto modely a vzory reprezentují znalosti získané z dat.“ [1]. Je tedy velmi důležité, že získáváme netriviální a skryté informace, které nejsou na první pohled zřejmé. Když budeme provádět pouze jednoduché výpisy z databáze, tak to nemůžeme považovat za získávání znalostí, protože z vypsaných informací se nedozvíme nic nového. Proces samotného získávání znalostí z databází se skládá z následujících sedmi fází:

1) *Čištění dat* – provádí se za účelem odstranění šumu a nekonzistence dat.
2) *Integrace dat* – v této fázi dochází ke spojování dat z různých zdrojů. První a druhá fáze se často provádí jako předzpracování hned po sobě a výsledek je následně ukládán do databáze.

3) *Výběr dat* – vybírají se jen ta data z databáze, která jsou určena k dalšímu rozboru.
4) *Transformace dat* – cílem je transformovat nebo konsolidovat data do vhodné podoby pro dolování tím, že provedeme sumarizační nebo agregační operace. Občas bývá tato fáze prováděna před třetí fází, protože je součástí tvorby datového skladu.
5) *Dolování dat* – tvoří základní proces při získávání znalostí, ve kterém jsou prováděny různé metody za účelem extrakce datových vzorů.
6) *Hodnocení vzorů* – hledá zajímavé vzory reprezentující znalosti založené na míře užitečnosti.
7) *Prezentace znalostí* – používá různé techniky vizualizace a reprezentace znalostí k zobrazení získaných znalostí.

2.3 Motivace a využití

Největší motivací a hnací silou vývoje a používání získávání znalostí z databází, je od počátku vidina nových znalostí o dané problematice. Mnoho různých databází přímo lákalo k jejich propojení a tak vytvoření ještě větší a komplexnější databáze, ze které se dá získat mnohem více nových znalostí. Mít znalosti, které nikdo jiný nemá, totiž znamená velkou výhodu a náskok před ostatními.

Získané znalosti se následně využívají při dalším řízení a rozhodování. Jedná se např. o rozbor prodeje a poptávky na trhu, přilákání nových a udržení stávajících zákazníků, řízení státní správy a podniků, detekce podvodů a další. Stejně tak se nachází využití v analýze multimediálních dat, která jsou v poslední době čím dál tím více populárnější. Uplatnění se najde skoro v každém oboru a začíná být také patrné i v běžném životě. Vyhledávače již umí hledat písničky jen podle melodie, kterou zabroukáte přes mikrofon, anebo najít obrázek podle tématiky či převládajících barev. Mnohé internetové obchody mají u zboží uvedeno, jaké další věci si většinou jiní zákazníci s tímto zbožím kupují. Je to v podstatě cílená reklama na základě zpracování dat o předešlých nákupech. Toto všechno by dříve bylo možné dokázat jen za ohromného úsilí a i tak by se pravděpodobně nedosáhlo stejně dobrého výsledku.

3 Shluková analýza

„Statistika nabízí celou řadu teoreticky dobře prozkoumaných a zdůvodněných a léty praxe ověřených metod pro analýzu dat. Pro oblast dobývání znalostí z databází mají význam (ať už přímo jako používané metody, nebo nepřímo jako zdroj inspirace):

- kontingenční tabulky – pro zjišťování vztahu mezi dvěma kategoriálními veličinami,
- regresní analýza – pro zjišťování funkční závislosti jedné numerické (spojité) veličiny na jiných numerických veličinách,
- diskriminační analýza – pro odlišení příkladů (pozorování) patřících do různých tříd,
- shluková analýza – pro nalezení skupin (shluků) navzájem si podobných příkladů.“ [2]

Existují i další používané metody jako je korelační analýza, analýza rozptylu nebo faktorová analýza.

3.1 Definice shlukové analýzy

Shluková analýza (anglicky nazývaná Cluster Analysis) je tedy proces seskupování daných objektů do shluků (nazývaných také třídy nebo clustery), ve kterých mají tyto objekty stejné nebo velmi podobné vlastnosti. Objekty v jednom shluku potom mají jiné nebo méně podobné vlastnosti, než objekty v dalším shluku.

Problém může nastat při určování, co náleží do jednoho shluku a co již patří do jiného. Třeba rozlišení jestli se jedná o muže nebo ženu je daleko jednodušší, než se snažit zjistit, které ženy mají podobné způsoby při nakupování. Proto se při posuzování podobnosti dvou objektů berou v potaz všechny jejich vlastnosti, na základě kterých se často za pomocí vzdálenostní funkce vypočítá míra podobnosti těchto dvou objektů. Následně se tato míra využije při rozhodování, jestli se objekt zařadí do nějakého shluku anebo se ukončí shlukování, protože si již žádné objekty nejsou podobné, aby se daly zařadit do shluku. Po ukončení shlukování se nám také může stát, že některé shluky obsahují jen jeden objekt. Těmto objektům se říká odlehlé objekty (anglicky bývá nazývané Outliers) a jejich hledáním se zabývá analýza odlehlých objektů. Někdy nás totiž mohou zajímat právě tyto objekty, které se více odlišují.

3.2 Využití v reálném životě

Shluková analýza je velmi používaná v mnoha aplikacích, ve výzkumu trhu, k rozpoznávání vzorů, k datové analýze, při zpracovávání obrazů. V obchodování pomáhá shlukování odhalit různé skupiny zákazníků a provést charakteristiku těchto skupin na základě jejich nákupních zvyklostí. V biologii se používá na odvozování systému rostlin a živočichů, na třídění genů s podobnou funkčností a na získání náhledu na vnitřní složení populací. Shlukování může být také použito při výzkumu země k identifikování podobných oblastí, k zjištění skupin domů ve městech, které mají podobný typ, cenu, polohu, ale také k určení skupin lidí s pojištěním auta, kteří mají vyšší než průměrné pojistné události. Shluková analýza může být také použita ke třídění dokumentů na internetu za účelem lepšího získávání informací. [3]

Aniž si to uvědomujeme, tak se shlukovou analýzu učíme již od našeho dětství a následně ji využíváme po celý život, při rozlišování všeho, co nás obklopuje. Vše začíná po narození jednoduchou analýzou lidí kolem nás na známé (rodina) a neznámé tváře. Později přichází rozlišování

předmětů, rostlin, zvířat, lidí, našich pocitů, nálad a dalších věcí. Čím jsme starší, tím lépe umíme aplikovat důmyslnější a přesnější metody shlukování. Příkladem může být příchod nového žáka do třídy, ve které se děti již znají. Z pohledu tohoto žáka existuje jeden shluk – třída a jeden odlehlý objekt – on. Po prvním dnu ve škole se předchozí uspořádání může změnit již na dva shluky. Menší shluk tvoří nový žák spolu s žákem, se kterým sedí v lavici a druhý shluk obsahuje zbytek třídy. Na tomto příkladu je krásně uvedeno, že při provádění shlukové analýzy je také důležité brát ohled na typ zpracovávaných dat. Různá data se mohou měnit různou rychlostí. Některá se nemění vůbec, u jiných může být změna patrná během několika sekund, dnů, měsíců, let ale také staletí, tisíciletí i více.

3.3 Shlukovací metody

Na světě existuje mnoho různých shlukovacích metod. Ne každá taková metoda se hodí pro danou situaci, a proto je výhodné používat i více než jednu metodu na danou databázi. Takto získané výsledky se potom analyzují a porovnávají mezi sebou.

3.3.1 Požadavky na různé metody

Zpracovávání velkých databází není jednotvárná činnost, protože se pracuje s velkým množstvím různých dat. Proto vzniklo více rozdílných metod shlukové analýzy, kde každá z těchto metod má trochu jiné vlastnosti. Z toho tedy vyplývá, že ne každá metoda se hodí na danou situaci. Na tyto metody jsou ovšem kladeny následující požadavky:

- Škálovatelnost: Je potřeba, aby shlukovací algoritmy mohly pracovat s libovolně velkým objemem dat. Zpracovávání velkých i malých databází by mělo dávat stejně dobré výsledky.
- Schopnost zpracovávat různé typy atributů: Databáze většinou obsahují položky s různými typy dat a proto je potřeba, aby se při shlukování braly v potaz všechny tyto typy.
- Získávání shluků libovolných tvarů: Metody používající Euklidovskou nebo Manhattanskou vzdálenostní funkci nacházejí kulovité shluky podobné velikosti a hustoty. Shluky ovšem mohou být různých tvarů a tak i lépe odpovídat hledaným znalostem.
- Minimální požadavky na znalost problému při určování parametrů: Potřeba zadávat různé parametry může zhoršovat výsledky shlukování, protože bývá obtížné zjistit optimální hodnoty.
- Schopnost zpracovat data obsahující šum: Kvalita výsledku shlukování může být také zhoršena tím, že v databázi jsou určitá data poškozena anebo úplně chybí.
- Necitlivost na pořadí vstupních záznamů: Je důležité, aby algoritmus dané shlukující metody, dával stejné výsledky bez ohledu na to, v jakém pořadí jsou data zpracovávána.
- Zpracovávání vysokodimenzionálních dat: Data se dvěma až třemi atributy lze dobře zpracovávat mnohými algoritmy a dokonce i lidským okem. Pro potřeby zisku nových znalostí jsou ovšem cennější algoritmy, které umí pracovat i s větším množstvím těchto atributů.
- Shlukování na základě omezování: Často potřebujeme nacházet shluky ve velkých databázích za použití určitých omezovacích pravidel, abychom dosáhli co nejpřesnějšího výsledku.
- Interpretovatelné a použitelné výsledky: Posledním a také velice důležitým požadavkem je umět dosažené výsledky interpretovat a použít při dalším rozhodování.

3.3.2 Datové struktury a typy dat

Pokud máme určitá data obsahující *n* objektů, kde každý objekt může reprezentovat např. rostlinu, zvíře, věc, člověka atd. a potřebujeme provést shlukování, tak většina shlukovacích algoritmů používá jednu z následujících dvou datových struktur:

- Datová matice: Reprezentuje *n* objektů (např. rostlin) pomocí *p* proměnných (neboli atributů) jako je jméno rostliny, její výška, stáří atd. Struktura tak může mít podobu relační tabulky, nebo matice *n* (objekty) × *p* (proměnné).

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

- Podobnostní matice: Obsahuje soubor vzdáleností všech jednotlivých dvojic *n* objektů. Nejčastěji se zobrazuje jako tabulka *n* × *n* prvků.

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \ddots & 0 & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

  V matici reprezentuje *d(i, j)* vzdálenost mezi objektem *i* a objektem *j*. Všeobecně je vzdálenost *d(i, j)* nezáporné číslo, které se blíží k 0, když jsou objekty *i* a *j* blízko u sebe anebo jsou hodně podobné. Čím je tato vzdálenost větší, tím více se dané objekty od sebe odlišují. Vzdálenost jednoho a toho samého objektu je *d(i, i) = 0*. Matice je souměrná a proto platí *d(i, j) = d(j, i)*.

Při provádění shlukové analýzy často pracujeme s různými typy dat. Může se jednat o proměnné intervalové, binární, nominální, ordinální a poměrové. Pro každý typ proměnné existují různé metody jejich zpracovávání. V databázích jsou většinou objekty, které obsahují několik atributů o různých typech dat. Shlukování by se dalo provádět zvlášť pro každý datový typ, ale tím by zde nastal problém nepřesnosti, protože by mohly vycházet různé výsledky. Nejvhodnější je provádět jednu shlukovou analýzu, při které se zpracovávají všechny atributy současně. Máme-li tedy určitá data (objekty *i* a *j*), která jsou popsána *p* atributy neboli proměnnými různých typů, určujeme jejich vzdálenost následovně:

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}$$

Vzdálenost d mezi objekty i a j tedy spočítáme tak, že pro každou proměnnou těchto dvou objektů vypočítáme vzdálenost (ve vzorci jako $d_{ij}^{(f)}$), tuto vzdálenost vynásobíme koeficientem $\delta_{ij}^{(f)}$ (viz níže) a následně všechny takto vypočítané vzdálenosti sečteme a podělíme součtem všech koeficientů $\delta_{ij}^{(f)}$.

Ve vzorci je $\delta_{ij}^{(f)} = 0$, když (1) x_{if} nebo x_{jf} schází v databázi, nebo

(2) $x_{if} = x_{jf} = 0$ a proměnná f je binární asymetrická,

ve všech ostatních případech je $\delta_{ij}^{(f)} = 1$.

Proměnná *f* se podílí na celkové vzdálenosti objektu *i* a objektu *j*, tedy $d_{ij}^{(f)}$ a vypočítá se následujícím způsobem podle typu proměnné *f*:

- *f* je intervalová proměnná: $d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{max_h x_{hf} - min_h x_{hf}}$, kde h se bere pro všechny zadané hodnoty objektů s proměnnou *f*.
- *f* je binární nebo nominální: $d_{ij}^{(f)} = 0$, když $x_{if} = x_{jf}$, jinak $d_{ij}^{(f)} = 1$.
- *f* je ordinální: vypočítáme hodnoty r_{if} a $z_{if} = \frac{r_{if} - 1}{M_f - 1}$ a zpracujeme z_{if} jako intervalovou proměnnou.
- *f* je poměrová proměnná: buď provedeme logaritmickou transformaci a získaná data zpracujeme jako intervalové proměnné anebo zpracujeme *f* jako ordinální proměnnou.

3.3.3 Metody založené na rozdělování

Pokud máme databázi, která obsahuje *n* objektů, tak tyto metody provádí rozdělení daných objektů do *k* tříd. Jednotlivé třídy potom představují různé shluky a dále platí, že počet $k \leq n$. Nově vzniklé třídy musí obsahovat nejméně jeden objekt a každý objekt patří pouze do jedné třídy.

Na začátku se náhodně vyberou objekty, které budou reprezentovat počáteční třídy. Za použití iteračních přemisťovacích technik se přiřazují objekty do těchto tříd tak, aby si objekty v jedné třídě byly co nejvíce podobné a objekty v různých třídách co nejméně podobné. Za účelem dosažení co nejlepšího rozdělení se používají tyto metody:

a) *Metoda založená na centrálním bodu (k-means)* – třída je reprezentována fiktivním objektem, jehož hodnota je vypočítána jako průměr ze všech hodnot objektů v dané třídě.
b) *Metoda založená na reprezentujícím objektu (k-medoids)* – nepoužívá fiktivní objekt nýbrž ten objekt, který je nejblíže středu dané třídy.

Tyto metody vyhledávají shluky kulovitého tvaru a pracují dobře na malých a středně velkých databázích. Nevýhodou je nutnost zadání počtu shluků, do kterých se mají data rozdělit.

3.3.4 Hierarchické metody

Vytváří hierarchický rozklad dané množiny objektů. Rozklad může probíhat dvěma způsoby:

a) *Shlukující hierarchické metody (zdola-nahoru)* – jednotlivé objekty jsou umístěny do vlastní třídy, následuje shlukování nejpodobnějších, až dokud není splněna ukončovací podmínka.
b) *Rozdělující hierarchické metody (shora-dolů)* – umístí všechny objekty do jedné třídy a provádí rozdělování na menší shluky, dokud není splněna ukončovací podmínka.

Těmito metodami se budu podrobně zabývat v kapitole 4.

3.3.5 Metody založené na hustotě

Základní myšlenkou je tvoření shluků za pomocí hustoty daných objektů v prostoru. Shluk se vytváří do té doby, dokud není překročen určitý práh, daný tím, že každý objekt ve vytvářeném shluku musí mít kolem sebe alespoň minimální počet objektů. Díky tomuto postupu se vyhledávají shluky různých tvarů. Tyto metody se využívají k filtrování šumu, protože objekty v oblastech s malou hustotou se dají považovat za šum a stejně tak se za pomoci těchto metod dají hledat odlehlé objekty, o kterých jsme hovořili v kapitole 3.1.

a) *Metoda DBSCAN* – tato metoda provádí zvětšování daných shluků podle analýzy hustoty okolí, kterému se říká ε-okolí (poloměr tohoto okolí je dán parametrem ε).
b) *Metoda DENCLUE* – je to metoda, která shlukuje objekty na základě analýzy hodnoty distribučních funkcí hustoty.

3.3.6 Metody založené na mřížce

Tyto metody rozdělují prostor, ve kterém jsou umístěny dané objekty, na konečný počet buněk a tyto buňky potom tvoří mřížkovou strukturu. Nad touto datovou strukturou se následně provádí všechny požadované shlukující operace. Velkou výhodu, kterou mají tyto metody oproti ostatním, je rychlé zpracování dat, které je obvykle nezávislé na počtu objektů v databázi. Vliv na rychlost zpracování je tedy závislý pouze na počtu buněk mřížkové struktury.

a) *Metoda STING* – tato metoda je typickým představitelem metod založených na mříže, jenž pracuje se statistickými informacemi uloženými v mřížce.
b) *Metoda WaveCluster* – jedná se o metodu založenou na mřížce i na hustotě, která při shlukové analýze aplikuje vlnovou transformaci.

3.3.7 Metody založené na modelech

Metody založené na modelech se snaží najít shluky dat za pomoci funkce hustoty, která zobrazuje prostorové rozdělení datových objektů. Tyto metody v podstatě hledají nejlepší shodu mezi daným datovým modelem a modelem, který je předpokládán pro každý shluk dat. Mezi metody založené na modelech patří např.:

a) *Metoda Expectation-Maximization (EM)* – jedná se o rozšíření algoritmu *k-means* tak, že nejsou dány pevné hranice mezi shluky a provádí se dva kroky (1) *expectation* – výpočet pravděpodobnosti, s jakou objekt náleží ke shluku a (2) *maximization* – využívá se pravděpodobností k výpočtu nového objektu, který reprezentuje shluk.
b) *Metoda COBWEB* – nejznámější metoda konceptuálního shlukování, provádí pravděpodobnostní analýzu, kde jako model pro dané shluky dat je brán koncept.
c) *Metoda SOM (self-organizing feature map)* – je to síťově založená neuronová metoda, která provádí shlukování zobrazením vysoce dimenzionálních dat do 2-D nebo 3-D mapy.

3.3.8 Metody pro shlukování vysoce-dimensionálních dat

Dříve zmíněné metody pracují většinou dobře na datech s menším počtem dimenzí (neboli atributů), problém ovšem nastává při zpracovávání vysoce dimenzionálních dat. Aplikování výpočtu vzdáleností mezi body nebo zjišťování hustoty se stává neefektivní, protože jak se zvětšuje počet dimenzí, tak se data stále více rozptylují a může vznikat větší množství šumu. Z těchto důvodů se začaly vyvíjet metody určené pro shlukování vysoce dimensionálních dat.

a) *Metody založené na shlukování podprostorů* – patří sem metody *CLIQUE* a *PROCLUS*, které hledají shluky v subprostoru, neboli podmnožině dané dimenze zadaných dat.
b) *Metody výběru atributů* – tyto metody získávají často se objevující atributy mezi podmnožinou dimenzí, které se nejvíce vyskytují. Tyto atributy se následně používají k seskupování objektů za účelem generování užitečných shluků. Příkladem může být metoda *pCluster*, která sdružuje objekty na základě podobnosti jejich atributů.

4 Hierarchické metody

Tyto shlukující metody pracují tím způsobem, že umísťují zadaná data, ať už jsou to jednotlivé objekty nebo celé shluky, do stromové struktury. V této struktuře je potom každý list tvořen jedním shlukem obsahujícím minimálně jeden objekt. Podle způsobu tohoto umísťování dat do stromové struktury, rozlišujeme hierarchické metody na - shlukující (zdola-nahoru) a rozdělující (shora-dolů).

Při výběru vhodnosti hierarchických metod na danou problematiku, by se měl brát ohled na jejich nevýhody. Podstata těchto nevýhod je v tom, že při vytváření stromové struktury, se po sloučení nebo rozdělení daných shluků nedá vzít tento krok zpět. Kdyby se tedy následně ukázalo, že se dalo provést lepší sloučení nebo rozdělení, tak již není možné tento „špatný“ krok zpětně najít, upravit a pokračovat, protože není známo, co se při jakém kroku provádělo.

V průběhu shlukování se o slučování nebo rozdělování rozhoduje na základě vzdálenosti jednotlivých shluků. Proto je potřeba na začátku vypočítat všechny vzájemné vzdálenosti zadaných objektů. Tento prvotní výpočet se liší většinou jen použitím jiné vzdálenostní funkce. Při výpočtu vzdáleností shluků během shlukování, se používá jedna z následujících metod:

1) *Metoda nejbližšího souseda* – vzdálenost mezi shluky se počítá jako minimum z jejich vzájemných vzdáleností.

$$d_{min}\left(C_i, C_j\right) = min_{p \in C_i, p' \in C_j} |p - p'|$$

2) *Metoda nejvzdálenějšího souseda* – vzdálenost mezi shluky se počítá jako maximum z jejich vzájemných vzdáleností.

$$d_{max}\left(C_i, C_j\right) = max_{p \in C_i, p' \in C_j} |p - p'|$$

3) *Centroidní metoda* – vzdálenost mezi shluky se počítá jako vzdálenost mezi středy těchto shluků.

$$d_{mean}\left(C_i, C_j\right) = \left|m_i - m_j\right|$$

4) *Metoda průměrné vzdálenosti* – vzdálenost mezi shluky se počítá jako průměr z jejich vzájemných vzdáleností

$$d_{avg}\left(C_i, C_j\right) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'|$$

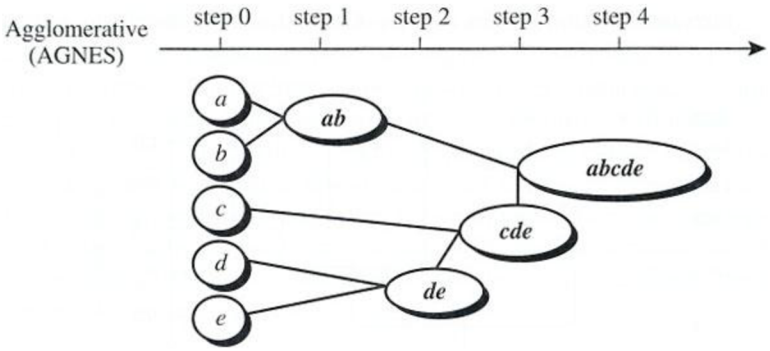
V těchto vzorcích znamená $|p - p'|$ vzdálenost mezi objekty p a p', m_i a m_j jsou středy tříd C_i a C_j, n_i a n_j jsou počty objektů třídy C_i a C_j.

4.1 Shlukující metody (zdola-nahoru)

Na začátku shlukování se provede umístění každého objektu do samostatného shluku. Vznikne nám soubor jednotlivých shluků, které se postupně mezi sebou slučují na základě vzdáleností mezi danými shluky. Shlukování se provádí do té doby, dokud nám např. nevznikne jeden shluk, získáme požadovaný počet shluků, vzdálenost mezi shluky přesáhne zadanou hodnotu anebo při splnění jiné, námi definované podmínky.

Typickým představitelem je metoda AGNES. Algoritmus této shlukující metody se dá rozdělit na dvě fáze. V první, inicializační fázi, se nejdříve provede umístění všech objektů do samostatných shluků a následně se vypočítá vzájemná vzdálenost mezi všemi objekty za použití nějaké vzdálenostní funkce. Druhá fáze je hlavní cyklus celého algoritmu, který se provádí

opakovaně, dokud není splněna ukončovací podmínka. Nejdříve se ve vypočítaných vzdálenostech najdou dva shluky, které jsou si nejblíže a ty se spojí do nového shluku. Pro takto nově vzniklý shluk se vypočítá nová vzdálenost od ostatních shluků. Celá druhá fáze se následně opakuje. Demonstrace této metody je vidět na následujícím obrázku 4.1.

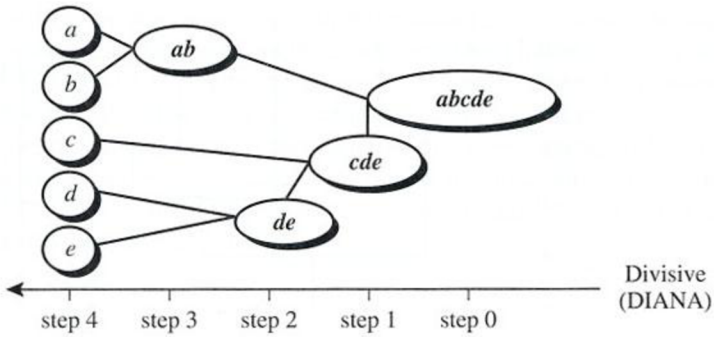


Obrázek 4.1 Ukázka shlukující metody AGNES (zdola-nahoru) [3] - převzato

4.2 Rozdělující metody (shora-dolů)

Rozdíl oproti shlukujícím metodám je ten, že se na začátku umístí všechny objekty do jednoho shluku. Tento shluk je následně rozdělován na stále menší shluky, až dokud např. každý shluk nebude obsahovat jeden objekt, nebo získáme požadovaný počet shluků, nebo průměr každého shluku splňuje zadanou podmínku anebo není splněna jiná předem daná ukončovací podmínka.

Představitelem rozdělujících metod je např. metoda DIANA. Algoritmus této shlukující metody se dá rozdělit na dvě fáze. Během první fáze inicializace se provede umístění všech daných objektů do jednoho shluku. Ve druhé fázi následuje hlavní cyklus tohoto algoritmu, který se opakuje do té doby, dokud není splněna zadaná ukončovací podmínka. V každém kroku hlavního cyklu je potom největší shluk rozdělen na dva např. na základě výpočtu největší Euklidovské vzdálenosti mezi nejbližšími sousedními objekty ve shluku. Demonstrace této metody je zobrazena na následujícím obrázku 4.2.



Obrázek 4.2 Ukázka shlukující metody DIANA (shora-dolů) [3] - převzato

5 Moje implementace

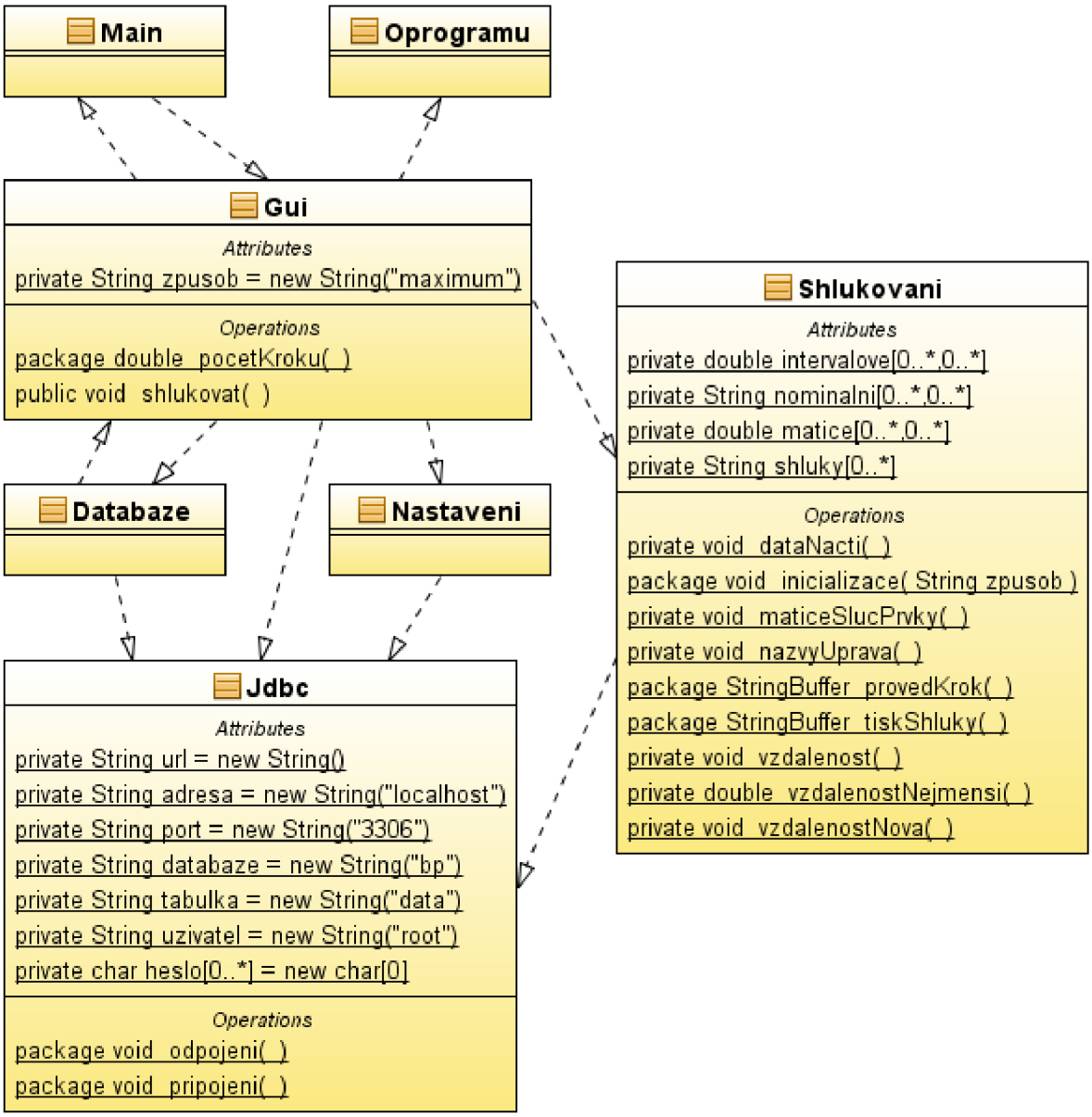
Výsledný program je implementován za použití programovacího jazyka Java, vývojového prostředí NetBeans a databázového systému MySQL. Pro komunikaci s MySQL je použit MySQL Connector/J, což je oficiální JDBC ovladač určený pro MySQL, jenž je dostupný na internetové adrese http://dev.mysql.com/downloads/connector/j/5.1.html (květen 2009).

5.1 Specifikace cíle

Tato bakalářská práce si klade za cíl implementovat shlukující hierarchickou metodu (zdola-nahoru), provést názornou demonstraci její funkčnosti a následně tuto implementaci porovnat s implementací metody DENCLUE, kterou provedl pan Bc. Radim Kapavík. Abychom dosáhli tohoto cíle, musíme postupovat podle následujících kroků. Nejdříve je tedy nutné navrhnout daný algoritmus za použití programovacího jazyka Java a databáze MySQL. Vstupem tohoto algoritmu budou data z databáze a parametry dané metody. Proto je potřeba navrhnout vhodný způsob uložení dat v databázi, jejich vkládání, zpracovávání a zobrazení těchto dat uživateli. Stejně tak by měl být proveden návrh vhodného zobrazení výsledků shlukování. Součástí této práce je dále návrh vhodného uživatelského rozhraní. To by mělo obsahovat všechny předešlé požadavky. Stejně tak je potřeba při jeho návrhu brát v potaz to, že kromě zmíněných dvou metod, které bude aplikace obsahovat, by mělo být možné v budoucnu přidávat i další metody pro shlukování.

5.2 Shlukující algoritmus

Samotná problematika shlukování je rozdělena do tří následujících tříd. První třída se jmenuje *Shlukovani* a je obsažena v souboru *Shlukovani.java*. Tato třída tvoří hlavní jádro celého procesu, protože zde jsou uloženy všechny metody a proměnné potřebné pro vykonávání samotného shlukování. Druhá třída má jméno *Jdbc* a je umístěna v souboru *Jdbc.java*. V této třídě jsou metody určené pro práci s vybranou databází a je zde také uloženo veškeré nastavení databáze. Třetí třída se nazývá *Gui* a nachází se v souboru *Gui.java*. V této třídě je umístěna hlavní řídící metoda *shlukovat()*. Tato metoda řídí shlukování dat tím, že obsahuje cyklus, ve kterém volá příslušné metody ze tříd *Shlukovani* a *Jdbc*. Kromě zmíněné metody je ve třídě *Gui* definováno také uživatelské rozhraní hlavního okna aplikace. Pro lepší přehlednost a názornost uvádím na obrázku 5.1 diagram tříd.



Obrázek 5.1 Diagram tříd

Celý proces shlukování začíná stiskem tlačítka „Shlukovat“. Toto tlačítko po stisku zavolá metodu *Gui.shlukovat()*, která nejdříve provede vymazání textového pole, abychom ho měli připravené k výpisu informací o shlukování. Dále zkontroluje, jestli uživatel specifikoval, že chce shlukovací algoritmus ukončit po provedení zvoleného počtu kroků. Pokud ano, je zavolána metoda *Gui.pocetKroku()*, jenž nám vrátí zadanou hodnotu, kterou si uložíme do proměnné typu *double*. Tímto způsobem získáme počet kroků, které budeme provádět. V případě, že uživatel specifikoval jako ukončující podmínku vznik jednoho shluku, se počet kroků neukládá. Nyní si uložíme počáteční čas procesoru, abychom po skončení zjistili celkovou dobu provádění shlukování. Dále se připojíme k databázi zavoláním metody *Jdbc.pripojeni()*. Tato metoda naváže komunikaci s databází použitím adresy a přihlašovacích údajů, jež se dají uložit v nastavení programu. Pokud uživatel tyto údaje nespecifikuje, tak se použijí výchozí – adresa: localhost, port: 3306, databáze: bp, tabulka: data, uživatelské jméno: root a heslo je prázdné. Nyní následuje inicializační fáze zavoláním metody *Shlukovani.inicializace(Gui.zpusob)*. Tato metoda se provádí pouze jednou před začátkem shlukování a je jí předán parametr typu *String*, který obsahuje slovo „maximum“, „minimum“ nebo „prumer“,

podle toho jakým způsobem budeme počítat novou vzdálenost pro slučované objekty. Jako výchozí hodnota je uloženo „maximum“, změnu je možné provést v uživatelském rozhraní zatrhnutím příslušného *radio buttonu*. Samotná inicializace provede načtení veškerých dat z databáze do polí za účelem rychlejšího zpracování. Intervalové proměnné uloží do dvourozměrného pole typu *double*, nominální proměnné uloží do dvourozměrného pole typu *String* a názvy jednotlivých objektů uloží do kolekce *ArrayList<String>*. Poslední, co se v inicializační fázi provádí je volání metody *Shlukovani.vzdalenost()*, jenž do dvourozměrného pole typu *double* vypočítá počáteční vzdálenosti mezi všemi objekty z databáze. Po dokončení inicializační fáze již není potřeba práce s databází, proto se od ní odpojíme pomocí metody *Jdbc.odpojeni()*. Následuje cyklus *while*, který provádíme, dokud nám nevznikne jeden shluk. Jestli si uživatel zvolil provádění jen určeného počtu kroků, tak je na začátku tohoto cyklu podmínka, která provádění shlukování ukončí, když dosáhneme požadovaného počtu kroků. Za tímto testem následuje příprava zobrazení informací. Nejdříve se proměnná typu *StringBuffer* vymaže, abychom si byli jisti, že v ní nezůstaly žádné předchozí informace. Jako první uložíme číslo aktuálního kroku. Dále se volá metoda *Shlukovani.provedKrok()*, jenž vykoná jeden krok shlukování a vrátí informace o tomto kroku. Při provádění jednoho kroku shlukování se nejdříve volá metoda *Shlukovani.vzdalenostNejmensi()*, která najde nejmenší vzdálenost mezi dvěma objekty v poli vzdáleností. Tímto získáme dva objekty určené ke sloučení, jenž se provede metodou *Shlukovani.maticeSlucPrvky()*. Zmíněná metoda provede odstranění druhého objektu a zachová pouze první, který je považován za nový shluk obsahující oba slučované objekty. Dalším krokem je provedení úpravy vzdálenosti mezi novým shlukem a ostatními objekty, což nám zajistí metoda *Shlukovani.vzdalenostNova()*. Nová vzdálenost se počítá jako maximum, minimum anebo průměr z původních vzdáleností slučovaných objektů. Zde tedy záleží pouze na uživateli, jaký způsob výpočtu nové vzdálenosti zadal před samotným prováděním shlukování. Ještě zbývá provést úpravu názvu nového objektu, který je pojmenován jako sloučenina jmen shlukovaných objektů. Nakonec se provede vrácení informací o tom, jaké dva objekty se sloučily a jaká byla mezi nimi vzdálenost. Posledním krokem cyklu *while* je, že se k informacím o prováděném kroku přidá výpis aktuálních shluků pomocí metody *Shlukovani.tiskShluky()*. Kompletní výpis jednoho kroku uložíme jako prvek do kolekce *ArrayList<String>*. Po ukončení hlavního cyklu *while* je zaznamenán čas konce, ze kterého je vypočtena doba trvání shlukování, jenž se zobrazí spolu s výsledky celého provedeného shlukování.

5.2.1 Ukončující podmínky

Výběr vhodných ukončovacích podmínek je důležitým aspektem každé shlukovací metody. Po prostudování literatury [1], [2], [3] a dalších zdrojů jsem našel několik způsobů jak provést ukončení hierarchické shlukovací metody (zdola-nahoru). Nejčastěji je v těchto zdrojích zmiňováno vytvoření jednoho shluku, provedení zadaného počtu kroků nebo získání zadaného počtu shluků. Zmíněna je také následující možnost, u které jde v podstatě o vytvoření grafické reprezentace průběhu shlukování. Jedná se o vytvoření tzv. dendrogramu, jenž zobrazuje proces shlukování od počátku až po vznik pouze jednoho velkého shluku. Následným rozborem této grafické reprezentace se dá odvodit optimální počet shluků tím způsobem, že označíme určitý krok za postačující a s tímto krokem následně získáme výsledek prováděného shlukování. Já jsem se rozhodl implementovat dvě ukončovací podmínky – vytvoření jedno shluku a zadání počtu kroků shlukování. V aplikaci je ponecháno na uživateli, kterou z těchto dvou podmínek si vybere.

Podmínka jednoho shluku - provádí shlukování do té doby, než nám vznikne jeden velký shluk, který v sobě obsahuje všechny jednotlivé objekty. Tento konečný výsledek pro nás není moc

užitečný, protože nás většinou zajímá rozdělení do více než jednoho shluku. Co už ale je velice přínosné, je detailní popis provádění shlukování. Můžeme tak sledovat, které objekty se v jakém kroku slučují a jak to ovlivňuje následující kroky shlukování.

Podmínka provedení počtu kroků – shlukování je prováděno, dokud není dosaženo zadaného počtu kroků. V případě, že jsme zadali větší počet kroků, než je možné se zadanými daty provést, je shlukování ukončeno po vzniku jednoho shluku, jenž obsahuje všechny objekty. Tato podmínka se dobře kombinuje s již zmíněnou podmínkou jednoho shluku. Taktéž ji lze využít, pokud chceme vytváření jednotlivých shluků krokovat po jednom anebo i více krocích.

Závěrem bych chtěl říci, že vybrané ukončující podmínky nejsou úplně ideální, protože vyžadují po uživateli, aby si sám určil, jaký krok považuje za konečný. Není tedy možné vybrat automaticky nějaký optimální výsledek a ten prezentovat jako výchozí bod, ze kterého by se daly provádět další úpravy a pozorování. Možné řešení tohoto problému je prezentováno v kapitole 8.

5.2.2 Typy dat a jejich zpracování

Hodně častým typem dat uložených v databázích a používaných při následném shlukování jsou intervalové a nominální proměnné. Proto tedy s oběma zmíněnými typy dat umí pracovat i moje popisovaná implementace. Existují samozřejmě i další používané typy proměnných, které byly uvedeny již v kapitole 3.3.2.

Intervalové proměnné se dají definovat jako spojitá měřítka, která jsou rozdělena přibližně lineárně. Tyto proměnné se tedy většinou používají k vyjádření např. zeměpisné šířky a délky, výšky a hmotnosti, počtu nějakých věcí, osob atd. Při zpracovávání těchto proměnných proto musíme brát v potaz i to, v jakých jednotkách je ta daná proměnná uložena. Změna použitých jednotek může totiž znamenat rozdílný postup při shlukování a tím dostaneme i jiný výsledek. Abychom se vyvarovali těmto problémům, tak se nejdříve provádí standardizace dat, jejímž cílem je dát všem proměnným stejnou váhu. Pokud ovšem chceme, aby určité proměnné měly větší váhu, než jiné, tak se standardizace nepoužívá. Jedna z možností je převod proměnných na bezjednotkové za použití např. standardní odchylky. Další možnost je následující:

1) Nejdříve vypočítáme střední odchylku s_f:

$$s_f = \frac{1}{n}\left(\left|x_{1_f} - m_f\right| + \left|x_{2_f} - m_f\right| + \cdots + \left|x_{n_f} - m_f\right|\right),$$

   kde $x_{1_f}, \cdots, x_{n_f}$ je n hodnot proměnné f a m_f je střední hodnota f.

2) Následuje výpočet standardizované hodnoty, neboli z-score:

$$z_{if} = \frac{x_{if} - m_f}{sf}$$

Po provedení standardizace (případně bez ní) jsou data nyní připravena k určení podobnosti anebo odlišnosti jednotlivých objektů. Tato podobnost se u intervalových proměnných nejčastěji určuje za použití výpočtu vzdálenosti mezi všemi objekty. Nejvíce používanou vzdálenostní funkcí je Euklidovská vzdálenost, jež se definuje následovně:

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2},$$

V tomto vzorci jsou $i = (x_{i1}, x_{i2}, \cdots, x_{in})$ a $j = \left(x_{j1}, x_{j2}, \cdots, x_{jn}\right)$ dva n-dimensionální objekty.

Další velice používanou vzdálenostní funkcí je Manhattan vzdálenost a ta se definuje následovně:

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{in} - x_{jn}|$$

Jak Euklidovská vzdálenost, tak i Manhattan vzdálenost splňují následující matematické požadavky pro vzdálenostní funkce:

1) Mezi dvěma objekty je vzdálenost nezáporné číslo: $d(i,j) \geq 0$
2) Vzdálenost jednoho a toho samého objektu je nulová: $d(i,j) = 0$
3) Vypočítaná vzdálenost je vždy symetrická: $d(i,j) = d(j,i)$
4) Přímá vzdálenost mezi objekty i a j není nikdy větší, než součet vzdáleností mezi objekty i, h a objekty j, h (trojúhelníková nerovnost): $d(i,j) \leq d(i,h) + d(h,j)$

Nominální proměnné mohou nabývat jedné, anebo více předem definovaných hodnot. Tyto proměnné jsou v podstatě zobecněním binárních proměnných, které mohou nabývat pouze dvou hodnot. Nominální proměnné se většinou používají k ukládání různých dat, která jsou vyjádřena slovně např. barvy auta, kdy by v databázi bylo uloženo – bílá, černá, červená, zelená, modrá, žlutá, stříbrná atd. Pro výpočet podobnosti dvou objektů, které jsou definované jako nominální proměnné, se používá jednoduchý koeficient shody:

$$d(i,j) = \frac{p - m}{p},$$

kde m je počet shod (počet proměnných, které mají stejnou hodnotu pro objekty i a j) a celkový počet proměnných označuje p. „Abychom tento koeficient více přizpůsobili konkrétním aplikacím, můžeme jednotlivým proměnným navíc přiřadit váhy podle jejich významu. Nebo můžeme přidělit větší váhu shodě proměnné, která může nabývat většího počtu stavů, než ostatní proměnné.“ [1]

5.3 MySQL

MySQL je relační databázový systém typu DBMS (Database Managment Systém). Tato databáze je dostupná jak pod bezplatnou licencí GPL, tak i pod placenou komerční licencí. Tímto způsobem dvojího licencování je považována za průkopníka v dané oblasti. Každá uložená databáze je tvořena z jedné nebo více tabulek, které mají řádky a sloupce. Jednotlivé záznamy jsou rozeznávány jako řádky, tzn. jeden řádek = jeden záznam. Každý sloupec má jméno a definuje jaký typ dat je v daném sloupci uložen. Sloupce tedy tvoří pole záznamů o stejném datovém typu. K implementaci jsou použity programovací jazyky C, C++ a práce s databází probíhá pomocí dotazů, které vychází z deklarativního programovacího jazyka SQL (Structured Query Language). Nespornou výhodou je její multiplatformnost, což znamená, že máme možnost provozovat tuto databázi na různých operačních systémech jako je Windows, Linux, Mac OS X a další. Uživatelé si také cení vysokou stabilitu, protože se vývojáři snaží každou novou verzi důkladně otestovat, před tím než ji vydají. Velice dobrá je i velká podpora přístupu k MySQL, kdy je umožněno přistupovat k databázi z mnoha programovacích jazyků, jedná se např. o C, C++, Java, Perl, Python, PHP atd. K dalším výhodám patří i to, že již od počátku vývoje byla prováděna optimalizace především rychlosti. Tohoto bylo dosaženo za cenu zjednodušení určitých věcí v různých oblastech. S postupným rozvojem ovšem tyto zjednodušené věci mnoha uživatelům začínají chybět, a proto se vývojáři snaží dané oblasti vylepšit. Tak se postupně doplnilo lepší zálohování, přidala se podpora pohledů, triggerů a uložených procedur. Popularita databáze MySQL neustále roste již od jejího vzniku. Z velké míry to způsobil rychlý rozvoj webu a hlavně PHP, které je často kombinované právě s databází MySQL. Propojením

tohoto rozvoje a velkých výhod, jež jsou nabízeny, se tak tato databáze postupně stala jednou z často používaných databází v dnešní době. Pro své potřeby ji využívá nejen hodně známých webů jako je Google, YouTube, Facebook, ale i další firmy.

Databázový systém MySQL vytvořil Michael Widenius a David Axmark v roce 1995, kdy se jeho vlastníkem a sponzorem stává švédská firma MySQL AB. Jméno pravděpodobně vzniklo jako složenina jména My, což je jméno dcery tvůrce Michaela Wideniuse a názvu jazyka SQL, pomocí kterého probíhá komunikace s databází. V roce 2002 proběhla soutěž o pojmenování delfína, který je ve znaku databáze MySQL. Vítězné jméno „Sakila" je ženského rodu a bylo vybráno s více než šesti tisíc možností. Do soutěže ho přihlásil Ambrose Twebaze. Počátkem roku 2008 tuto firmu kupuje firma Sun Microsystems a tak se z MySQL AB stává dceřiná společnost pod vedením Sun Microsystems. Ani ne rok poté, v pondělí 20. dubna 2009, ovšem přichází firma Oracle s oznámením, že se dohodla na odkoupení firmy Sun Microsystems. Během následujících měsíců má dojít k následné fúzi. Otázkou ovšem zůstává, co se stane s databází MySQL, která je přímým konkurentem mnohem dražší alternativy firmy Oracle - Oracle Database. Hodně lidí se tak obává, aby Oracle nezrušil vývoj tolik oblíbené databáze MySQL.

„MySQL je základem mnoha dnešních vysoce kvalitních a robustních databázových řešení. Uživatelé jsou samozřejmě stále náročnější. Bohaté aplikace, které běží pomalu, jsou nepřijatelné bez ohledu na funkce, jež nabízejí. Cenou za škrty v seznamu prací a nákladech za outsourcing je to, že většina přetížených profesionálů má sotva dost času na vytváření minimálního řešení, natož na zvládnutí detailů jejich databázového jádra, které se v případě MySQL stává bohatším a složitějším s každou novou verzí." [4] Jak je z tohoto textu patrné, tak ani samotné použití MySQL nemusí znamenat výhodu při implementaci dané aplikace. Stejně jako u všeho zde platí, že pokud se podcení analýza dané problematiky, tak se tím může zapříčinit vznik skrytých problémů, které se projeví až později. Oprava těchto problémů se následně stává velice nákladnou a mnohdy i obtížně proveditelnou, protože se musí zasáhnout do základních principů, které byly obsaženy v návrhu aplikace.

5.3.1 Tabulky v databázi

Při návrhu struktury uchovávání potřebných dat se musí vzít v potaz několik následujících požadavků. Mělo by být možné ukládat data o různých datových typech s možností jednoznačného určení jaký typ proměnných je právě zpracováván. Pro uchovávání dat se má použít již zmiňovaný databázový systém MySQL, který pracuje s tabulkami. Posledním požadavkem je, aby navrhovaná struktura byla snadno implementovatelná a také, aby nebyla moc složitá.

Základní struktura bude tedy tvořena tabulkou, jež bude obsahovat řádky určené pro jednotlivé záznamy a sloupce, které budou v podstatě reprezentovat pole záznamů jednoho typu proměnných. První sloupec tabulky je typu *varchar*, může mít libovolné jméno a obsahuje názvy daných objektů. Druhým sloupcem začínají data, která se budou zpracovávat při shlukování. Aby se určilo, o jaký typ proměnných se jedná, musí název sloupce obsahovat klíčové slovo. Při ukládání číselných hodnot se jedná o intervalové proměnné, jejichž klíčové slovo je „interval". Textové údaje reprezentují nominální proměnné a jejich klíčové slovo je „nominalni". S těmito typy proměnných umí navržená aplikace pracovat. Pokud by se přidávala podpora dalších typů proměnných, tak je zde potřeba definovat další příslušná klíčová slova. Kromě klíčového slova může název sloupce obsahovat i další text, který vhodně vystihuje uložená data. Na následujícím obrázku 5.2 je vidět ukázka, jak může například vypadat databázová tabulka.

	nazev	barva_nominalni	cislo_interval
	objekt 1	cervena	1
	objekt 2	zelena	2
	objekt 3	modra	3
	objekt 4	cervena	2
	objekt 5	zelena	3
	objekt 6	modra	1
	objekt 7	cervena	3
	objekt 8	zelena	1
	objekt 9	modra	2
*	(NULL)	(NULL)	(NULL)

Obrázek 5.2 Ukázka databázové tabulky

5.3.2 Vkládání dat

Aby bylo možné provádět shlukovou analýzu, tak je potřeba mít nějaká vstupní data, z nichž chceme získat nové znalosti. Většinou jsou tato data již uložena v databázi a jen se případně vybírá, kterou část budeme zpracovávat. Občas ale můžeme chtít vložit, neboli importovat, naše vlastní záznamy. Za tímto účelem bylo vhodné zvolit způsob, jakým bude možné vkládat uživatelem definované údaje do databáze. Tento způsob by měl umožňovat provádění jednoduchého a hlavně rychlého vkládání. Stejně tak je důležité, aby k pochopení principu vkládání dat, nebylo nutné znát nějakou složitou datovou strukturu, za pomocí které by se toto vkládání provádělo.

Jedno z možných řešení daného problému je implementování grafického rozhraní, které by umožňovalo zadání dat přímým způsobem z klávesnice. Navržené řešení je výhodné z toho důvodu, že nevyžaduje vůbec žádnou znalost problematiky databází. Na druhou stranu si je potřeba uvědomit, že by se při poškození databáze nebo při přechodu na jinou databázi musela všechna data zadat znovu. Kromě toho již existuje velké množství vhodných aplikací, které umožňují nejen toto vkládání, ale i další funkce. Jedná se např. o hodně známý phpMyAdmin, Navicat, SQLyog a další.

Nakonec jsem zvolil možnost vkládání dat za pomoci souboru, obsahujícího SQL dotazy. Je to výhodné z důvodu lehké přenositelnosti, bez nutnosti opakovaného zadávání údajů. Existuje také velké množství nástrojů, které umožňují export databázové tabulky do souboru za pomoci SQL dotazů. Všeobecně se dá říci, že jde o soubor obsahující text s libovolnou koncovkou. Vhodné ovšem je použít koncovku, která by odpovídala obsahu souboru a proto je použita koncovka *sql*. Takový soubor tedy obsahuje SQL dotazy, kde každý jednotlivý dotaz musí být ukončen středníkem, za kterým nesmí následovat další dotaz, ale může zde být mezera, tabulátor nebo ukončení řádku. Pro potřeby vkládání dat mohou být využity následující SQL dotazy:

- Vytvoření databázové tabulky:

```
CREATE TABLE jm_tabulky (def_sloupce, … [integritni_omezeni])
```

- Vložení jednoho nebo více záznamů (řádků) do tabulky:

```
INSERT INTO jm_tabulky [(jm_sloupce, … )] VALUES ('hodnota', … )
```

`jm_tabulky` – jméno databázové tabulky.
`def_sloupce` – jednotlivé sloupce tabulky definujeme následovně:

```
jmeno_sloupce typ [impl_hodnota] [seznam_io_sloupce]
```

`integritni_omezeni` – omezení kladená na hodnoty ve sloupcích tabulky, aby nedošlo k porušení integrity dat.

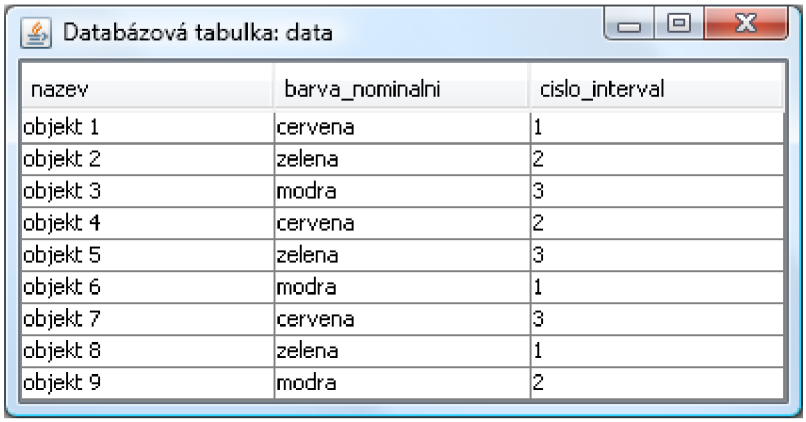
`jm_sloupce` – jméno sloupce v tabulce.
`hodnota` – vkládaná hodnota příslušného sloupce.

Algoritmus pro vkládání dat ze souboru do databáze je umístěn ve třídě *Databaze* (soubor Databaze.java) jako metoda *importData()* a funguje následovně. Nejdříve si otevřeme zvolený soubor a jeho obsah po řádcích ukládáme do proměnné typu *StringBuffer*. Místo znaku konce řádku se ukládá mezera, protože je jedno, jakým způsobem bude obsah formátovaný. Po načtení celého souboru je využito třídy *StringTokenizer*. Této třídě je předána proměnná obsahující načtený soubor a ta ho převede na jednotlivé tokeny. Jeden token je souvislý text oddělený zleva i zprava mezerou. V cyklu jsou potom jednotlivé tokeny skládány za sebe a jsou oddělovány mezerami. Vždy se testuje, jestli daný token na svém konci nemá středník, což značí konec SQL dotazu. V případě nálezu středníku je celý dotaz předán metodě *provedUpdate()* ve třídě *Jdbc* (soubor Jdbc.java), která provede vykonání dotazu nad zvolenou databází.

5.3.3 Zobrazení dat

Jedním ze stanovených cílů této práce je i vhodný návrh způsobu zobrazení dat z databáze. Výrazně se tím ulehčuje práce při provádění shlukování, protože není potřeba mít spuštěný jiný program, jenž by umožňoval prohlížení právě zpracovávaných dat. Tím se dostáváme k prvnímu požadavku na zobrazení dat, kterým je možnost pracovat současně s výpisem z databáze a implementovaným programem. Dále je nutné, aby v samotném výpisu byla zobrazena všechna požadovaná data z databáze, tzn. jméno databázové tabulky, popisky jednotlivých sloupců, názvy objektů a jejich dat. Kromě těchto požadavků by také bylo vhodné mít možnost provést současné zobrazení několika vybraných databázových tabulek.

Zobrazení dat jsem se rozhodl řešit tabulkovou strukturou, čili podobně jak jsou data uložena v databázi. Tento způsob se mi zdá nejpřehlednější. Každý jednotlivý řádek je tvořen jedním záznamem z databáze. První sloupec obsahuje názvy objektů a všechny další sloupce jsou proměnné daného objektu. K zobrazení jsem využil komponentu *JTable*, jenž poskytuje vhodné možnosti při zobrazování tabulek. Pro každou zobrazovanou tabulku je tak vytvořena vlastní instance *JTable*. Toto nám zajistí nejen možnost současného zobrazení více tabulek, ale i nezávislost na programu, protože není nutné zavírat výpis z databáze, aby bylo možné provádět shlukování. Jméno databázové tabulky je pro přehlednost zobrazeno v titulku okna. Na následujícím obrázku 5.3 je vidět ukázka, jak může například vypadat výpis databázové tabulky data:

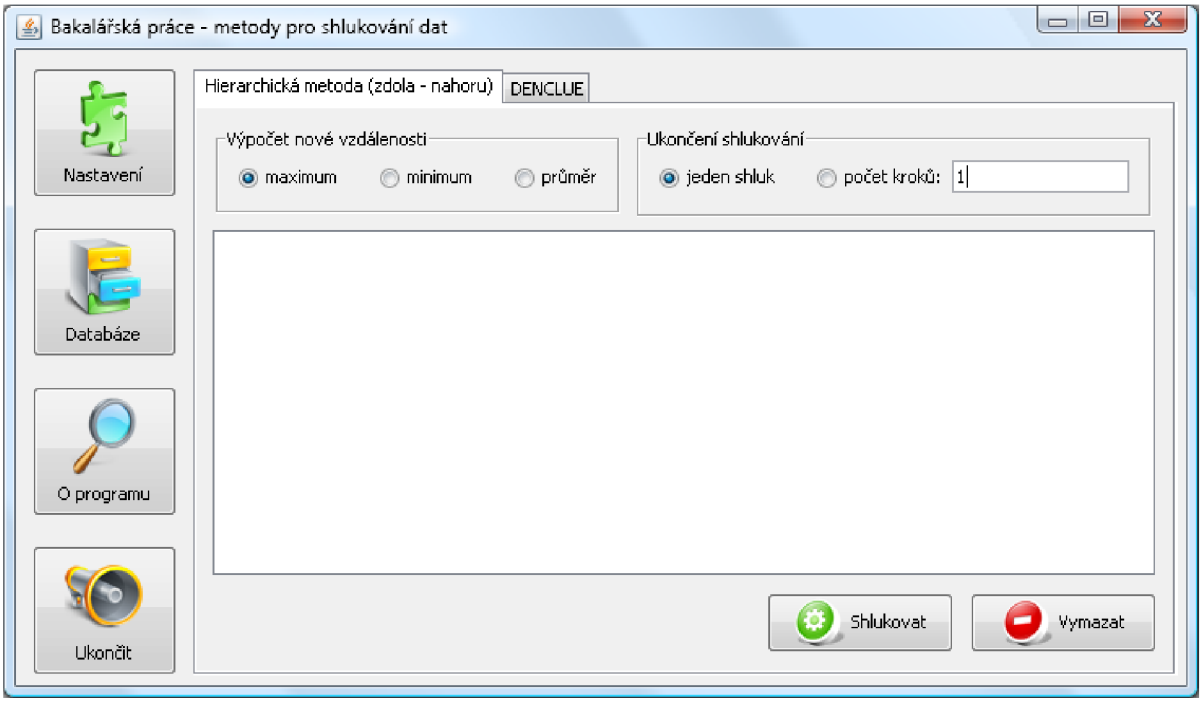


Obrázek 5.3 Ukázka výpisu dat z databáze

Zobrazení databázové tabulky je implementováno ve třídě *Databaze* (soubor Databaze.java) jako metoda *zobrazTabulku()*. Na začátku se nejdříve připojíme k databázi pomocí metody *pripojeni()* a vybereme požadovaná data příkazem `SELECT * FROM zvolena_tabulka`, který ve formě *Stringu* předáme metodě *provedQuery()*. Obě zmíněné metody jsou ze třídy *Jdbc* (soubor Jdbc.java). Metoda *provedQuery()* nám vrátí datovou strukturu *ResultSet*, která obsahuje požadovaná data. Do pole *Stringu* si postupně cyklem *for* načteme názvy jednotlivých sloupců. Pro načtení zbylých záznamů využijeme kolekce *Vector*, jenž bude jako jednotlivé *elementy* obsahovat řádky tabulky. Vzniknou nám tak dva objekty, jeden obsahuje názvy sloupců a druhý data z tabulky. Tyto objekty předáme konstruktoru *JTable* a ten z nich vytvoří stejnou tabulkovou strukturu jaká je v databázi. Již zbývá jen specifikovat rozměry okna pomocí metody *setBounds()*, nastavit titulek okna metodou *setTitle()* a celé výsledné okno zviditelnit metodou *setVisible()*.

5.4 Návrh a popis GUI

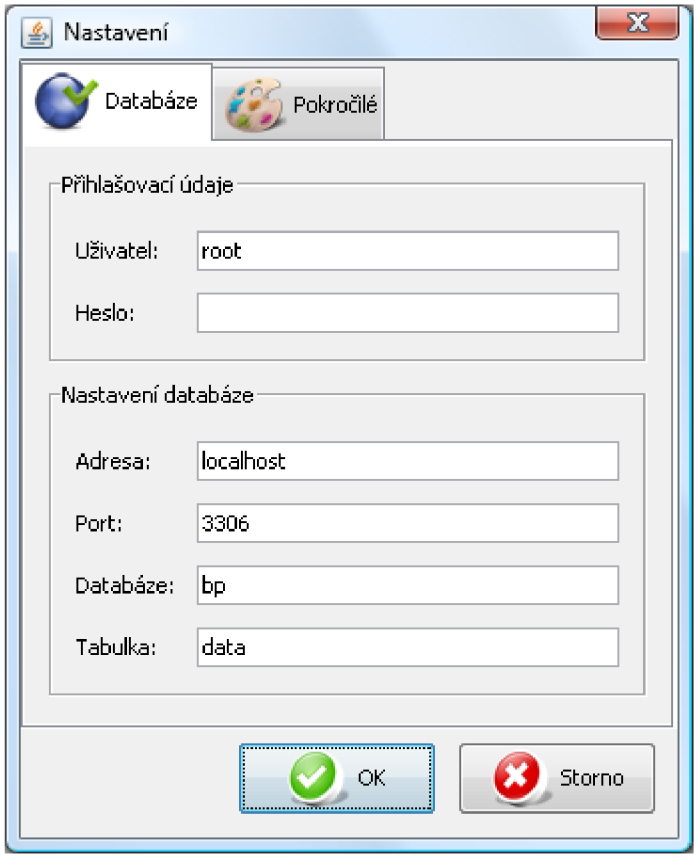
Uživatelské rozhraní (anglicky Graphical User Interface) je velice důležitá část každého programu, protože tvoří jakési spojení mezi uživatelem a funkcemi, které jsou v programu k dispozici. Při návrhu bylo nutné brát ohled na to, že v programu budou obsaženy dvě různé shlukovací metody, kdy každá může obsahovat jiné možnosti nastavení vstupních parametrů. Důležité je, aby bylo možné později doplňovat další shlukovací metody, bez toho aniž bychom museli provést nějaké složité úpravy uživatelského rozhraní. Při návrhu jsem bral ohled na všechny požadavky vzniklé při implementaci. Na následujícím obrázku 5.4 je vidět základní okno programu po jeho spuštění:



Obrázek 5.4 Hlavní okno programu po spuštění

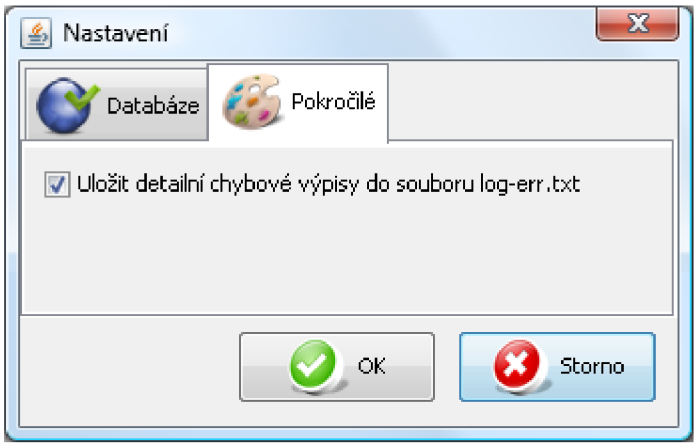
Základní okno programu je rozděleno na dvě hlavní části - levou a pravou. Levá část je vždy viditelná a obsahuje tlačítko *nastavení*, které zobrazí nové okno s nastavením programu (viz níže obrázek 5.5 a 5.6). Pod ním je tlačítko *databáze*, jež zobrazí nové okno umožňující práci s databází (viz níže obrázek 5.7). Následuje tlačítko *o programu*, které zobrazí informace o tomto programu.

Poslední tlačítko *ukončit* provede ukončení celého programu. V pravé části hlavního okna se nachází záložky s jednotlivými shlukovacími metodami. Použitím záložek je zajištěna možnost rozšíření daného programu o další shlukovací metody, protože zde pouze stačí přidat další záložku a do ní zvolenou shlukovací metodu. Každá záložka potom obsahuje vlastní možnosti nastavení vstupních parametrů a pole pro výpis informací o shlukování. Je tak možné provádět shlukování u více metod současně a získané výsledky mezi nimi porovnávat. Záložka *hierarchická metody (zdola – nahoru)* obsahuje dva vstupní parametry. Jedná se o *výpočet nové vzdálenosti*, který určuje, jak se bude počítat vzdálenost u nově vzniklého shluku a *ukončení shlukování*, jehož pomocí specifikujeme ukončovací podmínku pro shlukování. Pod parametry se nachází prázdné textové pole, do kterého se vypisují informace o shlukování. Toto pole se dá zvětšit tím, že zvětšíme okno celého programu. Výpis se dá vyčistit tlačítkem *vymazat* a tlačítkem *shlukovat* započneme provádění shlukovací metody.



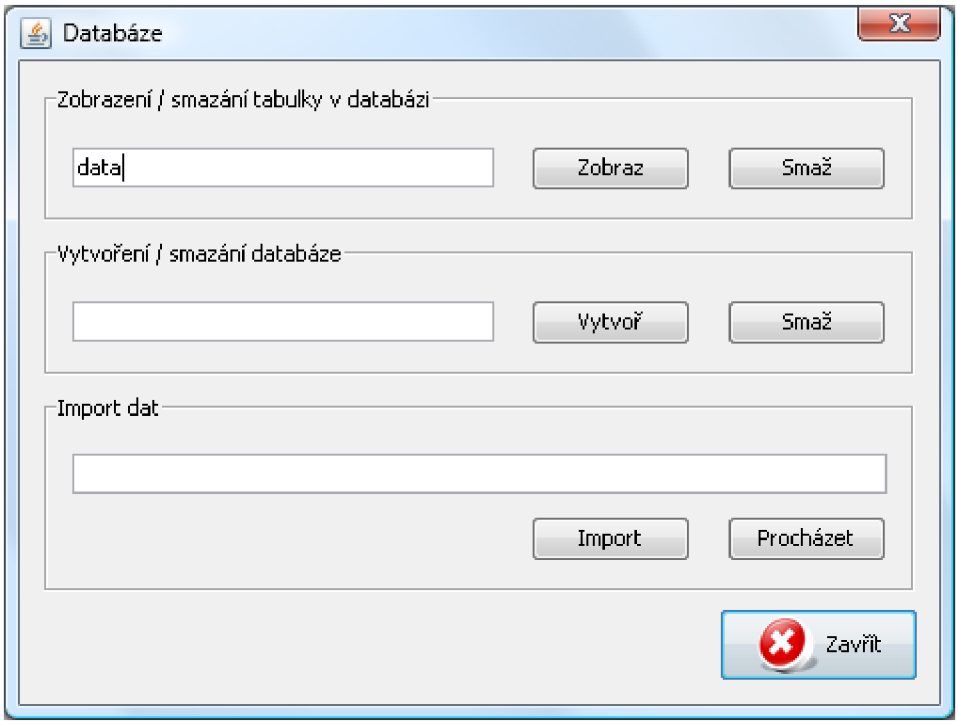
Obrázek 5.5 Nastavení programu – záložka databáze

Pokud v hlavním oknu programu stiskneme tlačítko *nastavení*, tak se nám zobrazí nové okno s nastavením, jak je vidět na obrázku 5.5. Toto okno může být otevřené nezávisle na hlavním oknu. Je zde možné specifikovat parametry databáze, které se použijí k připojení. V rámečku *přihlašovací údaje* se zadává uživatelské jméno a příslušné heslo. Rámeček *nastavení databáze* obsahuje adresu databáze, port, jméno příslušné databáze a databázové tabulky, se kterou budeme pracovat. Tlačítkem *ok* provedeme potvrzení zadaných údajů, tzn. jejich uložení a zavření okna nastavení. Pokud bychom použili tlačítko *storno*, tak se pouze zavře okno nastavení, bez ukládání provedených změn.



Obrázek 5.6 Nastavení programu – záložka pokročilé

Na předchozím obrázku 5.6 je vidět okno nastavení po zobrazení záložky *pokročilé*, které bylo pro lepší přehlednost zmenšeno. Vybraná záložka obsahuje podrobnější nastavení programu. V současné verzi je zde pouze možnost nechat si vypisovat podrobné chybové hlášení do souboru log-err.txt. Tento soubor se uloží do složky, ze které je aplikace spuštěna a ukládá se do něj datum, čas a detailní chybové hlášení vždy, když v aplikaci nastane nějaká chyba. Kromě tedy stručného výpisu, který se zobrazí přímo v aplikaci, je i možnost prohlédnout si detailní výpis v souboru.

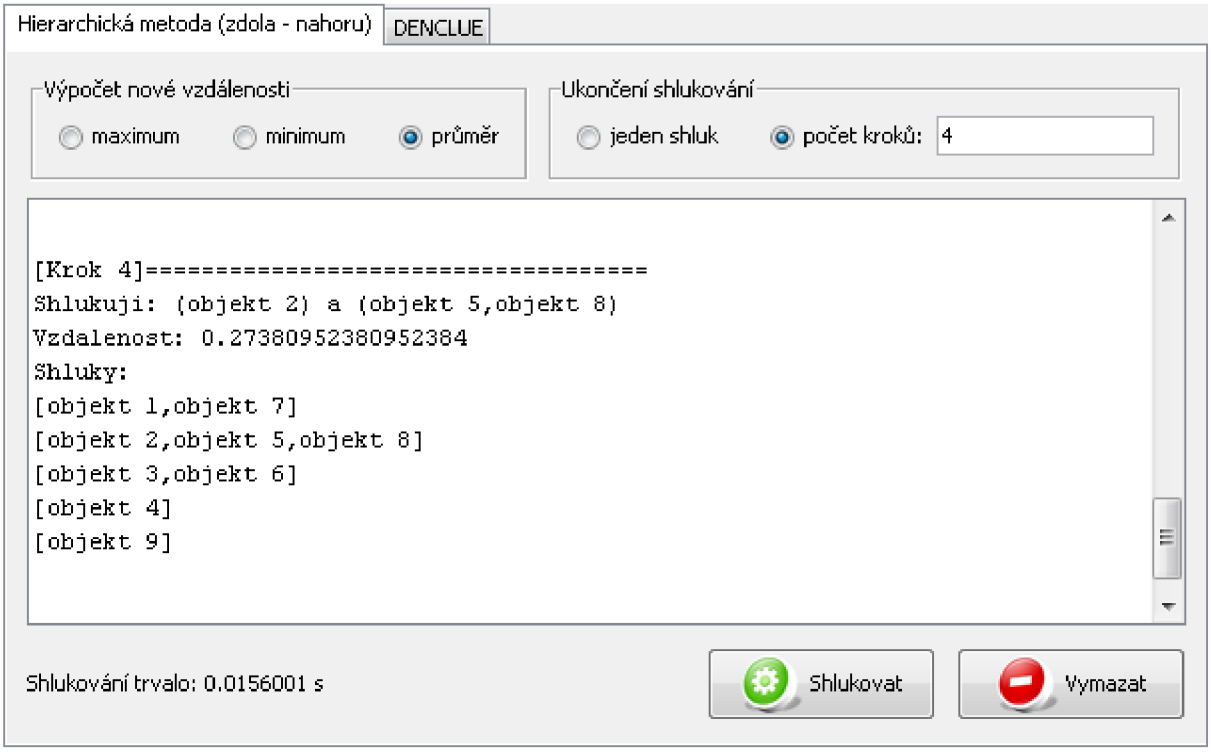


Obrázek 5.7 Práce s databází

Po stisknutí tlačítka *databáze* v hlavním okně programu, se zobrazí nové okno, které je vidět na obrázku 5.7. Okno je rozděleno na tři části. První část umožňuje zobrazit anebo smazat zadanou databázovou tabulku. Tlačítko *zobraz* otevře nové okno, podobné jako je na obrázku 5.3, jenž obsahuje výpis dat ze zadané tabulky. Ve druhé části je umožněno vytváření a mazání celé databáze. Poslední část slouží k importování dat ze souboru, který můžeme zvolit buď pomocí tlačítka *procházet* anebo můžeme zadat cestu ručně do připraveného textového pole.

5.4.1 Zobrazení výsledků

Důležitou částí při návrhu uživatelského rozhraní bylo také navržení vhodného způsobu zobrazení výsledků prováděného shlukování. Snažil jsem se, aby moje zvolené řešení bylo co nejvíce přehledné a obsahovalo všechny důležité informace. Ukázka výpisu je zobrazena níže na obrázku 5.8. Je zde zobrazen čtvrtý krok shlukování. Každý výpis jednotlivého kroku tedy nejdříve začíná číslem daného kroku. Za ním následuje výpis dvou objektů, které se v tomto kroku shlukují, kde každý objekt je uzavřen do kulatých závorek. Následuje výpis vzdálenosti mezi slučovanými objekty. Jako poslední se vypisuje aktuální přehled všech shluků, kde každý řádek obsahuje právě jeden shluk. Po skončení shlukování se vypíše čas (v sekundách), který potřeboval procesor na výpočet celého shlukování.



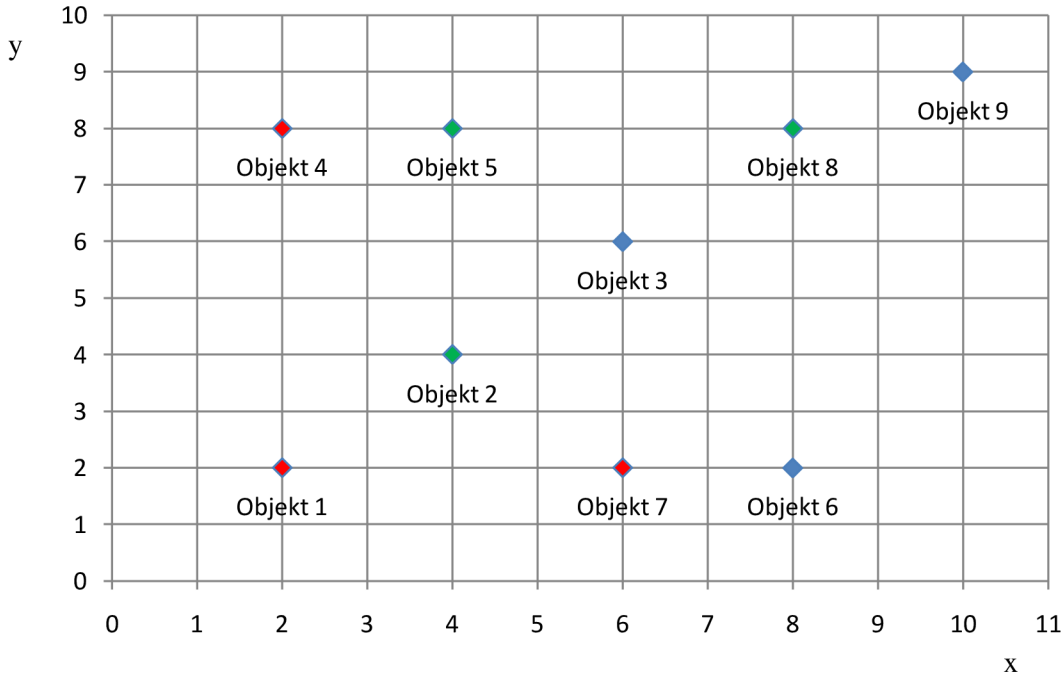
Obrázek 5.8 Zobrazení výsledků shlukování

6 Testování

Důležitou částí životního cyklu programu je fáze testování, kterou není možné vynechat. Pokud bychom tuto fázi zanedbali nebo úplně vynechali, tak by se to projevilo na chybné funkčnosti programu a na jeho možných neočekávaných pádech. Dobře navržený program je bez ověření funkčnosti zbytečný a neužitečný. Důkladné testování obvykle zabere velkou část z práce na programu, je velice důležité a mnohdy se provádí po celou dobu životnosti programu. Jediným cílem této fáze je nalezení chyb a jejich opravení. Při opravování chyb je velká pravděpodobnost, že se do programu zavedou chyby nové a proto je potřeba všechny provedené opravy důkladně testovat.

Moje testování probíhalo od počátku vývoje. Každá nová metoda byla nejdříve důkladně otestována, aby se zamezilo vzniku možných chyb v jednotlivých metodách. Dále jsem prováděl testování tříd jako samostatných celků, aby nedocházelo k jejich vzájemnému ovlivňování, a potom jsem teprve testoval všechny třídy současně. Po otestování funkčnosti uživatelského rozhraní jsem opět prováděl testování jako celku. Za tímto účelem jsem využil implementace metody DENCLUE, se kterou jsem porovnával výsledky. Testy byly prováděny na vzorových příkladech z bakalářské práce pana Bc. Radima Kapavíka a na vstupních datech o velikosti několika stovek objektů, jenž jsem za tímto účelem vytvořil.

Na následujícím obrázku 6.1 je zobrazena grafická reprezentace dat u jednoho prováděného testu. Z důvodu lepší přehlednosti a názorné ukázky byl počet objektů v databázi zredukován na malé množství. Soubor se vstupními daty je obsahem přílohy č. 1 a obsahuje tedy 9 různých objektů, kde každý jednotlivý objekt má specifikovanou barvu a pozici na souřadnicové ose *x* a *y*.



Obrázek 6.1 Grafická reprezentace testovaných dat

Z obrázku je patrné, že by výsledkem shlukování měly být tři shluky, kde každý shluk reprezentuje jeden typ barvy – červenou, zelenou a modrou. V programu zvolíme příslušná vstupní data z databáze a určíme, že chceme provádět výpočet nové vzdálenosti jako průměrnou hodnotu

z původních vzdáleností obou shluků. Požadovaného výsledku dosáhneme v šestém kroku shlukování, jak je vidět na níže uvedeném výpisu informací z programu.

```
[Krok 1]==================================
Shlukuji: (objekt 1) a (objekt 7)
Vzdalenost: 0.16666666666666666
Shluky:
[objekt 1,objekt 7]
[objekt 2]
[objekt 3]
[objekt 4]
[objekt 5]
[objekt 6]
[objekt 8]
[objekt 9]


[Krok 2]==================================
Shlukuji: (objekt 5) a (objekt 8)
Vzdalenost: 0.16666666666666666
Shluky:
[objekt 1,objekt 7]
[objekt 2]
[objekt 3]
[objekt 4]
[objekt 5,objekt 8]
[objekt 6]
[objekt 9]


[Krok 3]==================================
Shlukuji: (objekt 3) a (objekt 6)
Vzdalenost: 0.2738095238095238
Shluky:
[objekt 1,objekt 7]
[objekt 2]
[objekt 3,objekt 6]
[objekt 4]
[objekt 5,objekt 8]
[objekt 9]
```

```
[Krok 4]==================================
Shlukuji: (objekt 2) a (objekt 5,objekt 8)
Vzdalenost: 0.27380952380952384
Shluky:
[objekt 1,objekt 7]
[objekt 2,objekt 5,objekt 8]
[objekt 3,objekt 6]
[objekt 4]
[objekt 9]


[Krok 5]==================================
Shlukuji: (objekt 3,objekt 6) a (objekt 9)
Vzdalenost: 0.36309523809523814
Shluky:
[objekt 1,objekt 7]
[objekt 2,objekt 5,objekt 8]
[objekt 3,objekt 6,objekt 9]
[objekt 4]


[Krok 6]==================================
Shlukuji: (objekt 1,objekt 7) a (objekt 4)
Vzdalenost: 0.36904761904761907
Shluky:
[objekt 1,objekt 7,objekt 4]
[objekt 2,objekt 5,objekt 8]
[objekt 3,objekt 6,objekt 9]
```

6.1 Reálný vzorek dat

Jako reálný vzorek dat jsem zvolil vybraných 85 filmů z internetové databáze dostupné na adrese http://www.csfd.cz. Každý záznam je tvořen názvem filmu, rokem vzniku, délkou v minutách a poslední dva sloupce obsahují žánr filmu, vystihující obsah filmu. Soubor se vstupními daty *dataTest2.sql* je z důvodu velikosti uložen na CD, které je obsahem přílohy č. 2.

Shlukování vybraných vstupních dat je zaměřeno na zjišťování podobných filmů. Získané údaje mají široké možnosti použití. V oblasti obchodu se tohoto dá využít např. vkládáním reklamních letáků do prodávaných filmů na DVD. Zákazník si zakoupí např. rodinný film a pomocí vložené reklamy bude upozorněn na další rodinné anebo jiné, velmi podobné filmy. Další možné využití je např. v kinech, půjčovnách filmů nebo v televizních upoutávkách. Stejně tak návštěvníci již zmiňované internetové databáze by určitě ocenili možnost doporučení filmu, jenž je podobný zvolenému filmu. Často se stává, že zhlédneme film, který se nám líbil, a chtěli bychom vidět jiný s podobnou tématikou.

Zjišťování podobnosti mezi filmy je závislé na množství dat, které poskytneme pro zpracování. Já jsem použil např. pouze dva žánry, které co nejlépe vystihují film. V některých případech to může být málo, protože se nedá celý film zařadit jen do dvou žánrů. Rozšířením žánrů

by se zpřesnil výsledek. Níže je ukázán výpis prvních sedmi kroků shlukování. Jsou zobrazeny pouze filmy, které se slučují, protože celý výpis jednotlivých kroků je moc dlouhý.

```
[Krok 1]==================================
Shlukuji: (Revolutionary Road) a (Slumdog Millionaire)


[Krok 2]==================================
Shlukuji: (Frost/Nixon) a (Valkyrie)


[Krok 3]==================================
Shlukuji: (Bride Wars) a (New in Town)


[Krok 4]==================================
Shlukuji: (The Kingdom) a (Fracture)


[Krok 5]==================================
Shlukuji: (Zack and Miri Make a Porno) a (Definitely, Maybe)


[Krok 6]==================================
Shlukuji: (The Hurt Locker) a (Defiance)


[Krok 7]==================================
Shlukuji: (Madagascar: Escape 2 Africa) a (Kung Fu Panda)

                ⋮                ⋮                ⋮

[Krok 84]=================================
```

V prvním kroku je vidět sloučení dvou romantických dramat, druhý krok spojuje dvě historická dramata, ve třetím kroku se jedná o romantické komedie, čtvrtý krok zahrnuje akční thrillery, v pátém kroku jsou sloučeny romantické komedie, šestý krok obsahuje válečná dramata a sedmý krok spojuje rodinné komedie. Slučování probíhá až do kroku 84, kdy je vytvořen jeden velký shluk. Tento výsledek je už pro nás nepoužitelný, protože nás zajímají jen jednotlivé skupiny podobných filmů. Pozorováním shlukování od začátku získáváme přehled o nejpodobnějších filmech, čím více se blížíme konci, tím je větší riziko rozdílnosti filmů.

7 Porovnání s metodou DENCLUE

Jedním z cílů této práce je porovnat mnou implementovaný algoritmus shlukující hierarchické metody (zdola-nahoru), s metodou DENCLUE, kterou implementoval pan Bc. Radim Kapavík. Nejedná se pouze o porovnání těchto implementací, ale hlavně o demonstraci celkového shrnutí obou vybraných metod, jejich rozdílů, výhod a nevýhod.

7.1 Kvalita shlukování

Kvalita výsledných shluků se u obou metod liší v závislosti na vstupních datech a zvolených parametrech. DENCLUE jako metoda založená na hustotě umožňuje nacházet shluky o různých tvarech a umí dobře zpracovávat data, jež obsahují velké množství šumu. Kvalita shluků se u této metody odvíjí od zvoleného parametru hustoty σ a prahu šumu ε. Pokud jsou tyto parametry zvoleny nevhodně, tak to významně ovlivňuje výsledek shlukování. U shlukující hierarchické metody ovlivňuje kvalitu shlukování výběr objektů, které se mají sloučit. Po sloučení dvou objektů již nelze toto rozhodnutí vzít zpět, pokud by se ukázalo, že bylo vhodnější provést sloučení jiných dvou objektů. Výsledek shlukování je taktéž ovlivněn volbou způsobu výpočtu vzdálenosti u nového shluku. Při použití maxima nebo minima je tato metoda citlivá na odlehlé objekty a šum.

Výhodou shlukující hierarchické metody je možnost vytvoření jednoho shluku nebo provedení zadaného počtu kroků za účelem následného pozorování průběhu celého shlukování. Jednou z nevýhod je ovšem časová složitost výpočtu, která je alespoň $O(n^2)$, kde n je celkový počet objektů ve vybrané databázi. Z tohoto důvodu není vhodné používat tuto metodu na rozsáhlejší databáze, protože by výpočet trval velmi dlouho. Další nevýhoda je ta, že nelze měnit jednotlivé objekty mezi shluky, ani rozdělit již sloučené objekty. Výhodou metody DENCLUE je její rychlost, díky které se velice dobře hodí na zpracovávání velkého množství dat. Další výhoda je, že je tato metoda postavená na matematickém základě a tím pádem umožňuje matematicky popsat shluky libovolného tvaru. Nevýhodou je ovšem nutnost zvolení vhodných vstupních parametrů, které ovlivňují výsledek shlukování.

7.2 Závislost času na velikosti dat

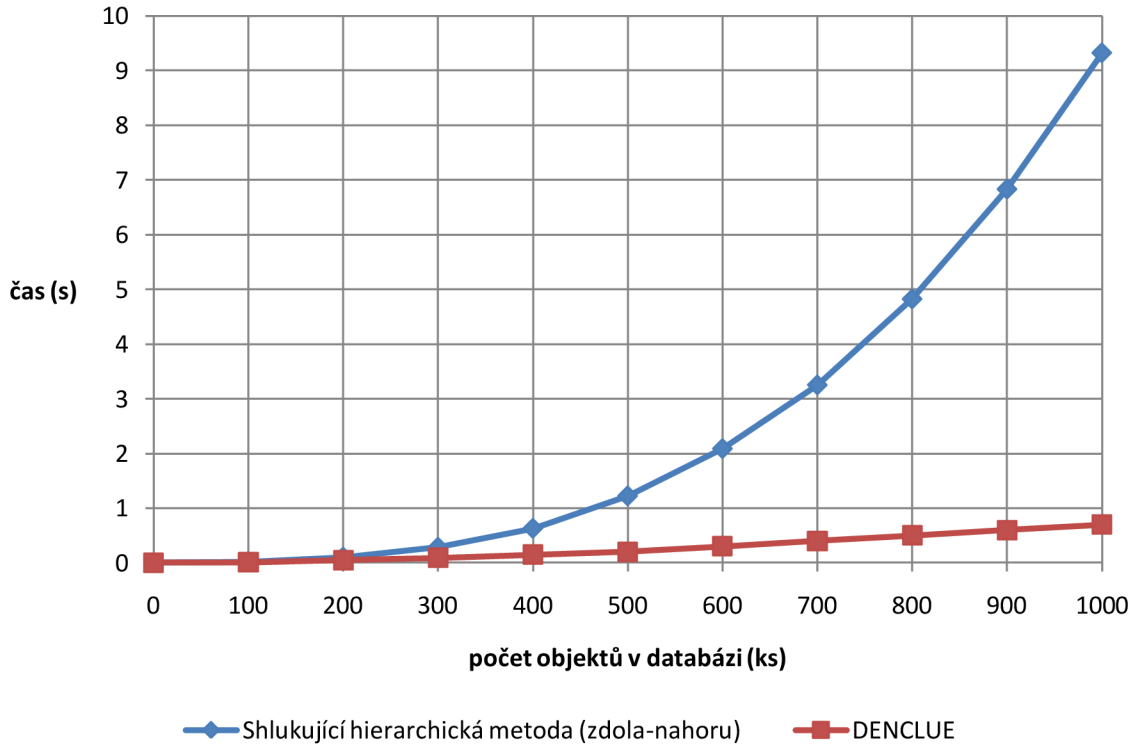
Shlukovací metody uplatňují různé postupy při získávání znalostí z databází. Jednou z vlastností těchto postupů je i rychlost provádění dané metody v závislosti na velikosti zpracovávaných dat. Toto kritérium rychlosti bývá často bráno v potaz při výběru vhodnosti metody na vybraný problém.

Při zjišťování závislosti času na velikosti dat jsem narazil na problém u metody DENCLUE. Tato metoda je implementována v programovacím jazyce C++ a pro uložení dat využívá textového souboru. Toto ovlivňuje rychlost zpracovávání, protože práce s textovým souborem je rychlejší než práce s databází MySQL a programovací jazyk C++ je také rychlejší než jazyk Java. Výsledný průběh závislosti času na velikosti dat to ovšem ovlivňuje pouze v hodnotách na časové ose. Při použití stejného programovacího jazyka a práce s databází by byl výsledný průběh stejný, změnily by se pouze časové hodnoty u vybraných metod. Aplikace obsahující metodu DENCLUE také přímo neumožňuje měřit čas, který procesor potřebuje k provedení shlukování. Proto je čas u této metody

zkreslen, ovšem toto zkreslení je u každého kroku stejné, takže celkový průběh není ovlivněn. Při lepším měření by došlo pouze ke zpřesnění časových údajů.

Měření hodnot, použitých při vytváření grafu na obrázku 7.1, jsem prováděl v deseti krocích. V prvním kroku jsem vložil do databáze 100 objektů a pro tento vzorek dat jsem u každé metody změřil čas, který procesoru zabralo shlukování. Měření času jsem provedl desetkrát po sobě a jako výslednou hodnotu jsem použil průměr z těchto měření. S každým dalším krokem jsem do databáze přidal 100 objektů a provedl stejné měření času jako u prvního kroku. Tímto způsobem jsem získal pro každou metodu 10 průměrných časových údajů, jež jsem následně zobrazil do grafu.

Následující obrázek 7.1 tedy zobrazuje graf závislost času na velikosti dat u porovnávaných metod. Na ose *y* je zobrazen čas v sekundách, jenž potřeboval procesor k vykonání shlukování. Osa *x* představuje počet objektů v databázi, které jsou vstupem shlukování. Z tohoto grafu je patrná jedna z nevýhod hierarchický metod, jenž spočívá v již zmíněné časové složitosti výpočtu. U metody DENCLUE je naopak vidět její dobrá škálovatelnost při zpracovávání většího množství dat.



Obrázek 7.1 Graf zobrazující závislost času na velikosti dat u porovnávaných metod

8 Závěr

Cílem této bakalářské práce bylo provést implementaci vybrané shlukující hierarchické metody (zdola-nahoru), navrhnout pro ni vhodné uživatelské rozhraní, které bude obsahovat dříve implementovanou metoda DENCLUE a nakonec obě tyto metody porovnat.

K dosažení tohoto cíle bylo nutné si nejdříve prostudovat obecnou problematiku získávání znalostí z databází a následně se zaměřit na její statistickou část – shlukovou analýzu. Ve shlukové analýze jsem se seznámil s různými metodami, jejich možnostmi a principy. Důraz jsem kladl na prostudování hierarchických metod. Před návrhem aplikace bylo nutné se zaměřit na programovací jazyk Java, databázový systém MySQL a možnosti jejich spolupráce. S těmito znalostmi jsem mohl provést implementaci zvolené shlukující hierarchické metody (zdola-nahoru). Navrhl jsem vhodnou strukturu databázových tabulek pro ukládání vstupních údajů, způsob vkládání dat do těchto tabulek a jejich možné zobrazení v aplikaci. Pro tyto účely bylo implementováno uživatelské rozhraní, do kterého byla zahrnuta implementace metody DENCLUE. V tomto rozhraní se zobrazují výsledky shlukování ve formě, kterou jsem vytvořil.

Důležitou součástí práce bylo ověření funkčnosti implementované metody. Za tímto účelem jsem provedl sérii testů, jejichž výsledky jsem porovnal s implementací metody DENCLUE. Dospěl jsem k závěru, že mnou navržená implementace funguje správně.

Úkolem práce bylo demonstrovat funkčnost shlukující hierarchické metody v praxi a její porovnání s metodou DENCLUE. Výhodou mnou implementované metody je možnost zobrazení celého průběhu shlukování nebo výběru jen zvoleného počtu kroků. Nevýhodou je, že není možné navrhnout optimální řešení daného problému. Výsledky je proto nutné analyzovat a rozhodnout, který krok shlukování považujeme za konečný. Metodu není vhodné používat pro velká množství dat, z důvodu její výpočetní složitosti. Výhodou metody DENCLUE je navržení optimálního řešení, ze kterého je možné následně vycházet a možnost dobré práce s velkým množstvím dat. Nevýhodou je, že metoda vyžaduje zadání vstupních parametrů hustoty σ a prahu šumu ε. Špatnou volbou těchto parametrů dochází ke zhoršení výsledků shlukování.

Budoucí možné rozšíření práce spočívá v přidání podpory více typů zpracovávaných proměnných. Pro lepší analýzu výsledků by bylo dobré přidat grafické zobrazení dendrogramu, ze kterého lze sledovat celý průběh zvoleného shlukování. Tímto lze částečně vyřešit problém nemožnosti navržení optimálního řešení. Další vylepšení se může vztahovat ke vkládání dat do databáze. Kromě importování celých tabulek by byla možnost úpravy jednotlivých záznamů a exportování těchto tabulek do souboru.

Literatura

[1] Zendulka, J., Bartík, V., Lukáš, R., Rudolfová, I.: Získávání znalostí z databází. Studijní opora, 2006, Fakulta informačních technologií VUT v Brně.

[2] Berka, P.: Dobývání znalostí z databází. Academia, Praha, 2003. ISBN 80-200-1062-9.

[3] Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Second Edition. Morgan Kaufmann Publishers, 2006. ISBN 13: 978-1-55860-901-3, ISBN 10: 1-55860-901-6.

[4] Schneider, R. D.: MySQL Oficiální průvodce tvorbou, správou a laděním databází. Grada Publishing, a.s., Praha, 2006. ISBN 80-247-1516-3.

[5] Kapavík, R.: Získávání znalostí z dat – shlukovací algoritmy. Bakalářská práce, 2008, Fakulta informačních technologií VUT v Brně.

Seznam příloh

Příloha 1. Soubor vstupních dat pro testování v kapitole 6.
Příloha 2. CD se zdrojovými texty, spustitelnou aplikací a vzorkem vstupních dat.

Příloha 1:

Soubor vstupních dat pro testování v kapitole 6

```
CREATE TABLE dataTest1
(
  nazev             VARCHAR(100)  NOT NULL,
  barva_nominalni   VARCHAR(100)  NOT NULL,
  x_interval        INT           NOT NULL,
  y_interval        INT           NOT NULL
);

INSERT INTO dataTest1 VALUES('objekt 1','cervena','2','2');
INSERT INTO dataTest1 VALUES('objekt 2','zelena','4','4');
INSERT INTO dataTest1 VALUES('objekt 3','modra','6','6');
INSERT INTO dataTest1 VALUES('objekt 4','cervena','2','8');
INSERT INTO dataTest1 VALUES('objekt 5','zelena','4','8');
INSERT INTO dataTest1 VALUES('objekt 6','modra','8','2');
INSERT INTO dataTest1 VALUES('objekt 7','cervena','6','2');
INSERT INTO dataTest1 VALUES('objekt 8','zelena','8','8');
INSERT INTO dataTest1 VALUES('objekt 9','modra','10','9');
```