Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačního a znalostního inženýrství

Obor: Znalostní technologie

Diplomová práce

Repozitář nalezených výsledků úloh dobývaní asociačních pravidel v projektu SEWEBAR

Diplomant: Bc. Tomáš Marek
Vedoucí práce: doc. Ing. Milan Šimůnek, Ph.D.

Praha 2012

Prohlášení

Prohlašuji, že jsem vypracoval samostatně diplomovou práci na téma *Repozitář nalezených výsledků úloh dobývaní asociačních pravidel v projektu SEWEBAR*. Použitou literaturu a další podkladové materiály uvádím v přiloženém seznamu literatury.

V Praze dne 10. prosince 2012

.....

podpis diplomanta

Poděkování

Rád bych poděkoval doc. Ing. Milanu Šimůnkovi, Ph.D. za cenné rady a vedení práce. Dále bych rád poděkoval Ing. Tomáši Kliegrovi, Ph.D. za konzultace k zapojení aplikace I:ZI Repository do systému SEWEBAR. Také bych rád vyjádřil díky své rodině a přítelkyni za poskytnutou podporu a zázemí po dobu mého studia.

Abstrakt

Tato diplomová práce se zaměřuje na návrh a implementaci aplikace I:ZI Repository. Aplikace I:ZI Repository poskytuje správu úložiště úloh dobývání znalostí z databází a jejich výsledků a funkce pro prohledávání tohoto úložiště. I:ZI Repository je REST API postavené na Java EE technologii, pro ukládání úloh DZD je použita Berkeley XML databáze. I:ZI Repository vznikla jako nová verze starší aplikace XQuery search. Oproti předchozí aplikaci došlo ke kompletnímu přepracování struktury aplikace s důrazem na zachování funkčnosti z předchozí aplikace. Dále byly přidány možnosti zadání obecnějšího vyhledávacího dotazu, přidány fuzzy přístupy do vyhledávání a možnost shlukovat výsledky vyhledávání. Součástí implementace je i vylepšené logování chodu aplikace zaměřené na zaznamenávání příchozích vyhledávacích dotazů a odchozích výsledků vyhledávání. Součástí práce jsou i výsledky testování aplikace.

Klíčová slova: Dobývání znalostí z databází, vyhledávání, fuzzy, shlukování, Java EE, Berkeley XML DB, REST, API, XML, Spring Framework

Abstract

This diploma thesis aims at design and implementation of I:ZI Repository application. I:ZI Repository application provides management of data mining tasks and theirs results repository and functions for search in this repository. I:ZI Repository is a REST API build on top of Java EE technology, Berkeley XML database is used for storing data mining tasks. I:ZI Repository application was created based on XQuery search application. The application has completely new structure compared to XQuery search application, all functionality of XQuery search application is present in I:ZI Repository application. Possibilities of using more general search query was added into I:ZI Repository application as well as fuzzy approaches for searching and possibility of clustering search results. Enhanced logging of application activities aimed at logging incoming search queries and outgoing search results is a part of implementation. Results of application testing are included as well.

Keywords: Data mining, searching, fuzzy, clustering, Java EE, Berkeley XML DB, REST, API, XML, Spring Framework

Obsah

Úvod

Dobývání znalostí z databází je netriviální proces. S procesem dobývání znalostí z databází jsou spojené náročné výpočetní operace, protože úlohy dobývání znalostí z databází jsou často složité.

Z tohoto pohledu se zdá být zbytečné provádět ty stejné úlohy DZD stále znovu a dokola. Efektivnější by bylo výsledky úloh, které již proběhly, ukládat do databáze úloh DZD. Před spuštěním úlohy DZD by uživatel měl možnost tuto databázi prohledat a zjistit, zda již stejná úloha DZD jako právě zadaná neproběhla. V případě, že taková úloha DZD v databázi již existuje, uživatel by měl možnost zjistit, jaké výsledky poskytla.

Touto problematikou se zabývala aplikace XQuery search vytvořená v rámci mé bakalářské práce *Využití XML databází pro zpřístupnění specifikací úloh dobývání znalostí z databází*. Tato aplikace byla schopna prohledávat výsledky úloh DZD uložené v XML databázi.

Původní aplikace poskytovala pouze funkcionalitu prohledávání výsledků úloh DZD. Velkým problémem původní aplikace byla výkonnost, z tohoto důvodu byla prakticky nepoužitelná pro větší objemy dat. Tento problém byl ovšem při vývoji pokračujícím i po bakalářské práci z větší části vyřešen.

Původní aplikace byla také navržena bez větších ohledů na možné budoucí rozšiřování funkcionality. To také vedlo k náročné údržbě aplikace a složitým opravám chyb.

V rámci této diplomové práce vznikla aplikace, která vychází z aplikace XQuery search. Funkcionalita aplikace XQuery search zůstane v nové aplikaci zachována, především však dojde k implementaci nových funkcí. Vzhledem k rozšíření funkcionality a kompletnímu přepracování a k těsné vazbě na aplikaci I:ZI Miner je nová aplikace nazvána I:ZI Repository. Tento název lépe vystihuje účel aplikace – úložiště úloh dobývání znalostí z databází a jejich výsledků a vyhledávání v nich.

Cíle práce

Tato diplomová práce si stanovuje několik hlavních cílů:

- Umožnit obecnější vyhledávání v databázi úloh DZD
- Zavést fuzzy přístupy do vyhledávání v databázi úloh DZD
- Umožnit shlukování výsledků vyhledávání

- Provést úpravy původní aplikace XQuery search pro umožnění snadných budoucích úprav nové aplikace I:ZI Repository, jejích oprav a zavádění vylepšení a nových funkcí

Výsledkem prvního cíle bude možnost vyhledávat v databázi úloh DZD pomocí obecnějšího zadání vyhledávacího dotazu. K tomu bude nezbytné umožnit vyhledávat v zadáních úloh DZD, nejen v jejich výsledcích.

Druhý cíl umožní vznik nové funkcionality – zavedení fuzzy přístupů do vyhledávání. Pro toto vylepšení je položen základ v implementaci spadající do čtvrtého cíle práce. Zavedení fuzzy přístupů do vyhledávání umožní získávat výsledky vyhledávání s uvedením ohodnocení jejich podobnosti vyhledávacímu dotazu. Zároveň by ohodnocení výsledku vyhledávání mělo reflektovat jeho syntaxi a další vlastnosti relevantní pro vyhledávání.

Třetí cíl je zaměřen na shlukování výsledků vyhledávání. Tato implementace umožní shlukovat výsledky vyhledávání podle uživatelem zvolených aspektů či podle aspektů odvozených z vlastností výsledků vyhledávání.

V rámci čtvrtého cíle práce je nutné provést kompletní přepracování funkcionality aplikace XQuery search, jež poslouží jako základ nové aplikace I:ZI Repository. Součástí tohoto cíle je i úplné přepracování původní struktury aplikace za účelem umožnění snadnějších úprav, přidávání nové funkcionality a oprav implementačních chyb aplikace I:ZI Repository.

Pro usnadnění zavádění nových funkcí se musí také změnit způsob interního zacházení s daty, se kterými aplikace I:ZI Repository bude pracovat. Data musí být strukturovaná a vhodně reprezentovaná.

Kapitola 1

Teoretická část

Úkolem této kapitoly je uvést do tématu pojednávaného touto diplomovou prací (podkap. 1.1, 1.2 a 1.3). Dále poskytuje informace o systému LISp-Miner (podkap. 1.4), jehož výstupy jsou v práci používány. Dalším cílem této kapitoly je položit teoretické základy pro vylepšení aplikace (podkap. 1.5 a 1.6), o které tato práce pojednává.

1.1 Proces dobývání znalostí

Dobývání znalostí se začalo objevovat počátkem 90. let 20. století. První workshopy věnované dobývání znalostí se objevily na mezinárodní konferenci o umělé inteligenci IJCAI'89 a na konferencích americké asociace umělé inteligence AAAI'91 a AAA'93, jak uvádí [3]. Tato oblast se začala rozvíjet díky potřebě analyzovat narůstající objemy uchovávaných dat v různých organizacích. Dobývání znalostí propojilo tři oblasti: statistiku, databáze a strojové učení.

Dobývání znalostí z databází (KDD - Knowledge Discovery and Data mining) je možné definovat jako netriviální extrakci implicitních, dříve neznámých a potenciálně užitečných informací z dat [6]. Jak uvádí [3], KDD je chápáno jako interaktivní a iterativní proces tvořený kroky výběru, předzpracování, transformace, „dolování" a interpretace.

Oproti použití statistických metod se v dobývání znalostí klade důraz na předzpracování dat a také interpretaci výsledků.

Používané metodiky

Metodiky mají za cíl poskytnout uživatelům jednotný rámec pro řešení různých typů úloh z oblasti dobývání znalostí, jak je uvedeno v [3]. Některé metodiky byly vyvinuty společnostmi vyvíjejícími programové systémy pro dobývání znalostí (např. metodika „5A" společnosti SPSS, metodika SEMMA společnosti SAS), jiné vznikají díky spolupráci výzkumných a komerčních subjektů (např. CRISP-DM).

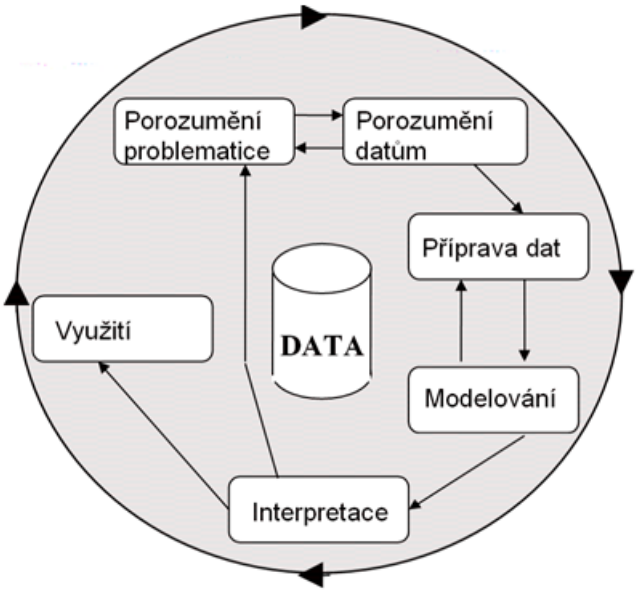
Metodiky v následujícím přehledu jsou seřazeny od nejpoužívanější k nejméně používané

podle hlasování na serveru kdnuggets.com z roku 2007[1].

Metodika CRISP-DM

Metodika CRISP-DM vznikla v rámci Evropského výzkumného projektu, jehož cílem bylo navrhnout univerzální postup, který bude použitelný v nejrůznějších komerčních aplikacích [3]. Název je zkratkou pro *CRoss-Industry Standard Process for Data Mining*.

Projekt se skládá ze šesti fází, jejich pořadí není pevně dáno. Výsledek jedné fáze ovlivní volbu následujících kroků, často je nutné se k některým krokům vracet. Grafické znázornění kroků je na obrázku 1.1.



Obrázek 1.1: Metodika CRISP-DM – Zdroj: `http://zt.vse.cz/sewebar/`

Fáze metodiky CRISP-DM[2]:

- **porozumění problematice** – fáze zaměřená na pochopení cílů projektu a požadavků na řešení formulovaných z manažerského hlediska,
- **porozumění datům** – obsahuje sběr dat a činnosti umožňující získat představu o datech,
- **příprava dat** – zahrnuje činnosti, které vedou k vytvoření datového souboru, který bude zpracováván jednotlivými analytickými metodami,
- **modelování** – využití analytických metod zahrnujících algoritmy pro dobývání znalostí,

[1] KDnuggets – Polls – Data Mining Methodology (Aug 2007), URL: `http://www.kdnuggets.com/polls/2007/data_mining_methodology.htm`

[2] Převzato ze stránek projektu Euromise, URL: `http://euromise.vse.cz/kdd/index.php?page=proceskdd`

- **vyhodnocení výsledků** – dosažené výsledky jsou vyhodnocovány z pohledu uživatelů (z pohledu splnění cílů definovaných na začátku projektu),
- **využití výsledků** – může obsahovat pouze sepsání závěrečné zprávy, stejně tak zavedení systému pro klasifikaci nových případů.

Metodika SEMMA

Z metodiky *SEMMA* vychází softwarový produkt *Enterprise miner* společnosti SAS. Metodika *SEMMA* je vlastní metodikou společnosti SAS, název je akronymem jednotlivých kroků (dle [3]):

- **Sample** – vybrání vhodných objektů,
- **Explore** – vizuální explorace a redukce dat,
- **Modify** – seskupení objektů a hodnot atributů, datové transformace,
- **Model** – analýza dat (neuronové sítě, rozhodovací stromy, statistické techniky, asociace a shlukování),
- **Assess** – porovnání modelů a interpretace.

Metodika klade důraz na interpretaci výsledků, která musí být pochopitelná pro obchodního uživatele.

Metodika „5A“

Metodika „5A“ pochází od společnosti SPSS. Název metodiky je akronymem pro jednotlivé kroky (uvedeno dle [3]):

1. **Assess** – posouzení potřeb projektu,
2. **Access** – shromáždění potřebných dat,
3. **Analyze** – provedení analýz,
4. **Akt** – přeměna znalostí na akční znalosti,
5. **Automate** – převedení výsledků analýzy do praxe.

Prvním krokem v analytickém procesu je stanovení kontextu, protože žádná data nemají význam, pokud jsou od kontextu oddělena. Druhým krokem je sběr a příprava dat, je nutné mít k dispozici vhodná data. Třetím krokem je nalezení odpovědí na otázky stanovené v prvním bodě, k čemuž je používáno různých analytických postupů. Ve čtvrtém kroku procesu se znalosti získané ve třetím bodu mění na znalosti akční, vznikají různá doporučení, dodatečné otázky a následná rozhodnutí. Pátý bod převádí výsledky analýzy do praxe.

1.2 Asociační pravidla

Asociační pravidla byla výrazně zpopularizována Rakeshem Agrawalem počátkem 90. let 20. století v souvislosti s analýzou nákupního košíku. IF-THEN konstrukce se nachází ve všech programovacích jazycích, nachází se i v běžné mluvě. Jak uvádí [3], jedná se o jeden z nejčastěji používaných prostředků pro reprezentaci znalostí.

Základní charakteristiky pravidel

Pro pravidlo *Předpoklad (antecedent)* $\Rightarrow$ *Závěr (consequent, succedent)* nás budou zajímat následující statistiky – kolik příkladů ve zkoumaných datech splňuje předpoklad a kolik závěr, kolik příkladů splňuje předpoklad i závěr současně atp. Tyto statistiky lze uspořádat do následující kontingenční tabulky (převzato z [3])

	Suc	**¬Suc**	$\sum$
Ant	a	b	r
¬Ant	c	d	s
$\sum$	k	l	n

Tabulka 1.1: Kontingenční tabulka charakteristik asociačního pravidla

kde
a ... počet objektů pokrytých současně předpokladem i závěrem,
b ... počet objektů pokrytých předpokladem a nepokrytých závěrem,
c ... počet objektů nepokrytých předpokladem ale pokrytých závěrem,
d ... počet objektů nepokrytých ani předpokladem ani závěrem.

Z údajů uvedených v tabulce 1.1 lze vypočítat základní charakteristiky asociačních pravidel podle Agrawalova pojetí - *podpora (support)* a *spolehlivost (confidence).*

Podpora udává počet objektů splňujících předpoklad i závěr. Absolutní podpora je a, relativní se vypočítá jako

$$P(Ant \wedge Suc) = \frac{a}{a+b+c+d}$$

.

Spolehlivost (někdy také platnost, konzistence či správnost) je dána vzorcem

$$P(Suc|Ant) = \frac{a}{a+b}$$

.

Mezi další důležité charakteristiky patří (dle [3]):

- absolutní resp. relativní počet objektů, které splňují předpoklad

  a + b resp. $P(Ant) = \frac{a+b}{a+b+c+d}$

- absolutní resp. relativní počet objektů, které splňují závěr

$$\text{a + c resp. } P(Suc) = \frac{a+c}{a+b+c+d}$$

- pokrytí, tj. podmíněná pravděpodobnost předpokladu pokud platí závěr

$$P(Ant|Suc) = \frac{a}{a+c}$$

- kvalita, tj, vážený součet spolehlivosti a pokrytí

$$Kvalita = w_1\frac{a}{a+b} + w_2\frac{a}{a+c}, \text{ kde } w_1 \text{ a } w_2 \text{ se obvykle volí tak, aby } w_1 + w_2 = 1.$$

1.3 Metoda GUHA

Asociační pravidla popsaná v podkapitole 1.2 v Agrawalově pojetí byla interpretována jako implikace – $Antecedent \Rightarrow Succedent$. Obecnější pohled na typy pravidel poskytuje původní česká metoda GUHA (General Unary Hypothesis Automaton – automat na obecné unární hypotézy), na které začal tým okolo Petra Hájka pracovat již v 60. letech 20. století (tedy již přibližně 30 let před Agrawalem). Metoda užívá vlastní terminologii založenou na predikátové logice, předpoklad označuje jako *antecedent*, závěr jako *sukcedent*. Pro typ pravidla zavádí termín *kvantifikátor*, který vychází z užití v matematice (obecný – $\forall$, existenční – $\exists$) [3].

„GUHA ze zadaných množin antecedentů a sukcedentů systematicky generuje všechny hypotézy a testuje, zda jsou nějak podporovány zpracovávanými daty. Řekneme-li, že data podporují nějakou hypotézu, znamená to zhruba, že v datech platí něco, co by bylo velmi nepravděpodobné, kdyby hypotéza neplatila.“[5]

Jádrem metody GUHA je spojení metod pro generování hypotéz s metodami pro jejich statistické testování. Na Vysoké škole ekonomické byla Janem Rauchem a Milanem Šimůnkem vyvinuta GUHA procedura *4ft-Miner* [17], která navazovala na původní GUHA metodu $ASSOC^3$ profesora Petra Hájka.

1.3.1 Procedura 4ft-Miner

Procedura *4ft-Miner* je implementací metody GUHA a je určena pro získávání asociačních pravidel z matic dat. Procedura pracuje s booleovskými atributy odvozenými ze sloupců analyzované matice dat, každý atribut – sloupec matice může nabývat konečného počtu hodnot, ty se nazývají *kategorie* [16].

Ze sloupců matice dat se vytvářejí booleovské atributy. *Základní booleovský atribut* má tvar $A(\alpha)$ – α je neprázdná podmnožina množiny hodnot atributu A. Booleovské atributy jsou odvozeny od základních booleovských atributů pomocí logických spojek $\wedge, \vee, \neg$.

[3] Nejvíce používaná GUHA procedura, hledá vztahy, které jsou zobecněním asociačních pravidel [16].

1.3.2 4ft-kvantifikátory

Asociační pravidlo pro procedura *4ft-Miner* definujme jako $\varphi \approx \psi$, kde φ a ψ jsou booleovské atributy, symbol $\approx$ reprezentuje 4ft-kvantifikátor. Asociační pravidlo se týká analyzované matice dat M. Každému 4ft-kvantifikátoru odpovídá určitá podmínka, která se týká hodnot čtyřpolní tabulky [16].

Mezi důležité 4ft-kvantifikátory patří (převzato z [16, 15]):

- 4ft-kvantifikátor **fundované implikace** $\Rightarrow_{p,Base}$, pro $0 < p \leq 1$ a $Base > 0$ definován podmínkou
$$\frac{a}{a+b} \geq p \wedge a \geq Base,$$
- 4ft-kvantifikátor **fundované dvojité implikace** $\Leftrightarrow_{p,Base}$, pro $0 < p \leq 1$ a $Base > 0$ definován podmínkou
$$\frac{a}{a+b+c} \geq p \wedge a \geq Base,$$
- 4ft-kvantifikátor **fundované ekvivalence** $\equiv_{p,Base}$, pro $0 < p \leq 1$ a $Base > 0$ definován podmínkou
$$\frac{a+d}{a+b+c+d} \geq p \wedge a \geq Base,$$
- 4ft-kvantifikátor **kladné odchylky od průměru** $\approx_{p,Base}$, pro $0 < p$ a $Base > 0$ definován podmínkou
$$\frac{a}{a+b} \geq (1+p)\frac{a+c}{a+b+c+d} \wedge a \geq Base.$$
Tento 4ft-kvantifikátor se také nazývá *AA-kvantifikátor*, z anglického *Above Average*.

1.4 LISp-Miner

Tato podkapitola se zaměřuje na popis systému LISp-Miner, jehož výstupy jsou zpracovávány aplikací I:ZI Repository. Informace o dataminingových úlohách a jejich výsledky jsou ukládány do databáze aplikace I:ZI Repository. Celá tato podkapitola vznikla na základě literatury [23].

1.4.1 Popis systému

Vývoj systému začal na přelomu let 1995 a 1996. Systém nově implementoval metodu GUHA, navazoval ovšem na dřívější teoretické práce a implementace této metody.

LISp-Miner je jednou z implementací metody GUHA, resp. implementací několika GUHA procedur. Navazuje na zkušenosti z předchozích implementací metody i na dlouhodobě budovanou teorii. Přináší také nové možnosti v zadávání úloh, prezentaci výsledků apod. Významné je i rozšíření o paralelní výpočet jedné úlohy na více počítačích zároveň, což je prováděno pomocí počítačového gridu[4].

[4]Počítačový grid – seskupení počítačů, které jsou připraveny spouštět výpočet na základě požadavků

1.4.2 Procedura 4ft-Miner

Procedura *4ft-Miner* hledá v datech vztahy v podobě *4ft-asociačních pravidel – Antecedent* ≈ *Sukcedent* – a podmíněná *4ft-asociační pravidla – Antecedent* ≈ *Sukcedent / Podmínka*. Cílem procedury je nalézt všechna asociační pravidla, která v analyzovaných datech platí.

Procedura *4ft-Miner* byla první analytickou procedurou zařazenou do systému LISp-Miner a zůstává zřejmě stále nejpoužívanější. Jedná se o novou a rozšířenou (oproti dřívějším implementacím) implementaci GUHA procedury ASSOC.

1.4.3 Asociační pravidla v systému LISp-Miner

Asociační pravidla jsou v systému LISp-Miner hypotézami generovanými procedurou *4ft-Miner*. Od klasických asociačních pravidel se mírně liší, proto jsou nazývány *4ft-asociační pravidla*. Rozdílnost spočívá především v bohatší syntaxi asociačních pravidel v proceduře *4ft-Miner*.

Syntaxe 4ft-asociačního pravidla

4ft-asociační pravidla obsahují cedenty. Termín cedent vyjadřuje fakt, že vlastnosti antecedentu, sukcedentu a podmínky jsou obdobné a lze je zobecnit do pojmu cedent. V souvislosti s implementací dalších procedur v systému LISp-Miner je nutné rozlišovat jednotlivé typy cedentů. Cedent je část hypotézy a skládá se z dílčích cedentů. U procedury *4ft-Miner* se v hypotéze objevují jako antecedent, sukcedent nebo podmínka.

Z pohledu této práce je důležitý *4ft-cedent*. Jedná se o odvozený booleovský atribut s bohatou syntaxí dílčích cedentů, literálů a koeficientů vycházející z koncepce cedentů v proceduře *4ft-Miner*.

Literál je kombinace atributu a koeficientu, může být doplněna o predikát negace. Z pohledu GUHA metody se jedná o odvozený booleovský atribut. Koeficient reprezentuje množinu kategorií.

Z pohledu nastavení procedury *4ft-Miner* a také informací o tomto nastavení, jež jsou součástí dat ukládaných v databázi aplikace I:ZI Repository, je důležité zadání literálu. To se skládá ze zadání typu literálu[5], typu gace[6] a zadání typu koeficientu.

Typ koeficientu

Nastavení typu koeficientu při nastavování *4ft-Miner* procedury definuje přípustné kombinace kategorií, které mohou být zároveň vloženy do koeficientu literálu. Jde o tyto možnosti:

od hlavního počítače [23]

[5]Type literálu definuje požadavek na přítomnost literálu v dílčím cedentu – možnosti *Basic* a *Remaining*, více [23]

[6]Typ gace ovlivňuje generování variant literálu s negací, více [23]

- **Subset** – libovolná kombinace kategorií,
- **Interval** – v koeficientu mohou být pouze sousední kategorie,
- **Cyclical Interval** – jako *Interval*, ale za sousední je považovaná i první a poslední kategorie,
- **Left Cut** – jako *Intreval*, ale vždy musí začínat první kategorií,
- **Right Cut** – jako *Intreval*, ale vždy musí končit poslední kategorií,
- **Cut** – sjednocení koeficientů generovaných jako *Left Cut* a *Right Cut*,
- **Boolean true** – přítomna může být pouze kategorie označená jako *Boolean Type True*,
- **Boolean false** – přítomna může být pouze kategorie označená jako *Boolean Type False*,
- **Both Boolean** – přítomna může být pouze kategorie označená jako *Boolean Type True* nebo *Boolean Type False*,
- **One Category** – koeficient může tvořit pouze jedna explicitně zvolená kategorie.

Atribut

Z pohledu systému LISp-Miner je atribut kategorizovaný pohled na sloupec databázové tabulky. Implementované GUHA procedury dokáží pracovat pouze s diskrétními (nespojitými) veličinami.

Booleovský atribut popisuje existenci či neexistenci určité vlastnosti u záznamů v datové matici. Pro každý řádek datové matice nabývá logické hodnoty 0 nebo 1 na základě toho, zda objekt danou vlastnost v tomto řádku má resp. nemá.

Jednoduchý booleovský atribut je tvořen z atributu výběrem jedné z jeho kategorií. Jedná se tedy o dvojici *atribut (kategorie)*, která pro každý řádek datové matice nabývá logické hodnoty 1 v případě, že objekt v daném řádku má ve sloupci databázové tabulky dané kategoriálním atributem hodnotu patřící do dané kategorie.
Příklad: Quality (good) ... nabývá hodnoty 1, pokud sloupec *Quality* obsahuje hodnotu *good*, v opačném případě je 0.

Odvozený booleovský atribut je odvozen od jednoduchého booleovského atributu pomocí logických operátorů *diskujnkce, konjunkce a negace.*

1.5 Logiky a množiny

Tato podkapitola byla zařazena jako teoretický základ pro pozdější návrhy zavedení fuzzy principů do vyhledávání v datamingových úlohách, které aplikace I:ZI Repository poskytuje.

1.5.1 Fuzzy a klasické množiny

Teorii fuzzy množin uvedl v roce 1965 Lofti A. Zadeh, definoval ji jako *„třídu objektů s kontinuálním ohodnocením členství. Taková množina je charakterizována funkcí členství, která přiřazuje každému objektu hodnotu jeho členství v rozmezí 0 a 1.“*[22]

Anglický výraz „fuzzy“ lze přeložit jako neostrý, rozmazaný nebo také chlupatý. Především výrazy *neostrý* a *rozmazaný* dobře vyjadřují základní myšlenku fuzzy množin.

Klasické množiny (crisp množiny) jsou jasně definovány buď výčtem prvků, určením vlastností nebo pomocí charakteristické funkce

$$\varphi_A : U \to \{0, 1\}$$

takové, že

$$\forall x \in U; \varphi_A(x) \begin{cases} 0, & \text{právě když } x \notin A \\ 1, & \text{právě když } x \in A. \end{cases}$$

V reálném světě ale nelze vždy přesně určit, zda daný prvek patří do dané množiny či nikoliv. Například lze snadno určit, že automobil patří do skupiny motorových dopravních prostředků. Do této skupiny ale zajisté nepatří kolečkové brusle, protože nemají motor. Skupinu motorových dopravních prostředků lze považovat za skupinu klasickou (crisp), protože jde jasně určit, zda prvek do skupiny patří či ne.

Pokud ale vezmeme v úvahu sílu motoru automobilu, nelze jasně určit, zda vozidlo patří do skupiny motorově silných či slabých. Neexistuje žádná obecně uznávaná hranice, která by obě skupiny jasně rozdělovala.

I kdyby taková hranice existovala, takové rozdělení vozidel by bylo velice ostré. Představme si takovou hranici určenou například výkonem motoru 150 kW. Automobil, který má 149 kW by ještě patřil do skupiny slabých, ovšem automobil, který má 150 kW by již patřil do skupiny silných.

V této situaci by očividně bylo vhodné zavést množinu, která bude zohledňovat míru příslušnosti prvku do dané množiny. Pokud bychom brali v úvahu interval výkonu motoru od 50 kW do 300 kW a tento rozsah rozdělili do dvou skupin, tyto skupiny se mohou především kolem poloviny intervalu překrývat. To znamená, že vozidlo s výkonem 170 kW (pokud budeme brát jako polovinu daného intervalu hodnotu 175 kW) spadá z větší části do skupiny vozidel slabých a z menší části do skupiny vozidel silných. Toto rozdělení je pro člověka daleko přirozenější, než jasně vymezené a ostré hranice.

Fuzzy množiny zavádějí do úvah vágnost a neurčitost. Umožňují vyjádřit, že prvek do dané množiny patří jen do určité míry. Tato míra je nejčastěji v intervalu $[0; 1]$, přičemž 0 znamená, že prvek do množiny nepatří, 1 znamená opak.

Fuzzy množiny jsou definovány pomocí tzv. funkce příslušnosti (dle [4])

$$\mu_A : U \to [0, 1]$$

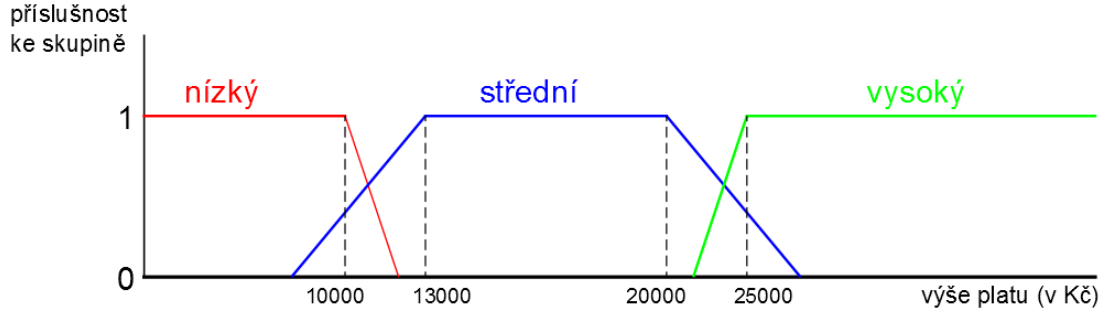
tak, že každému prvku $x \in U$ je přiřazena hodnota z intervalu $[0, 1]$.

Neostrost hranic lze předvést na následujícím příkladu. Mějme tři hladiny platu – nízká, střední a vysoká. Interval hodnot pro každý stupeň je uveden v tabulce 1.2.

Hladina platu	**Interval (v Kč)**
Nízká	$[0; 10000)$
Střední	$(13000; 20000]$
Vysoká	$(25000; 50000]$

Tabulka 1.2: Příklad fuzzy množiny – tabulka rozdělení hodnot

Grafické zobrazení jednotlivých hladin je na obrázku 1.2. Zde je také vidět, že se jednotlivé skupiny mohou i částečně překrývat.



Obrázek 1.2: Příklad fuzzy množiny – graf rozdělení hodnot

Pro fuzzy množiny jsou také definovány následující množinové operace (uvedené dle [4]):

- kardinalita fuzzy množiny A (počet prvků)

$$|A| = \sum_{x \in U} \mu_A(x)$$

- doplněk

$$\mu_{U \setminus A}(x) = 1 - \mu_A(x)$$

- průnik

$$\mu_{A \cap B}(x) = min(\mu_A(x), \mu_B(x))$$

- sjednocení

$$\mu_{A \cup B}(x) = max(\mu_A(x), \mu_B(x))$$

- A fuzzy podmnožina B

$$A \subseteq B \text{ právě když } \forall x; \mu_A(x) \leq \mu_B(x)$$

1.5.2 Klasická logika

Literatura [11] uvádí, že logika je studium metod a principů usuzování ve všech možných formách. Klasická logika pracuje s tvrzeními, která jsou buď pravdivá nebo nepravdivá. Každé tvrzení má svůj opak, který se běžně nazývá *negace* tvrzení.

1.5.3 Vícehodnotová logika

Základní předpoklad, na kterém je klasická (nebo dvouhodnotová) logika postavena, byl odedávna zpochybňován, jak uvádí [11]. Problém vytvářejí především předpoklady o budoucích událostech, které aktuálně nejsou ani pravdivé ani nepravdivé, ale potenciálně obojí. Jejich pravdivostní hodnota je tudíž neurčitelná.

Proto může být klasická logika rozšířena na 3-hodnotovou a to několika různými způsoby. Literatura [4] například uvádí rozšíření 2-hodnotové logiky na 3-hodnotovou pomocí hodnoty X, která má význam $UNKNOWN$ (tvrzení může být pravda nebo nepravda).

1.5.4 Fuzzy logika

Dle [20] je fuzzy logika ve smyslu nástroje přibližného usuzování zobecněním logiky vícehodnotové. Oborem pravdivostních hodnot této logiky je interval [0;1], 1 pro absolutní pravdu, 0 pro absolutní nepravdu. Důležitou roli hraje přirozené uspořádání reálných čísel, protože pravdivostní hodnoty lze díky tomu lineárně uspořádat.

„Důležitou vlastností fuzzy logik, stejně jako logik klasických či vícehodnotových (a na rozdíl od logik modálních, či logiky intuicionistické) je jejich extenzionalita.“[20, str. 109] Tím je myšleno, že lze určit pravdivostní hodnotu například formule $\varphi \vee \psi$ na základě pravdivostních hodnot jednotlivých formulí φ a ψ pomocí nějaké pravdivostní funkce $V^* : [0,1]^2 \to [0,1]$.

1.5.5 Užití fuzzy

Fuzzy vyhledávání

Fuzzy vyhledávání (angl. fuzzy search) je také někdy nazýváno přibližným porovnáváním textových řetězců (angl. approximate string matching).

Zjednodušeně lze fuzzy vyhledávání charakterizovat jako vyhledávací techniku, která dokáže vyhledat výsledky odpovídající vyhledávanému výrazu i v případě, že se v prohledávaném textu nebo vyhledávaném výrazu vyskytují chyby, jako například překlepy.[13, 21]

Důvody pro přibližné porovnávání

Na důvody použití přibližného porovnávání textových řetězců existují dle [7] dva pohledy: oprava chyb a získávání informací.

Můžeme předpokládat, že vyhledávaný řetězec či výraz se přesně shoduje s uloženými daty, záznamy. Během procesu vyhledávání může ovšem dojít k porušení vyhledávaného řetězce, popřípadě je poškozen uložený záznam. Lze hovořit o menších či větších porušeních, řetězec dokonce může být porušen takovým způsobem, že je stejný jako jiný řetězec, například jako jiné slovo. Pokud je tedy poškození větší než rozdíl mezi správnými řetězci, musíme

uvažovat navrácení chybných výsledků vyhledávání. Lze uvažovat nad opravou chyb způsobených poškozením procesem snažícím se při získávání výsledků chybu opravit a v případě získání výsledku, který není relevantní, jej označit za chybu. Zde se jedná o pohled opravující chyby.

Z pohledu získávání informací můžeme narazit na dva problémy – výsledkem vyhledávání budou nechtěné záznamy a chtěné výsledky budou opomenuty. Tyto úkazy jsou často nazývány

- *přesnost* – poměr získaných relevantních záznamů,
- *správnost* – poměr opravdu získaných relevantních záznamů.

Fuzzy asociační pravidla

Klasická asociační pravidla používají pro vyjádření hodnot jednotlivých cedentů klasické množiny (crisp množiny). Fuzzy asociační pravidla ovšem pro tato vyjádření používají fuzzy množiny, jak uvádí [9].

Vzhledem k tomu, že teorie fuzzy množin kvantifikuje a usuzuje za pomoci přirozeného jazyka, je vhodné použití fuzzy množin pro kvantitativní hodnoty.

Literatura [8] například uvádí situaci, kdy je při minování uvažována skupina lidí ve středním věku, tato skupina je určena intervalem od 30 do 50 let. Osoba, jejíž věk je 29 let, je tak 0% reprezentantem takové skupiny, zatímco osoba stará 31 let 100% reprezentuje tuto skupinu. V reálném světě ovšem tato hranice není tak ostrá a právě zde je vhodné využít fuzzy množin.

Stejný problém může také nastat při práci s kategoriálními daty, protože někdy nelze prvek bezvýhradně přidělit do skupiny. Klasické (crisp) množiny dovolují přidělit prvek pouze do jedné skupiny, zatímco fuzzy množiny nabízejí stupně příslušnosti do skupiny.

Literatura [8] také uvádí tři možné přístupy k fuzzy asociačním pravidlům:

Kvantitativní přístup Převedení kvantitativního atributu na atribut kategoriální za využití teorie fuzzy množin. Tento kategoriální atribut respektuje teorii fuzzy množin, proto například daná položka v databázi, která je určena kvantitativně, může částečně patřit do více kategorií. Pro daný atribut je třeba určit dané fuzzy množiny a jejich funkce příslušnosti, což by mělo být prováděno expertem.

Fuzzy taxonomické struktury Taxonomií je myšlena kategorizace položek definovaná uživatelem. Jedná se o hierarchii reprezentovanou stromovou strukturou, kde potomek náleží právě jednomu rodiči. V některých případech je ovšem vyžadován přístup, kdy potomek může patřit s určitou mírou příslušnosti více rodičům.
V klasických taxonomických strukturách náleží dítě k rodiči s mírou 1. Fuzzy taxonomie umožňuje potomkovi patřit více rodičovským uzlům na základě míry příslušnosti, takže každý potomek patří rodiči s mírou μ, kde $0 \leq \mu \leq 1$.

Přístup přibližných itemsetů Úkol vyhledávání frekventovaných itemsetů pracuje s prozkoumáváním častých částí v sekvenci případů. Data mohou být nahlížena jako sekvence případů spojených určitým časovým výskytem. Častou částí se rozumí sada případů, které se spolu často vyskytují. Základním úkolem je v prvé řadě takové případy nalézt. Pokud pracujeme s daty, ve kterých jsou položky často přeskočeny či ztraceny, problémy nastanou při prozkoumávání. Pro práci s takovými situacemi lze užít fuzzy časté itemsety, což znamená, že taková sada položek nemusí být jasně odhalena, ale pouze přibližně. Tradiční přístup by mohl z výsledků vyřadit zajímavé případy pouze proto, že nedosahují minimální, uživatelem specifikované, hodnoty podpory. Úkolem proto je do výsledků zařadit i případy, které se vyskytují méně často, ale i přesto mohou být zajímavé.

1.6 Shlukování

Úkolem této podkapitoly je položit teoretický základ pro možnosti zavedení shlukování výsledků vyhledávání do aplikace I:ZI Repository. Podkapitola ukazuje možné způsoby shlukování (podkap. 1.6.2 a 1.6.3) a také informace o post-processingu asociačních pravidel (podkap. 1.6.4), jehož může být shlukování součástí (podkap. 1.6.5).

1.6.1 Definice shlukování

Shlukování lze definovat jako učení bez učitele, které se zaměřuje na rozklad dané sady objektů na podskupiny nebo shluky na základě podobnosti. Cílem je rozložit dataset takovým způsobem, že objekty patřící do stejného shluku si jsou co nejpodobnější a objekty patřící do různých shluků si jsou co nejvíce nepodobné.[14]

1.6.2 Shluková analýza

„Shluková analýza zahrnuje řadu metod a postupů, které slouží k vytváření co nejsourodějších skupin objektů nazývající se shluky. V těchto shlucích by si objekty (vícerozměrná pozorování charakterizovaná řadou vlastností) měly být co nejvíce podobné z hlediska vnitroshlukové struktury a co nejméně podobné z hlediska mezishlukové struktury“ uvádí Tomáš Löster ve své disertační práci[12, str. 11].

Dle Berky shluková analýza *„hledá odpověď na otázku, zda lze pozorované příklady rozdělit do skupin (shluků) vzájemně si blízkých příkladů. Vychází se tedy z toho, že umíme měřit vzdálenost mezi příklady.“*[3, str. 55]

Shluková analýza je statistická metoda, jež je nástrojem odkrývajícím doposud skryté struktury v sadě neuspořádaných objektů. Zaměřuje se na vyhledávání podobností objektů a jejich následné seskupování do tříd, které jsou charakterizovány podobností obsažených objektů.

Základní způsoby měření podobnosti shluků

Pro roztřídění objektů do skupin či tříd je třeba nějakým způsobem určit, jak si jsou nebo nejsou podobné. Tuto podobnost lze vyjádřit vzdáleností jednotlivých objektů, k čemuž může sloužit mnoho různých metod, které vzdálenost určují. Zde jsou uvedeny pouze základní z nich.

Uvedeno dle [12]:

- **Vzdálenost založená na centroidech**
  Používá se, pokud jsou objekty určeny kvantitativními proměnnými. Vyjádřeno pro h-tý a h'-tý shluk:
  $$D_C(C_h, C_{h'}) = D(\overline{x}_h, \overline{x}_{h'}),$$
  kde D(...) je libovolná míra vzdálenosti[7] a $\overline{x}_h, \overline{x}_{h'}$ jsou centroidy h-tého a h'-tého shluku. Centroid $\overline{x}_h$ může být reprezentován m-rozměrným vektorem průměrných hodnot jednotlivých kvantitativních proměnných, počítaných na základě objektů zařazených do daného shluku.

- **Vzdálenost nejbližšího souseda**
  V tomto případě se určují vzdálenosti pro všechny dvojice objektů, z nichž jeden patří do h-tého shluku a druhý do h'-tého shluku. Vzdálenost se určí jako minimum ze zjištěných vzdáleností objektů:
  $$D_{BS}(C_h, C_{h'}) = \min_{x_i \in C_h, x_j \in C_{h'}} D(x_i, x_j).$$
  Tento postup lze použít pro libovolné proměnné, za předpokladu použití vhodné míry odlišnosti.

- **Vzdálenost nejvzdálenějšího souseda**
  Podobně jako v předchozím případě se i zde určují vzdálenosti všech dvojic objektů, z nichž jeden je v h-tém shluku a druhý v h'-tém shluku. Vzdálenost shluků se zde ovšem stanovuje jako maximum z určených vzdáleností objektů:
  $$D_{VS}(C_h, C_{h'}) = \max_{x_i \in C_h, x_j \in C_{h'}} D(x_i, x_j).$$

- **Vzdálenost průměrného souseda**
  Aritmetický průměr vzdáleností pro všechny dvojice objektů, z nichž jeden patří do h-tého shluku a druhý do h'-ho shluku:
  $$D_{PS}(C_h, C_{h'}) = \frac{1}{n_h n_{h'}} \sum_{x_i \in C_h} \sum_{x_j \in C_{h'}} D(x_i, x_j).$$

1.6.3 Shlukovací metody

Na shluky může být nahlíženo jako na podmnožiny zkoumaných dat (datasetů). Shlukovací metody mohou být děleny několika způsoby, jedním z nich je rozdělení dle toho, zda podmnožiny zkoumaných dat jsou *crisp* (klasické množiny) nebo *fuzzy*. Podle toho lze metody

[7] Pro stanovení míry vzdálenosti existuje mnoho funkcí a metod, blíže například v [12], kapitola 1.1.

rozdělit na *hard clustering* (tvrdé shlukovací) metody či *fuzzy clustering* (fuzzy shlukovací) metody, jak to nabízí například [2].

Tvrdé shlukovací metody. Tyto metody jsou založeny na teorii klasických množin, takže objekt musí do podmnožiny (shluku) buď úplně patřit nebo úplně nepatřit. Tvrdé shlukování rozděluje data na specifický počet vzájemně výlučných podmnožin (shluků).

Fuzzy shlukovací metody. Tyto metody umožňují objektu patřit zároveň do více shluků s různými stupni příslušnosti. Fuzzy shlukování je v mnoha situacích přirozenější než tvrdé shlukování. Objekty ležící na hranici více shluků nemusí nutně patřit pouze do jedné podmnožiny, ale mohou mít stanoven stupeň příslušnosti a patřit tak do shluků či shluku pouze částečně.

Tvrdé a fuzzy dělení

Cílem shlukování je rozdělit dataset Z na c shluků (tříd, skupin). Předpokládejme, že c je předem známo. Následující je uváděno dle [2].

Tvrdé dělení datasetu Z lze definovat za pomoci klasických množin jako skupinu podmnožin $\{A_i | 1 \leq i \leq c\} \subset \mathcal{P}(Z)$[8] s následujícími vlastnostmi:

$$\begin{aligned} &\bigcup A_i = Z, \\ &A_i \cap A_j = \emptyset, 1 \leq i \neq j \leq c, \\ &\emptyset \subset A_i \subset Z, 1 \leq i \leq c. \end{aligned}$$

První řádek určuje, že sjednocení podmnožin A_i musí obsahovat všechna data. Podle druhého řádku musí být podmnožiny disjunktní a žádná nesmí být prázdná či obsahovat všechna data, což je určeno ve třetím řádku.

Rozdělení může být vhodně reprezentováno maticí rozdělení $U = [\mu_{ik}]_{c \times N}$, i-tý řádek této matice obsahuje hodnoty funkce příslušnosti μ_i pro i-tou podmnožinu A_i ze Z. Z vlastností pro tvrdé dělení vycházejí následující vlastnosti pro U:

$$\begin{aligned} &\mu_{ik} \in \{0, 1\}, 1 \leq i, \leq c, 1 \leq k \leq N, \\ &\sum_{i=1}^{c} \mu_{ik} = 1, 1 \leq k \leq N, \\ &0 < \sum_{k=1}^{N} \mu_{ik} < N, 1 \leq i \leq c. \end{aligned}$$

Prostor všech možných matic tvrdého dělení pro Z je tudíž definován jako

$$M_{hc} = \left\{ U \in \mathbb{R}^{c \times N} \middle| \, \mu_{ik} \in \{0, 1\}, \forall i, k; \sum_{i=1}^{c} \mu_{ik} = 1, \forall k; 0 < \sum_{k=1}^{N} \mu_{ik} < N, \forall i \right\}.$$

[8] $\mathcal{P}(Z)$ je potenční množinou Z – množinou, která obsahuje všechny podmnožiny Z

Fuzzy dělení. Zobecněním tvrdého dělení na fuzzy dělení je dosaženo umožněním $\mu_i k$ obsahovat hodnoty z reálné řady čísel v rozmezí $[0, 1]$. Podmínky pro matici rozdělení, analogické k podmínkám pro matici tvrdého rozdělení:

$$\mu_{ik} \in \{0, 1\}, 1 \leq i, \leq c, 1 \leq k \leq N,$$
$$\sum_{i=1}^{c} \mu_{ik} = 1, 1 \leq k \leq N,$$
$$0 < \sum_{k=1}^{N} \mu_{ik} < N, 1 \leq i \leq c.$$

i-tý řádek matice fuzzy rozdělení U obsahuje hodnoty i-té funkce příslušnosti pro fuzzy podmnožinu A_i ze Z. Druhý řádek podmínek omezuje součet všech sloupců na 1, tudíž celková hodnota příslušnosti pro každé z_k v Z je jedna. Prostor fuzzy rozdělení pro Z je

$$M_{fc} = \left\{ U \in \mathbb{R}^{c \times N} \middle| \, \mu_{ik} \in \{0, 1\}, \forall i, k; \sum_{i=1}^{c} \mu_{ik} = 1, \forall k; 0 < \sum_{k=1}^{N} \mu_{ik} < N, \forall i \right\}.$$

1.6.4 Post-processing asociačních pravidel

Literatura [10] uvádí rozdělení metod pro zpracování množiny asociačních pravidel podle toho, zda metoda využívá doménové znalosti či nikoliv.

Pokud máme k dispozici doménové znalosti ve strukturované formě, je velice vhodné je využít pro zpracování množiny vygenerovaných asociačních pravidel. Tímto způsobem lze asociační pravidla například rozdělit do následujících skupin:

- asociační pravidla popisující znalosti v souladu s částí doménových znalostí,
- asociační pravidla, která jsou s dosavadními doménovými znalostmi v rozporu,
- ostatní asociační pravidla.

Asociační pravidla se používají také pro hledání důležitých závislostí a zajímavých asociací v datech. Používají se ovšem také pro průzkum dat, kde je často dostupnost apriorních znalostí minimální, jak uvádí [10]. V uvedených případech apriorní doménové znalosti často nejsou ve strukturované, počítačem zpracovatelné podobě.

Literatura [10] se proto věnuje zpracování asociačních pravidel bez užití apriorních doménových znalostí a uvádí mimo jiné některé příklady z literatury, jakým způsobem k této problematice přistupovat.

Jedním z příkladů je rozdělení pravidel uživatelem do čtyř skupin: pravdivé-nezajímavé, nepravdivé-nezajímavé, nepravdivé-zajímavé a pravdivé-zajímavé. Toto rozdělení navrhuje článek [18].

„Cílem použití metod (algoritmů) hledání asociačních pravidel je získání znalostí o existujících závislostech v datech, znalostí dalších vlastností analyzovaných dat“[10, str. 42].

Důležitou vlastností a rysem dobývání znalostí z databází je reálná a praktická použitelnost a nasaditelnost. Metody asociačních pravidel jsou velmi často používány v případech, kdy jsou výsledky zpracovávány člověkem spíše než softwarovým nástrojem. To je motivací ke zpracovávání množin asociačních pravidel, která jsou výsledkem dataminingu. Předat člověku množinu získaných pravidel ke zpracování je proto vhodné až po zpracování například desetitisíců získaných pravidel.

Principy metod zpracování množiny asociačních pravidel (dle [10]):

- odstranění asociačních pravidel nevyhovujících dodatečně určeným podmínkám (filtrování),
- odstranění nadbytečných či nezajímavých asociačních pravidel,
- vizualizace asociačních pravidel,
- seskupování asociačních pravidel,
- uspořádání asociačních pravidel.

Odstraňování asociačních pravidel

Literatura [10] uvádí následující metody odstraňování asociačních pravidel. V další literatuře se ovšem objevují také další metody, které zde nejsou uvedeny.

Dodatečné filtrování – templates Umožňuje uživateli provést výběr asociačních pravidel, která by mohla být považována za zajímavá. Uživatel může uvést, že například požaduje asociační pravidla, která obsahují jen zadaný parametr v antecedentu. Uživatel také může udat požadovaný počet atributů v jednotlivých cedentech či například minimální a maximální hodnoty pro dané atributy (kvantitativní) asociačního pravidla.
Podobný princip filtrování se také nachází v implementaci 4ft-Mineru, ovšem již ve fázi určení zadání pro generování asociačních pravidel. Principy *templates* lze tudíž využít jak v post-processingu tak i při generování asociačních pravidel.

Dedukční pravidla Nadbytečná pravidla může uživatel snadno („pouhým okem“) odvodit z jiných asociačních pravidel. Logické vyplývání asociačních pravidel se řídí dedukčními pravidly. Redukce takových nadbytečných pravidel je velice užitečná a uživateli poskytuje možnost zpřehlednit získaná asociační pravidla.

1.6.5 Seskupování asociačních pravidel

Součástí post-processingu je také seskupování či shlukování asociačních pravidel.

Získaná skupina asociačních pravidel může obsahovat pravidla, která jsou různá a nesouvisející, může ovšem obsahovat i pravidla, která spolu nějakým způsobem souvisí, jsou si podobná. Taková pravidla je vhodné seskupit do podmnožin, díky čemuž může být i velký počet nalezených pravidel přehledný a pro člověka snáze zpracovatelný.

Specifickou metodou post-processingu asociačních pravidel je procedura SD4ft-Miner. Tato procedura hledá zajímavé dvojice asociačních pravidel, které se týkají dvou podmnožin záznamů z datové matice. Cílem procedury je nalézt všechny takové dvojice asociačních pravidel, které v analyzovaných datech platí. Proceduru lze využít pro hledání zajímavých odlišností mezi dvěma skupinami záznamů. [23]

Článek [1] uvádí další algoritmy pro seskupování pravidel: Objective Grouping (OG) a Subjective grouping (SG). OG algoritmus seskupuje pravidle podle jejich syntaktické struktury bez využití doménových znalostí. Druhý, SG algoritmus naopak využívá doménové znalosti a pravidla seskupuje podle sémantických informací o objektech v pravidle. Oba tyto algoritmy seskupují podobná pravidla k sobě a uživateli poskytují vysokoúrovňový přehled nad pravidly. Oba algoritmy vyžadují od uživatele částečnou spolupráci a aktivitu v průběhu seskupování pravidel. Seskupování pomocí těchto dvou algoritmů může být použito na asociační pravidla odvozená z databáze obsahující velké množství různých položek a atributů.

Kapitola 2

Zadání úprav aplikace

Tato kapitola obsahuje informace o zadání úprav a vylepšení existující aplikace XQuery Search. Zpracováním těchto úprav a vylepšení se diplomová práce zabývá. V rámci této práce bude aplikace také přejmenována na I:ZI Repository. Tento název lépe vystihuje účel aplikace.

Podkapitola 2.1 obsahuje popis zadání úprav a nových funkcí aplikace, podkapitola 2.2 popisuje prostředky, které budou při zavádění změn a nových funkcí do aplikace použity.

2.1 Popis zadání úprav a vylepšení aplikace

Zadání lze rozdělit do dvou hlavních částí – úpravy stávajících funkcí aplikace a přidání nových funkcí.

Hlavní částí úprav aplikace je změna způsobu reprezentace dat při průchodu aplikací. Doposud aplikace zachází s daty jako s nestrukturovaným textovým řetězcem. Především z pohledu nově navrhovaných funkcionalit aplikace je nutné data transformovat na strukturovaná. Příchozí dotaz i výsledky vyhledávání z úložiště budou do aplikace stále přicházet v nestrukturované podobě, aplikace by ovšem před začátkem zpracovávání dat měla tato převést na strukturovaná. Z pohledu použitých technologií se nabízí transformace dat (jež jsou v XML formě) na objekty.

Do oblasti nových funkcí spadá hned několik změn – zavedení možnosti vyhledávání výsledků běhu dataminingových úloh podle jejich zadání, zavedení fuzzy vyhledávání pro výsledky běhu dataminingových úloh (případně i pro zadání úloh) a shlukování pro nalezené výsledky běhu dataminingových úloh.

Vyhledávání pomocí zadání dataminingových úloh by mělo umožnit získat výsledky běhu úloh podle jejich zadání. Z tohoto pohledu je nutné navrhnout úpravy současně používaného vyhledávacího dotazu, pomocí kterého aplikace komunikuje s ostatními službami, případně navržení nového vyhledávacího dotazu. Zavedení tohoto druhu vyhledávání by s sebou také mělo nést možnost vyhledávat podle obecnějších atributů, které jsou právě součástí zadání dataminingových úloh.

Fuzzy vyhledávání by mělo reflektovat podobnost vyhledávacího dotazu a nalezeného výsledku z úložiště. Součástí fuzzy ohodnocení výsledku vyhledávání by také mělo být zohlednění dalších aspektů nalezeného výsledku, nejen podobnosti s dotazem.

Shlukování by mělo umožnit rozdělit výsledky nalezené pomocí vyhledávacího dotazu do skupin (shluků) podle zvolené vlastnosti výsledků, tato volba může být případně na uživateli.

Výše popsané změny s sebou ponesou nutnost změn již používaných prostředků, především vyhledávacího dotazu a také výstupních dat aplikace.

2.2 Prostředky k dosažení cílů

Postup zavádění úprav a nových funkcí se bude skládat ze čtyř částí – analýzy, návrhu, implementace a testování. Tento postup by měl zaručit prozkoumání potřeb pro zavedení úprav a nových funkcí, nalezení různých možností řešení, zvolení správných technologických prostředků a otestování správnosti provedených úprav aplikace.

Z pohledu technologií jsou některá omezení dána již existující aplikací XQuery search – použití XML databáze Berkeley a platformy Java EE (enterprise edition). Pro nasazení aplikace je použit aplikační server Glassfish. Z tohoto pohledu je nutné používat kompatibilní technologie. Pro usnadnění vývoje je vhodné použít pomocné knihovny a případně užít i aplikační framework.

Kapitola 3

Analýza

Tato kapitola je zaměřena na analýzu možných směrů řešení zadání. Kapitola se zabývá možnými způsoby hodnocení podobnosti nalezených asociačních pravidel a zadání vyhledávání a možností navržení fuzzy vyhledávání z komplexního pohledu. Dále je navržen způsob shlukování nalezených výsledků vyhledávání.

Součástí kapitoly je také popis výchozí situace před vypracováním diplomové práce, především výchozí stav aplikace XQuery search.

3.1 Definice používaných pojmů

Některé používané pojmy jsou vysvětleny v podkapitole 1.4.

Dataminingová úloha Úloha dolování informací z dat, skládá se ze zadání a výsledků.

Zadání dataminingové úlohy Z pohledu práce se jedná o strukturu dat, která obsahuje informace o zvoleném nastavení dataminingové úlohy, jejím zadání pro dolování informací z databází. Obecně se jedná o zvolená kritéria pro dolování informací z databází – například jaké atributy v jaké části asociačního pravidla nás zajímají, jaký výpočet použít pro hodnocení pravidla atp.

Výsledky dataminingové úlohy Asociační pravidla získaná během dataminingové úlohy. Informace vytěžené z dat reprezentované asociačními pravidly. Tato asociační pravidla jsou spolu se zadáním dataminingové úlohy ukládána v úložišti.

Vyhledávací dotaz Struktura dat popisující zadání vyhledávání pro aplikaci I:ZI Repository (stejně tak pro původní aplikaci XQuery search). Sestává se z podobných prvků jako asociační pravidlo – definuje jaký atribut s jakou hodnotou a ve které části asociačního pravidla vyhledávat.

Výsledek vyhledávání Asociační pravidlo uložené v úložišti spolu se zadáním dataminingové úlohy, které bylo nalezeno při vyhledávání podle kritérií ve vyhledávacím dotaze. Jeden výsledek vyhledávání je jedno asociační pravidlo.

Míra podobnosti Tato hodnota popisuje míru podobnosti vyhledávacího dotazu a nalezeného výsledku vyhledávání. Vzorem pro hodnocení je vyhledávací dotaz, hodnoceným objektem je výsledek vyhledávání – asociační pravidlo. Pro vyhledávání v zadání dataminingových úloh je opět vzorem pro hodnocení vyhledávací dotaz, hodnoceným objektem je zadání nalezené dataminingové úlohy.

3.2 Fuzzy přístupy ve vyhledávání

3.2.1 Výsledky běhu dataminingové úlohy

Pro vyhledávání ve výsledcích datamingových úloh připadá v úvahu několik možných způsobů zadání vyhledávacího dotazu a zaměření na data vznikající v různých částech běhu dataminingové úlohy (její zadání a výsledky):

- vyhledávání podle zadaných atributů a parametrů – výsledky dataminingových úloh,
- vyhledávání podle vzorového asociačního pravidla – výsledky dataminingových úloh,
- získávání výsledků běhů dataminingových úloh podle jejich zadání.

Rozšíření vyhledávání ve výsledcích dataminingových úloh o fuzzy přístupy by přineslo možnost poskytovat uživateli výsledky, které by pro něj mohly být zajímavé, i když nevyhovují plně zadání ve vyhledávacím dotazu. Zde se bude jednat o přímé vyhledávání v asociačních pravidlech získaných z běhů dataminingových úloh, ať při vyhledávání podle vybraných atributů, tak i například při vyhledávání podle vzorového asociačního pravidla.

Pokud bychom chtěli uživateli poskytnout dostatečně obsáhlé a vyčerpávající výsledky na zadaný vyhledávací dotaz, nabízí se ve výsledných datech vrátit i informace o výsledcích dataminingových úloh podobných zadání ve vyhledávacím dotazu. Pro uživatele může být užitečné podívat se na výsledky dataminingových úloh, které mají například jiné hodnoty v atributech asociačního pravidla, atributy navíc či některé chybějící atributy a další možné odchylky od vyhledávacího dotazu.

Zde se nabízí problém, jakým způsobem vypočítat odchylku od vyhledávacího dotazu. Tato odchylka je vhodná zejména pro řazení a filtrování vrácených výsledků vyhledávání, může také uživateli pomoci s výběrem dataminingových úloh a jejich výsledků, na které by se mohl více zaměřit a prozkoumat je detailněji.

Pro výsledky běhů dataminingových úloh (asociační pravidla) je důležité se zaměřit na rozdílnost jednotlivých atributů v uloženém asociačním pravidle obsažených a jejich strukturu, spojení, případné negace atributů. Mimo atributů a jejich struktury je třeba se také soustředit na charakteristiky asociačního pravidla, i když se z pohledu vyhledávání jedná o méně důležité informace o pravidle.

Hodnocení pomocí řídících pravidel

Pro výpočet odchylky mezi vyhledávacím dotazem a zkoumaným asociačním pravidlem z databáze lze například zvolit určitá řídící pravidla, která by podle nastavených kritérií snižovala míru vyhovění vyhledávacímu dotazu. Příkladem může být řídící pravidlo *„Pokud chybí v antecedentu asociačního pravidla jeden atribut (oproti vyhledávacímu dotazu), sniž míru vyhovění aktuálně zkoumané dataminingové úlohy o 5 %“*.

Takovou konstrukci lze nazývat řídícím pravidlem. Řídící pravidlo by se skládalo z podmínky a snížení ohodnocení zkoumaného výsledku vyhledávání. Řídící pravidlo by bylo používáno výhradně pro ohodnocení výsledku vyhledávání, pojem není nijak příbuzný s asociačním pravidlem – výsledkem běhu dataminingové úlohy.

Hodnocení jako vektor

Další možností je vyjádření vyhovění zkoumaných výsledků vyhledávání (asociačních pravidel) vektorem. Zde by se ovšem jednalo zejména o zkoumání použitých atributů. Pokud bychom brali jako výchozí stav vyhledávací dotaz, jednotlivé složky ve výsledném vektoru by reprezentovaly přítomnost či nepřítomnost atributu z vyhledávacího dotazu ve zkoumaném výsledku vyhledávání.

Přítomnost by například byla reprezentována jako 1, nepřítomnost jako 0. V tuto chvíli se ovšem nejedná o fuzzy přístup. Problémem je také fakt, že je vhodné uvést i informace o tom, že ve zkoumaném výsledku běhu dataminingové úlohy jsou oproti vyhledávacímu dotazu atributy navíc. Vybrané hodnocení není příliš vhodné ani z následujícího pohledu – atributy mohou mít například zvolenu jen jinou hodnotu (pokud je zadání úlohy konkretizováno zvolením hodnoty pro některý z atributů), což je slabší rozdílnost, než kdyby atribut nebyl vůbec uveden.

Kombinace předchozích

Možným řešením by bylo vylepšení vektoru výše uvedenými řídícími pravidly. To by znamenalo určovat podobnost nalezeného výsledku běhu dataminingové úlohy a vyhledávacího dotazu nikoliv pomocí hodnot 0 a 1, ale zavést do určování podobnosti fuzzy hodnoty v intervalu od 0 (atribut se vůbec nevyskytuje) do 1 (atribut je úplně stejný). Míra podobnosti jednotlivých atributů (ve vyhledávacím dotazu a ve výsledku vyhledávání) by se tedy řídila zvolenými řídícími pravidly, která by míru podobnosti podle určitých odlišností snižovala.

Zde je opět důležité vhodné určení hodnoty, o kolik by dané řídící pravidlo snižovalo míru podobnosti nalezeného asociačního pravidla a vyhledávacího dotazu.

Stále ještě nebyl vyřešen problém s atributy, které ve vyhledávacím dotazu nejsou obsaženy a které by také měly mít vliv na konečné hodnocení podobnosti vyhledávacího dotazu a výsledku vyhledávání. Zde by měl být brán zřetel na počet atributů, které jsou ve výsledku vyhledávání navíc a také jejich konkrétnost. Tento problém by bylo možné vyřešit sadou řídících pravidel, která by určovala, o kolik se míra podobnosti vyhledávacího dotazu a výsledku vyhledávání sníží a za jakých podmínek.

Výsledné hodnocení nalezeného výsledku

Ve výše uvedených podkapitolách jsou zkoumány možnosti ohodnocení podobnosti vyhledávacího dotazu a výsledku vyhledávání. Součástí výsledného fuzzy ohodnocení výsledku vyhledávání by ovšem neměla být pouze míra podobnosti s vyhledávacím dotazem, ale také další vlastnosti výsledku vyhledávání.

Součástí uloženého výsledku běhu dataminingové úlohy mohou být i další informace, které nebyly použity pro hodnocení podobnosti vyhledávacího dotazu a výsledku vyhledávání, měly by ale být také součástí výsledného fuzzy ohodnocení výsledku vyhledávání.

Takovou informací může být uživatelská anotace zajímavosti asociačního pravidla. Taková anotace nemusí být přítomna, pokud přítomna je, pravidlo je označeno anotací jako zajímavé či nezajímavé. Především nezajímavost pravidla by se měla do celkového fuzzy ohodnocení asociačního pravidla promítnout.

Výsledným hodnocením pravidla by tak mohl být vektor, který by obsahoval hodnocení podobnosti vyhledávacího dotazu a výsledku vyhledávání. Stejně tak by vektor fuzzy ohodnocení obsahoval hodnocení dalších aspektů výsledku vyhledávání, které nesouvisí s podobností s vyhledávacím dotazem, jsou nicméně také důležité.

3.2.2 Zadání dataminingové úlohy

Jedním z možných typů dotazů, které by mohly uživatele zajímat, je dotaz *„Existuje již v úložišti stejná nebo podobná dataminingová úloha? Jaké byly její výsledky?“*. Takovým dotazem uživatel požaduje zjistit, jestli v minulosti již běžela dataminingová úloha se stejným či podobným zadáním, které si uživatel aktuálně navolil (např. v nějakém GUI nástroji) a jakých výsledků bylo dosaženo.

Pro takový druh dotazu je potřeba mít v úložišti mimo výsledků běhu dataminingových úloh také informaci o zadání úlohy. Pokud takovou informaci úložiště obsahuje, je nutné vybrat vhodný způsob, jakým datamingovou úlohu v úložišti porovnat s vyhledávacím dotazem.

Pokud se jedná o vyhledání dataminingové úlohy plně shodné s vyhledávacím dotazem, řešení je vcelku jednoduché – stačí porovnat jednotlivá nastavení dataminingové úlohy s kritérii ve vyhledávacím dotazu, použité atributy a další charakteristiky a pokud se přesně shodují, přidat takovou dataminingovou úlohu do množiny výsledků vyhledávání. V tuto chvíli uživatele zřejmě nemusí tolik zajímat zadání dataminingové úlohy (které musí být stejné jako vyhledávací dotaz), spíše pro něj budou přínosné výsledky běhu takové dataminingové úlohy.

V případě, že chceme uživateli mimo výsledků dataminingové úlohy přesně odpovídající vyhledávacímu dotazu poskytnout i výsledky dataminingových úloh podobných vyhledávacímu dotazu, můžeme použít mírně upravené řešení pro vyhledávání výsledků běhů dataminingových úloh.

Výrazným omezením pro vyhledávání podle zadání dataminingových úloh jsou dostupné údaje. Oproti vyhledávání ve výsledcích běhů dataminingových úloh jsou možnosti pro

vyhledávání v zadání úloh omezené. Je to dáno strukturou vyhledávacího dotazu, dostupnými informacemi o zadání dataminingových úloh a způsobem vyhledávání. Z tohoto důvodu je k dispozici pro fuzzy vyhledávání v zadáních dataminingových úloh pouze možnost poměřování počtu atributů ve vyhledávacím dotazu a v uložených informacích o zadání dataminingové úlohy.

Možným východiskem nedostatku dat pro toto fuzzy vyhledávání by mohlo být zobecnění vyhledávacího dotazu. Tak by aplikace před dotázáním do úložiště vyhledávací dotaz upravila, aby snížila konkrétnost jeho kritérií. Díky tomu by bylo k dispozici větší množství dat k porovnávání a mohlo by také být využito více aspektů ukládaných informací o zadání a nastavení dataminingové úlohy.

3.3 Shlukování ve vyhledávání

Využití shlukování pro potřeby této práce a aplikace I:ZI Repository lze rozdělit na dvě samostatnější části – shlukování a sdružování.

Zatímco pro sdružování se předpokládají přesně vymezené hranice rozdělení výsledků vyhledávání zvolené uživatelem (například rozdělení podle kategorií zvoleného atributu), pro shlukování se předpokládá výpočet hranic až na základě získaných výsledků, ze strany uživatele se předpokládá omezení shlukování uvedené například jako specifikace počtu shluků apod.

V souvislosti se sdružováním i shlukováním je používán pojem prvek skupiny/shluku – tento prvek je výsledek vyhledávání neboli asociační pravidlo. Skupinou se rozumí seskupení prvků se stejnými vlastnostmi pro sdružování. Shlukem se rozumí seskupení prvků se stejnými vlastnostmi pro shlukování.

3.3.1 Shlukování

Jako shlukování je zde myšlena analýza dat (v našem případě výsledků vyhledávání) a jejich rozdělení do shluků podle zvolené charakteristiky. Pro odlišení od druhé části (sdružování) bude tato část zaměřena především na vytváření shluků podle zvoleného cílového počtu shluků, případně dalších charakteristik shlukování, které si uživatel zvolí. Hlavním znakem této činnosti bude rozřazování výsledků vyhledávání do jednotlivých shluků na základě výpočtu prováděného z charakteristik pravidel získaných z úložiště při vyhledávání.

Charakteristikou výsledku vyhledávání je myšlena informace, která je přímo součástí výsledku vyhledávání, nebo ji lze z informací, které jsou součástí výsledku vyhledávání, získat například výpočtem. Příkladem takové charakteristiky může být fuzzy ohodnocení výsledku vyhledávání.

Využití centroidů

Příkladem shlukování může být rozřazování výsledků vyhledávání do jednotlivých shluků na základě tzv. centroidů (viz podkapitola 1.6.2). Centroidem se v této části rozumí ur-

čitý bod, který je charakterizován prvky spadajícími do shluku, pro nějž je daný centroid vypočítáván. Centroid takový shluk, dá se říci, reprezentuje a charakterizuje.

Centroidem může být bod určený výpočty z charakteristik prvků, které obsahuje shluk tímto centroidem reprezentovaný. V tomto případě by se hodnota takového bodu mohla změnit, pokud do shluku přibude nebo ze shluku ubude nějaký prvek.
Centroidem by mohl být i jeden z výsledků vyhledávání (jeden pro každý shluk), který by charakterizoval daný shluk. I zde by mohlo dojít ke změně centroidu, v případě změn charakteristik shluku (přidávání a ubírání výsledků), kdy se může stát, že se tyto charakteristiky shluku změní natolik, že jako reprezentativní by byl vhodný jiný výsledek vyhledávání.

Pro přiřazování výsledků vyhledávání do shluků se bude používat míra podobnosti shluku (reprezentovaného centroidem) a výsledku vyhledávání. Ta by například pro případ, kdy je výsledek vyhledávání pro shlukování charakterizován svým fuzzy ohodnocením – vektorem, byla mírou podobnosti dvou vektorů. Reprezentantem takové míry podobnosti může být pro představu kosínová míra podobnosti dvou vektorů (viz podkapitola 5.4.2).

Nastavení hranice příslušnosti do shluku

Přiřazování do shluků může uživatel ovlivnit zvolením určitého prahu, jehož překročením by výsledek spadal již do jiného shluku nebo by se dokonce mohl stát novým centroidem a shluk vytvořit. Při využití míry podobnosti shluku a výsledku vyhledávání by takový práh určil minimální podobnost shluku a výsledku vyhledávání, pro kterou ještě může výsledek vyhledávání patřit do shluku[1].

Nastavení výsledného počtu shluků

Uživateli by také mohla být dána možnost ovlivnit či přímo nastavit výsledný počet shluků. Pokud bychom chtěli dosáhnout přesného splnění uživatelského zadání, nemělo by být pro tuto možnost k dispozici zvolení prahu příslušnosti či nepříslušnosti do shluku. Tato hranice by totiž měla být vypočítávána až v průběhu shlukování.

Nastavení počtu prvků ve shluku

Určitou variantou ovlivnění vzniku shluků by mohlo být i určení maximálního počtu prvků pro jeden shluk. Otázkou je ovšem možnost využití takové varianty pro uživatele ve smyslu užitečnosti získaných dat. Tato možnost by také narážela na problém dělitelnosti počtu nalezených prvků počtem prvků ve shluku definovaným uživatelem.

[1] Mějme stupnici od 0 do 1 – 0 říká, že výsledek vyhledávání a shluk si nejsou vůbec podobné, 1 říká, že výsledek vyhledávání a shluk jsou totožné. Příkladem takového prahu pak může být hodnota 0,5. Výsledky vyhledávání, které mají míru podobnosti s daným shlukem nižší či rovnou prahu do shluku patřit nemohou, výsledky vyhledávání mající míru podobnosti vyšší, než je zvolený práh, do shluku patřit mohou.

Pevná omezení vytváření shluků

Nezbytným faktorem bude také nastavení omezení pro vytváření jednotlivých shluků, které by mělo být součástí samotné aplikace a uživatelem neovlivnitelné. Mělo by se jednat o nastavení, která budou zaručovat správnou funkčnost shlukování. Možné omezení demonstruje příklad: Jeden prvek musí patřit právě do jednoho shluku.

Pokud bylo zmíněno omezení, že prvek by měl patřit právě do jednoho shluku, lze spolu se zavedením fuzzy přístupu pravidlo upravit: prvek musí patřit alespoň do jednoho shluku. V rámci fuzzy přístupu by totiž bylo možné prvku přiřadit míru příslušnosti do shluku. Nesporně by zde došlo k ztížení především v části implementace. Zřejmě by zde mohlo dojít i k snížení přehlednosti pro uživatele a otázkou je i využitelnost této vlastnosti.

3.3.2 Sdružování

Sdružováním je pro potřeby této práce myšleno soustředění výsledků vyhledávání do určitých celků podle zvoleného parametru či způsobu rozlišení jednotlivých celků.

Sdružování by mělo uživateli pomoci především lépe se orientovat ve výsledcích vyhledávání. Uživatel by měl mít na výběr způsob, jakým sdružené celky budou tvořeny, případně co bude kritériem pro rozřazování do celků.

Jako možné příklady sdružování lze uvést:

- **Sdružení podle zadání úlohy**
  V případě, že vyhledáváme ve výsledcích běhů dataminingových úloh má řazení podle zadání úloh pro uživatele jasný přínos v přehlednosti. Přínosem je i možnost podle zadání úlohy prozkoumat její výsledky například pouze u úloh, které mu svým zadáním přijdou zajímavé.
  Tento přístup je aplikovatelný i na vyhledávání v zadání dataminingových úloh, především pokud uživatel požaduje jako součást výsledků vyhledávání i výsledky dataminingové úlohy.
- **Sdružení podle hodnot vybraného atributu**
  Tento způsob sdružování je vhodný pro vyhledávání ve výsledcích běhů úloh. Uživatel by měl možnost zvolit si atribut a na základě hodnot daného atributu obsaženého v nalezeném pravidle by došlo k rozřazení nalezených pravidel do jednotlivých celků. Takový způsob by byl využitelný i pro spojité atributy, kdy by si uživatel mohl například specifikovat, po jak velkých intervalech daného atributu mají být výsledky sdruženy.
- **Sdružení podle potvrzení/vyvrácení zadaného pravidla**
  Takovéto sdružení výsledků je vhodné pro dotazy typu *„Najdi v úložišti pro asociační pravidlo specifikované ve vyhledávacím dotazu taková asociační pravidla, která toto asociační pravidlo potvrzují nebo vyvracejí.“*
  Uvažujme asociační pravidla – výsledky dataminingové úlohy – ve tvaru *předpoklad* → *závěr*, asociační pravidla v úložišti potvrzující asociační pravidlo z vyhledávacího dotazu budou mít stejný závěr a stejný či podobný[2] předpoklad.

[2]Podobný předpoklad můžeme pro účely práce definovat jako podmnožinu atributů zadaného předpokladu, které mají neprázdný průnik hodnot s atributy v předpokladu.

Stejná situace bude i pro asociační pravidla vyvracející asociační pravidlo z vyhledávacího dotazu – za stejného či podobného předpokladu vychází rozdílný (či dokonce opačný[3]) závěr.
Nalezená pravidla by poté byla sdružena do dvou skupin – potvrzující a vyvracející asociační pravidlo z vyhledávacího dotazu.

- **Sdružování podle délky asociačního pravidla**
  Lze předpokládat i situaci, kdy uživatel bude požadovat sdružit nalezená asociační pravidla podle jejich délky[4]. Zde se nabízejí dva možné způsoby zvolení sdružování.
  První možností je sdružování podle celkové délky asociačního pravidla, druhou sdružování podle délky jednotlivých částí asociačního pravidla zvlášť (podle délky předpokladu či závěru).
  Druhá možnost se dále může rozpadnout na volbu výběru pouze jedné části (například sledování délky pouze pro předpoklad) či obou částí při zachování jejich odlišení (tento způsob by mohl ovšem vést i k nepřehlednosti výsledků).
  Délka asociačního pravidla může vypovídat o obecnosti či konkrétnosti pravidla, nejedná se ovšem o přesný a jasně vymezující ukazatel. Je to spíše pomůcka pro uživatele, na základě které se může rozhodnout.

- **Sdružování podle tvaru částí asociačního pravidla**
  Pro uživatele by mohlo být zajímavé i rozdělení nalezených asociačních pravidel podle jejich závěru. Tento způsob sdružování by dával smysl ve chvíli, kdy má uživatel k dispozici možnost vyhledávat v uložených asociačních pravidlech pomocí vymezení tvaru jednotlivých částí (předpoklad, závěr). Jedná se vlastně o obecnější způsob sdružování, které by nabízelo sdružování podle potvrzení/vyvrácení asociačního pravidla z vyhledávacího dotazu.
  Pokud uživatel vymezí ve vyhledávacím dotazu tvar jedné z částí asociačního pravidla a bude hledat přesnou shodu v uložených asociačních pravidlech, nabízí se možnost sdružení podle druhé (v dotazu neurčené) části asociačního pravidla jako vhodná a užitečná.

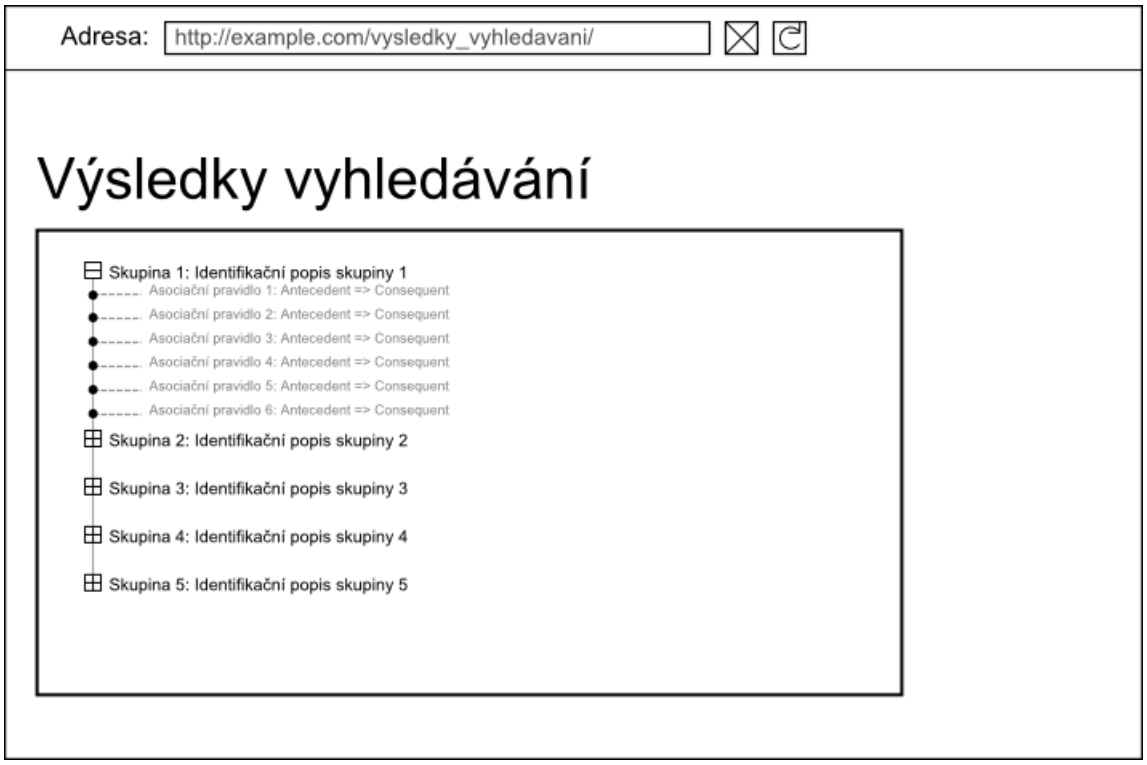
3.3.3 Příklad možné vizualizace skupin

Možným způsobem vizualizace sdružených skupin by mohl být určitý druh stromu, kde by na 1. úrovni byly jednotlivé skupiny (ty by byly nějakým způsobem jasně a unikátně definovány pro jejich jednoduché rozlišení, součástí základního přehledu by mohly být i informace popisující danou skupinu) a po rozevření skupiny by se zobrazily výsledky vyhledávání, které do dané skupiny spadají.

Součástí této práce ovšem není grafický návrh aplikace pro vyhledávání, tento návrh možného zobrazování výsledků vyhledávání rozdělených do skupin a práce s nimi je zde uveden především pro představu. Příklad vizualizace výsledků vyhledávání ve skupinách je na obrázku 3.1.

[3]Opačný závěr můžeme definovat na příkladu: k atributu *Bydliště(Praha)* je opakem atribut *not Bydliště(Praha)*. Opakem by také mohl být pro binární atribut (mající pouze dvě možné hodnoty) následující příklad: pro atribut *Student(ano)* je opakem atribut *Student(ne)*.

[4]Délkou asociačního pravidla je myšlen počet atributů v něm obsažených.



Obrázek 3.1: Příklad možné vizualizace skupin

3.4 Analýza čestnosti

V rámci poskytování informací o datech držených v úložišti by pro uživatele mohla mít přínos i analýza četnosti. Jedná se především o doplňující informace, které by mohly být součástí výsledků vyhledávání v úložišti či některé přímo přístupné jako samostatně vypovídající.

Možné způsoby využití a poskytování informací o četnosti:

- **Četnost pravidel potvrzujících či vyvracejících dotaz**
  V tomto případě by se mohlo jednat o informaci, která by byla využitelná především pro dotazy na potvrzení a vyvrácení pravidla v dotazu. Jednalo by se o informaci pouze s doplňujícím a shrnujícím charakterem.
  Četnost by mohla být vyjádřena v procentech i absolutních číslech. Vypovídající hodnotou by bylo, zda a jak moc je dané pravidlo na základě dat z úložiště potvrzeno či vyvráceno.
- **Četnost vybraného atributu pro dotazem charakterizovaná pravidla**
  Příkladem může být pravidlo charakterizované svým závěrem, který uživatel zadá jako dotaz. Zároveň by si uživatel vybral atribut, jehož četnost chce zkoumat.
  Výsledkem takového dotazu by mohla být statistika pro jednotlivé hodnoty vybraného atributu pro pravidla odpovídající vybrané charakteristice (v rámci dotazu).
  Jednodušší variantou by byl případ, kdy uživatel vybere atribut i s jeho hodnotou a je vrácena četnost výskytu vybrané kombinace v úložišti.
- **Četnost vybraného atributu pro celé úložiště**

Jedná se o modifikaci a zobecnění předchozího bodu. Uživatel by si vybral atribut, pro který chce vrátit četnost a ta by mu byla v rámci výsledků poskytnuta.

3.5 Výchozí stav pro vypracování diplomové práce

3.5.1 Aplikace vytvořená v bakalářské práci

Aplikace XQuery search, z níž vychází nová aplikace I:ZI Repository, vznikla v roce 2010 jako hlavní výstup z mé bakalářské práce nazvané „Využití XML databází pro zpřístupnění specifikací úloh dobývání znalostí z databází“. Návrh zahrnoval využití XQuery dotazovacího jazyka pro získávání dat z XML databáze. Pro práci s XML databází bylo zvoleno Java API, které komunikovalo s okolím pomocí HTTP protokolu, jednalo se o tzv. RESTful aplikaci.

Aplikace umožňovala uložení XQuery dotazu, který byl následně doplněn příchozím dotazem v podobě XML a kompletní dotaz vzniklý tímto spojením byl odeslán do databáze pro získání výsledků. Výsledky byly strukturovány a upraveny do výchozího formátu taktéž pomocí XQuery. Java API poskytovalo pouze přístupovou bránu k XML databázi, celé dotazování včetně zpracování vstupu i výstupu bylo prováděno pomocí XQuery.

Velkým problémem tohoto modelu byla výkonnost. Dotazy nad velkým počtem dat trvaly i desítky sekund. Výhodou oproti jiným řešením (např. dotazování do relační databáze za pomoci Lucene indexu) byla variabilita dotazu a možnost vyjádření struktury požadovaného asociačního pravidla. Tyto vlastnosti vycházely ze skutečnosti, že dotaz i uložená asociační pravidla měly formu XML dokumentu.

3.5.2 Vývoj v rámci projektu SEWEBAR

Od začátku vývoje aplikace v rámci bakalářské práce bylo v úmyslu její využití v systému vznikajícím v rámci projektu SEWEBAR[5]. Díky REST architektuře nebylo propojení s dalšími aplikacemi nijak problematické. Aplikace XQuery search byla pomocí KBI modulu napojena na CMS Joomla!, který je využíván ve výuce k publikování výsledků dataminingových úloh a také k ukázkám možností využití celého systému.

Jako první byla možnost ukládat PMML dokumenty vložené do CMS zároveň do XML databáze. Dále byla využita možnost dotazování se nad uloženými daty – pro každé asociační pravidlo, které bylo součástí reportu uloženého v CMS, byla možnost z XML databáze zjistit, zda existují asociační pravidla podobná či pravidla mající ze stejného předpokladu jiné východisko. Taková pravidla se uživateli zobrazila a poskytovala možnost se pomocí odkazu přesunout přímo do reportu, ze kterého asociační pravidlo pocházelo.

Vzhledem k původním problémům s výkonností vyhledávání došlo k úpravám, které vedly k výraznému zlepšení situace. Celý proces samotného vyhledání dat z XML databáze byl před úpravami obsluhován XQuery skriptem, což se ukázalo jako velice neefektivní. Při takovém modelu docházelo pouze k okrajovému využití možností XML databáze, prakticky

[5]Internetové stránky projektu – `http://sewebar.vse.cz/`

nemohlo dojít k využití indexu dat. To bylo způsobeno zejména využitím XQuery pro celý proces vyhledávání – XQuery skript vybral všechna uložená asociační pravidla, která následně podroboval analýze, zda vyhovují zadanému dotazu.

Problém s výkonností byl proto vyřešen lepším využitím technologie XPath a snížením využití XQuery. Přijatý dotaz je zpracován v rámci Java API a z něj je vytvořen XPath dotaz, který je zasazen do XQuery skriptu, který provede vyhledání podle XPath a výsledky vyhledání upraví ve smyslu struktury a přidání některých informací. Při tomto modelu lze velice dobře využít optimalizací, které nabízí XML databáze, především pak indexu dat. Díky tomu se provedlo zlepšit čas potřebný na vyhledání na stovky milisekund až několik málo sekund (v závislosti na složitosti dotazu a rozsahu zkoumaných dat).

Spolu s rozšířením využití XPath došlo také k zavedení transformace ukládaných dat. Pomocí XSL transformace jsou přijatá data upravena do podoby vhodné pro vyhledávání – stromová struktura. Příchozí data by totiž musela být rekurzivně prohledávána, což je výpočetně náročnější než průchod stromovou strukturou (k čemuž jsou jazyky XPath a XQuery a také XML databáze navrženy).

3.5.3 Využití aplikací I:ZI Miner

Prozatím největší využití aplikace zaznamenala od napojení aplikace I:ZI Miner[6], který je taktéž vyvíjen v rámci projektu SEWEBAR Radkem Škrabalem.

I:ZI Miner poskytuje webové rozhraní pro zadávání dataminingových úloh a získávání jejich výsledků. Samotný běh dataminingové úlohy probíhá v systému LISp-Miner (viz kap. 1.4).

Do aplikace XQuery search se ukládají výsledky dataminingu, které I:ZI Miner obdrží od LISp-Mineru. Ukládána jsou asociační pravidla po jednom a to pouze v případě, že je uživatel označí jako zajímavá nebo naopak nezajímavá. I:ZI Miner na základě výsledků vyhledávání v databázi uložených a takto označených asociačních pravidel, které jsou aplikací poskytovány, provádí označování nezajímavých asociačních pravidel. Za nezajímavá asociační pravidla jsou považována taková, z nichž vyplývá již známá znalost. To by mělo uživateli I:ZI Mineru pomoci soustředit se na potencionálně zajímavá asociační pravidla.

[6]Demonstrační verze aplikace I:ZI Miner – `http://sewebar.lmcloud.vse.cz/izi-miner/`

Kapitola 4

Návrh

Obsahem této kapitoly je návrh aplikace I:ZI Repository. Podkapitola 4.1 se zaměřuje na návrh nové struktury aplikace, podkapitola 4.2 obsahuje informace o datech, která aplikace používá. Podkapitoly 4.3, 4.4, 4.5 a 4.6 jsou zaměřeny na návrh nové funkcionality, případně na návrh nového provozování již existující funkcionality. Podkapitola 4.7 obsahuje informace o podpůrných modulech aplikace a podkapitola 4.8 je zaměřena na popis technologických oblastí, které budou použity a změny aplikace umožní.

Účelem návrhu aplikace je transformovat výstupy získané v analýze na konkrétnější přístupy k následné implementaci. Účelem není získat konkrétní implementační postup či návrh úzce spojený s implementační platformou. Výstupem je návrh aplikace z hlediska struktury, vybraných řešení navržených v analýze případně možné řešení již známých problémů. Je třeba identifikovat procesy aplikace i s nimi spojená data.

4.1 Struktura aplikace

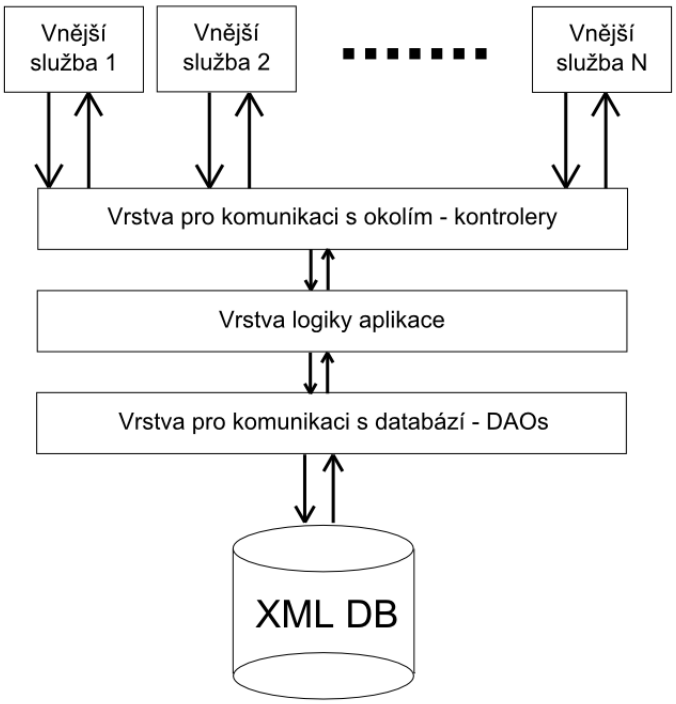
Aplikace bude rozdělena do tří hlavních částí. Rozdělení je navrženo především pro přehlednost kódu aplikace, sdružení částí aplikace se stejným zaměřením do celků, snadnější úpravy. Toto rozdělení také umožňuje snadnější orientaci dalších osob, které se s kódem aplikace případně budou seznamovat. Výhodou je také snadnější možnost výměny některých částí aplikace, například databáze.

Aplikace bude členěna na

- vrstvu pro komunikaci s okolím,
- vrstvu obsahující logiku aplikace
- a vrstvu zajišťující komunikaci s databází.

Obrázek 4.1 zobrazuje vrstvy aplikace, komunikaci mezi nimi a komunikaci s vnějšími službami[1] a úložištěm.

[1] Vnější službou je myšlena aplikace, která není součástí popisované aplikace. Tato vnější služba komunikuje s aplikací I:ZI Repository a využívá jejích služeb.



Obrázek 4.1: Vrstvy aplikace, komunikace mezi nimi, vnějšími službami a úložištěm

Na obrázku 4.2 je zobrazen objektový model aplikace. Jedná se o model objektů, které zajišťují hlavní funkcionalitu. Model tudíž neobsahuje doménové objekty a pomocné a další třídy, které jsou zobrazeny na dalších nákresech.

Pro správné pochopení obrázku 4.2 je třeba doplnit rozdíl mezi službou a modulem. Služba poskytuje jednoduchý přístup k dané funkcionalitě, dalo by se říci, že zastřešuje a usnadňuje použití funkcí, které poskytuje příslušný modul. Služba tak většinou dává k dispozici jednu metodu, přijímající hlavní objekty, které jsou příslušným modulem zpracovávány. Služba je také zodpovědná za zajištění doplňujících dat, která může modul potřebovat. Modul tak již obsahuje více metod, které poskytují konkrétní funkcionalitu. Služba tuto funkcionalitu využívá a konsoliduje. Služba je tak zjednodušujícím přístupovým rozhraním k funkcím modulu.

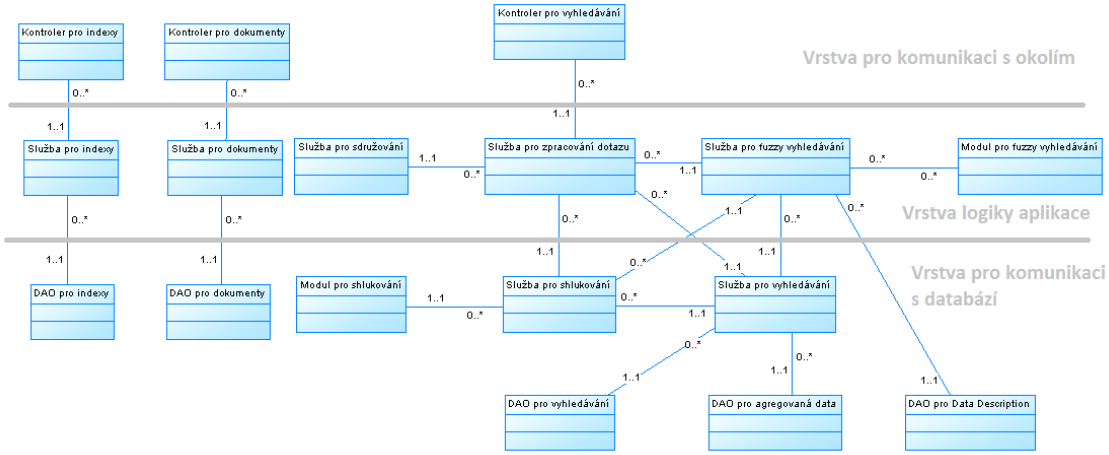
Služba pro sdružování nemá k sobě příslušný modul. Služba neposkytuje žádnou implementačně náročnou a rozsáhlou funkcionalitu, kterou by bylo třeba vyčlenit do modulu. K velké části své činnosti také využívá pomocnou utility třídu – viz bližší popis v podkapitole 4.6.

4.1.1 Doménové objekty

Doménové objekty[2] v aplikaci budou reprezentovat objekty, které budou v rámci aplikace zpracovávány, předávány apod. Příkladem doménového objektu může být dokument či vyhledávací dotaz.

Každý doménový objekt bude mít své vlastnosti vyjádřené pomocí proměnných. Doménový

[2]Doménový objekt v aplikaci reprezentuje objekt reálného světa.



Obrázek 4.2: Objektový model aplikace I:ZI Repository

objekt dokument by tak mohl mít své ID, název a tělo (to, co obsahuje).

Doménovými objekty budou v aplikaci I:ZI Repository reprezentovány

- dokumenty,
- dotazy do databáze – vyhledávací dotazy,
- výsledky dotazů z databáze – výsledky vyhledávání.

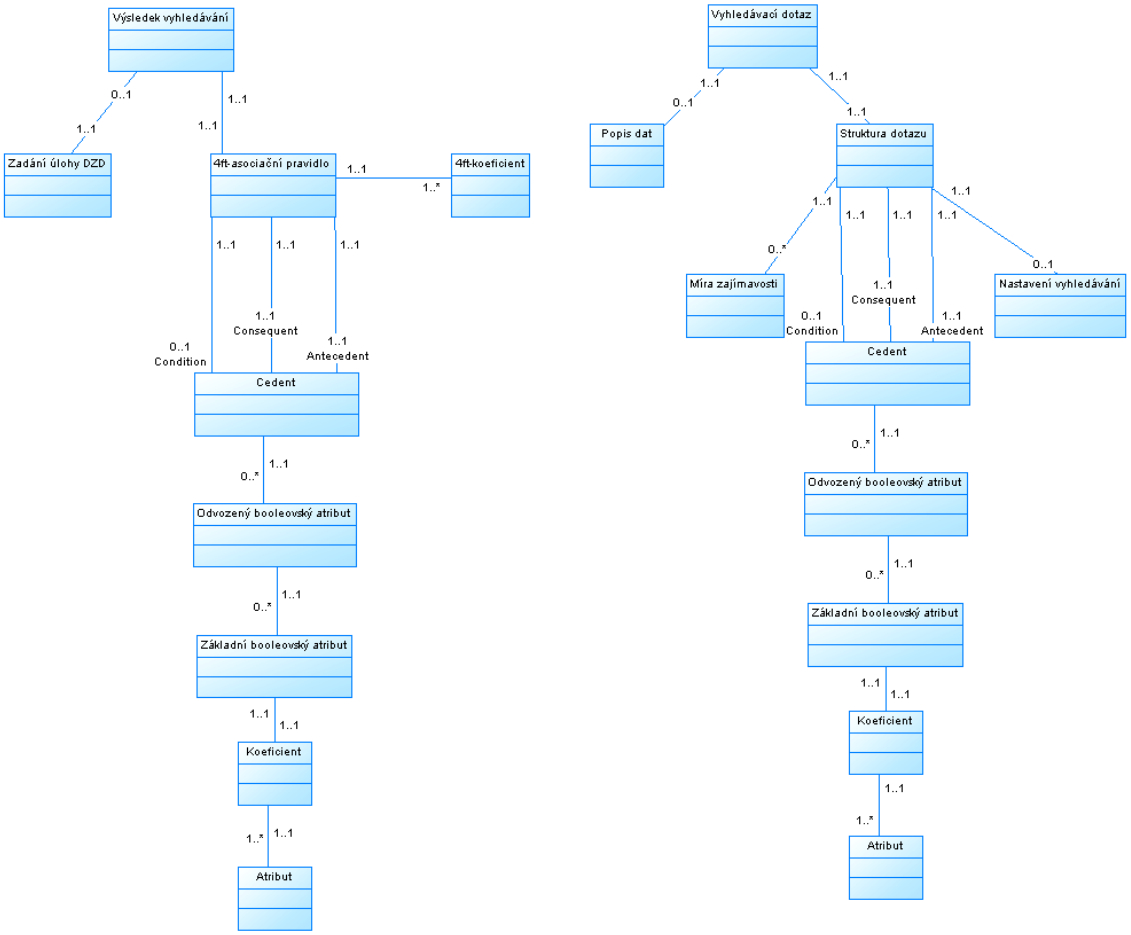
Jednotlivé výsledky vyhledávání v databázi se budou skládat z menších částí:

- zadání úlohy,
- výsledků běhu úlohy,
- v případě potřeby také popisu dat.

Návrh struktury doménových objektů pro výsledky vyhledávání a vyhledávací dotazy zobrazuje obrázek 4.3. Doménový objekt dokumentu není zobrazen – vzhledem k tomu, že není při průchodu aplikací nijak významně zpracováván, není ani reprezentován složitou strukturou, nicméně jedním objektem.

Pro oba doménové objekty (výsledku vyhledávání i vyhledávacího dotazu) se může objekt *odvozený booleovský atribut* skládat z dalších těchto objektů nebo z objektů *základního booleovského atributu*. Vysvětlení pojmů *základní (jednoduchý) booleovský atribut* a *odvozený booleovský atribut* lze nalézt v podkapitole 1.4.3.

Diagram tříd konkrétního provedení obou popisovaných objektů (vyhledávací dotaz, výsledek vyhledávání) v aplikaci se nachází v příloze č. 2 na str. 103 a v příloze č. 3 na str. 104.



Obrázek 4.3: Návrh doménových objektů pro výsledky vyhledávání a vyhledávací dotaz

4.1.2 Vrstva pro komunikaci s okolím

Moduly v této sekci budeme nazývat *kontrolery*[3]. Kontrolery budou reagovat na podněty z okolí – například příchozí dotaz na zobrazení zvoleného dokumentu bude zpracován a dále postoupen kontrolerem pro dokumenty, konkrétně jeho metodou pro získávání dokumentů.

Každá činnost vystavovaná okolí aplikace I:ZI Repository by proto měla mít svou metodu v kontroleru. Metody lze do kontrolerů sdružovat například podle doménových objektů, případně dalších objektů, se kterými bude aplikace pracovat.

Aplikace proto bude mít kontrolery pro

[3]Toto označení pochází například z aplikační architektury MVC (Model-View-Controller). Popis jednotlivých částí (uvedeno dle `http://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/`)

- Model – datová struktura obsahující data, model také může obsahovat např. výpočty, je v něm tudíž i logika aplikace.
- View – jedná se o způsob vizualizace či prezentace dat obsažených v modelu. Může to být například šablona webové stránky, do které jsou data z modelu vkládána.
- Controller – lze interpretovat jako modul či jednotku, která se stará o provázání modelu a view. Sama nevykonává práci s daty, pouze reaguje např. na akci od uživatele a impuls pošle dále na model (např. změna dat) a view (vykreslení nových dat).

- dokument,
- dotazování (vyhledávání),
- práci s indexy.

Výsledky vyhledávání nemají vlastní kontroler. Pro vyhledávání jako činnost je kontroler navržen – ten bude přijímat vyhledávací dotazy a vracet výsledky vyhledávání.

4.1.3 Vrstva logiky aplikace

Vrstva logiky je z pohledu funkčnosti aplikace nejdůležitější. Na rozdíl od vrstvy pro komunikaci s okolím a komunikaci s datovým zdrojem obsahuje tato vrstva veškerou hlavní logiku aplikace. Tato vrstva je zodpovědná za práci s obdrženými daty, jejich zpracování, úpravy a operacemi s nimi.

Logická vrstva bude obsahovat funkcionalitu pro

- zpracování příchozích dotazů,
- zpracování dokumentů k uložení,
- zpracování výsledků vyhledávání – fuzzy vyhledávání, shlukování, sdružování,
- zpracování dat pro výstup.

4.1.4 Vrstva komunikace s databází

Tato vrstva bude zajišťovat komunikaci s databází pomocí tzv. DAO. DAO, neboli Data Access Object, je objekt pro přístup k datům z datového zdroje, v našem případě z XML databáze. Takový objekt většinou umožňuje data ze zdroje získávat, upravovat či mazat.

Pro každý doménový objekt, se kterým bude v rámci této vrstvy pracovat, bude separátní objekt pro přístup do databáze. DAO budou proto existovat pro

- dokument,
- výsledky vyhledávání,
- index databáze.

Navíc bude součástí této vrstvy taky agregační DAO, které bude zajišťovat informace o seznamu všech dokumentů, jejich počtu a o seznamu všech indexů a jejich počtu.

4.2 Data používaná aplikací

4.2.1 Příchozí dotaz

Nově přidaná funkcionalita s sebou nese potřebu upravit strukturu dotazu, který přichází do aplikace z napojených služeb.

Nastavení vyhledávání

Typ dotazu Součástí vyhledávacího dotazu by měl být způsob, jakým se bude transformovat příchozí vyhledávací dotaz na výsledný dotaz do databáze. V úvahu připadají dvě možnosti, které vznikly z dosavadního používání původní aplikace XQuery search a z potřeb napojených služeb.

- **Dlouhý dotaz** Prvním typem je dotaz, který vyhledává asociační pravidla obsahující minimálně takové atributy, které jsou součástí příchozího dotazu. To znamená, že v databázi budou vyhledávána asociační pravidla, která obsahují alespoň v dotazu uvedené atributy (a jejich aspekty, které jsou případně součástí dotazu) a případně další atributy.
- **Krátký dotaz** Druhým typem dotazu je takový, který vyhledá asociační pravidla, která mají maximálně takové atributy, které jsou v dotazu obsaženy. Tím je myšleno, že dojde k vyhledání asociačních pravidel, která obsahují podmnožinu atributů v dotazu, maximálně všechny atributy z dotazu, ale neobsahuje žádné jiné atributy, v dotazu neuvedené.

Cíl dotazu Dalším nastavením, které je potřeba v dotazu zahrnout, je určení, co je cílem dotazu – zda vyhledávání bude probíhat v asociačních pravidlech (výsledcích běhu dataminingových úloh) či v zadání úlohy. Z hlediska struktury příchozího dotazu není velký rozdíl mezi strukturou pro dotaz na asociační pravidla a pro dotaz na zadání úloh. Nicméně je potřeba vyznačit, které vyhledávání je požadováno, protože rozdílnosti v obou dotazech, například v některých použitých atributech, existují.

Druh analýzy výsledků Třetím nastavením, které je v dotazu uvést, je určení, jaká analýza výsledků bude použita – zda fuzzy ohodnocení, shlukování, sdružování (rozdělování do skupin), případně žádná analýza.

Parametry analýzy Dalším prvkem, který by měl být v dotazu zahrnut jsou parametry některých druhů analýz. Například pro sdružování je třeba určit, jaký aspekt asociačního pravidla bude pro rozdělování do skupin použit.

Legacy výstup Pro komunikaci s okolními službami se používá formát výsledků vyhledávání, který jednotlivé typy elementů sdružuje do stejných míst, tzn. základní booleovské

atributy jsou pospolu, odvozené booleovské atributy taktéž apod. Jednotlivé elementy asociačního pravidla jsou propojeny na základně identifikátorů a odkazů na ně. Vzhledem k tomu, že tento formát je pro člověka hůře přehledný, existuje možnost výstup strukturovat stejně, jako jsou data uložena v databázi – ve stromové struktuře. Formát výstupu je ovlivňován tímto nastavením.

Omezení počtu výstupů V nastavení vyhledávání je také poskytnuta možnost omezení počtu výsledků vyhledávání.

Konkrétní použití nastavení vyhledávání ve vyhledávacím dotazu je ukázáno v příloze č. 5 na str. 106.

Typy dotazů

Vzhledem k různým druhům vyhledávání jsou k dispozici tři druhy vyhledávacích dotazů (pro komunikaci s okolními službami) – dotaz na asociační pravidla (výsledky běhů úloh), dotaz na zadání úloh, hybridní dotaz.

Není potřeba vytvářet zcela novou strukturu dotazu, dostačující je změna dosavadní struktury. Hlavním důvodem je potřeba předávat téměř stejná data, která tvoří strukturu dotazu, jako v původním dotazu. Hlavní změna spočívá v přidání možnosti řídit a nastavovat způsob zpracování výsledků případě způsob utváření dotazu do databáze. Další změnou je menší úprava pro dotaz na zadání úloh a hybridní dotaz.

Dotaz na asociační pravidla Tento druh dotazu byl podporován již v původní aplikaci XQuery search, z tohoto pohledu se nic nemění. Obsažená data jsou dostatečná. Pouze dochází k zavedení informací o nastavení vyhledávání. Informace o nastavení jsou ovšem pouze volitelnou částí dotazu, nemusí tudíž být přítomna. Volitelnost této části zajišťuje zpětnou kompatibilitu s původní aplikací.
Konkrétní ukázka vyhledávacího dotazu (XML data) je v příloze č. 5 na str. 106.

Dotaz na zadání úloh Z hlediska struktury je tento dotaz stejný, jako dotaz na asociační pravidla. Nicméně některé elementy obsahují odlišné typy informací. To pouze znamená, že je dotaz v aplikaci jinak zpracován, struktura zůstává až na detaily stejná. Z tohoto důvodu je také třeba v nastavení vyhledávání určit, že se jedná o vyhledávání v zadáních úloh.
Konkrétní ukázka vyhledávacího dotazu (XML data) je v příloze č. 6 na str. 108.

Hybridní dotaz Tento dotaz umožňuje využít vlastností obou výše uvedených dotazů. Stejně je to i se strukturou dotazu. V oblasti specifikace požadovaných dat (pro vyhledávání) se nacházejí oba dotazy – dotaz na asociační pravidla i dotaz na zadání úloh. Zpracování toho dotazu se rozděluje na samostatná zpracování dotazu na asociační pravidla a dotazu na zadání vyhledávání. Výsledky obou zpracování jsou poté spojeny do výsledného dotazu pro databázi.
Konkrétní ukázka vyhledávacího dotazu (XML data) je v příloze č. 7 na str. 109.

4.2.2 Dataminingové úlohy

Dataminingové úlohy, někdy také v této práci označovány jako dokumenty, tvoří důležitou součást aplikace. Tyto úlohy, které pocházejí ze systému LISp-Miner, jsou ukládány do úložiště. Tato data jsou následně prohledávána na základě obdržených vyhledávacích dotazů.

Přijímaný dokument má tři hlavní části:

- **Popis dat** – také referován jako Data Description. Tato část obsahuje popis dat, která byla použita pro získávání hypotéz systémem LISp-Miner. V popisu dat jsou uvedeny jednotlivé atributy se svými kategoriemi;
- **Zadání dataminingové úlohy** – v této části jsou uváděna omezení a nastavení pro vytváření hypotéz systémem LISp-Miner. Tato část tak obsahuje například specifikace cedentů a dalších vlastností hypotéz, které systém LISp-Miner z poskytnutých dat vytváří;
- **Výsledky dataminingové úlohy** – tato část dokumentu obsahuje hypotézy vytvořené systémem LISp-Miner z dostupných dat na základě zadání dataminingové úlohy. Jedná se o nalezená asociační pravidla.

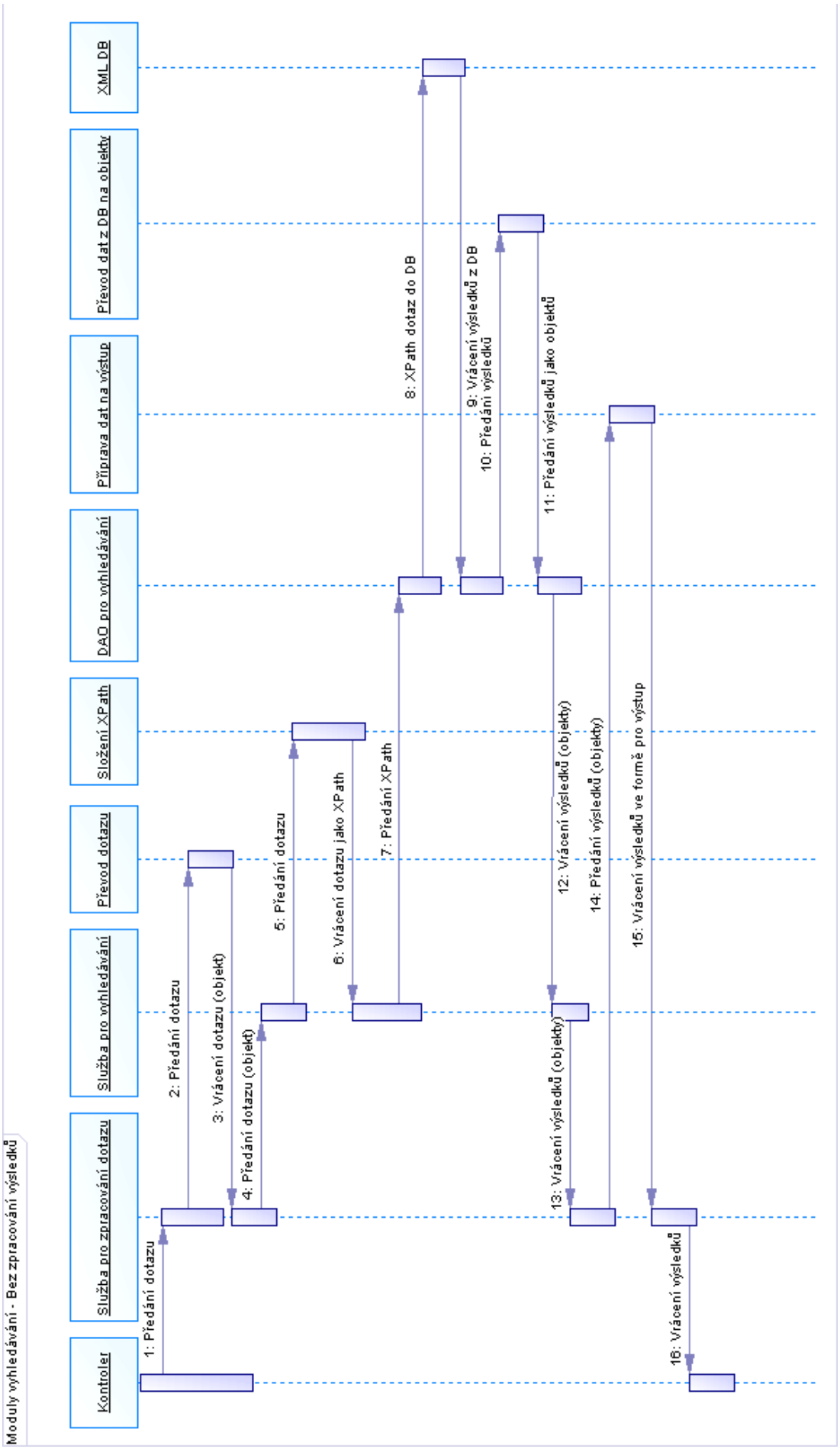
4.3 Základní vyhledávání

Při základním vyhledávání nedochází k žádnému dalšímu zpracování obdržených výsledků z databáze. Pouze k jejich úpravě pro vnitřní zpracování (převod na objekty) a jejich transformaci do formy vhodné pro výstup. Jedná se o základní funkcionalitu, která byla přítomna i v původní aplikaci. V aktuálně navrhované aplikaci I:ZI Repository ovšem budou výsledky i dotaz reprezentovány objekty, což výrazně zjednoduší a zpřehlední operace s nimi při průchodu aplikací.

Proces základního vyhledávání

Popis procesu začíná po přijetí požadavku kontrolerem od vnější služby. Grafický nákres procesu je na obrázku 4.4.

1. Po příchodu požadavku je tento odchycen kontrolerem, který obsluhuje události spojené s vyhledáváním. Po přijetí požadavku kontroler předá obsažený dotaz službě zpracovávající dotaz, která je součástí servisní vrstvy.
2. Služba pro zpracování dotazu jako první předá dotaz modulu pro převod dotazu.
3. Modul převodu dotazu slouží k převedení dotazu z formy textu do formy objektu. Dotaz ve formě objektu je snáze zpracovatelný a použitelný v dalších fázích vyhledávání. Převedený dotaz je jako objekt předán zpět službě pro zpracování dotazu.
4. Služba pro zpracování dotazu následně předá dotaz ve formě objektu službě pro vyhledávání.



Obrázek 4.4: Sekvenční graf – základní vyhledávání

5. Služba pro vyhledávání předá dotaz (objekt) modulu pro převod dotazu do XPath formy. XPath dotaz je třeba pro získání dat z XML databáze.
6. Modul pro XPath převod vrátí dotaz v XPath formě jako textový řetězec.
7. XPath dotaz je předán DAO pro vyhledávání.
8. DAO odešle XPath dotaz do XML databáze.
9. DAO obdrží výsledky z XML databáze ve formě objektu, ve kterém je textově reprezentován výsledek vyhledávání. Každé nalezené asociační pravidlo přichází jako jeden objekt.
10. DAO předá nalezené výsledky modulu pro převod výsledků na objekty.
11. Modul převodu vrátí výsledky jako objekty DAO.
12. DAO vrací výsledky jako objekty službě pro vyhledávání.
13. Služba pro vyhledávání předá obdržené výsledky službě pro zpracování dotazu.
14. Služba pro zpracování dotazu předá výsledky modulu pro převod dat pro výstup.
15. Data ve formě vhodné pro výstup jsou vrácena zpět službě pro zpracování dotazu.
16. Služba pro zpracování dotazu předá výsledky ve vhodné formě kontroleru, který je vrátí jako odpověď vnější službě, která poslala požadavek.

4.4 Návrh modulu pro fuzzy vyhledávání

Mezi nově navrhované funkční celky aplikace patří fuzzy vyhledávání. Funkcionalitu lze rozdělit na dva celky – moduly:

- modul pro vyhledávání v zadání úloh,
- modul pro vyhledávání ve výsledcích úloh (asociačních pravidlech).

Oba celky budou funkčně do jisté míry podobné, rozdíl je ve zpracovávaných datech a jejich struktuře.

Asociační pravidla nalezená v databázi budou porovnávána s dotazem, který aplikace obdržela. Výsledkem porovnávání bude ohodnocené asociační pravidlo. Ohodnocení bude vyjadřovat míru podobnosti daného asociačního pravidla a obdrženého dotazu. Součástí ohodnocení bude také analýza asociačního pravidla.

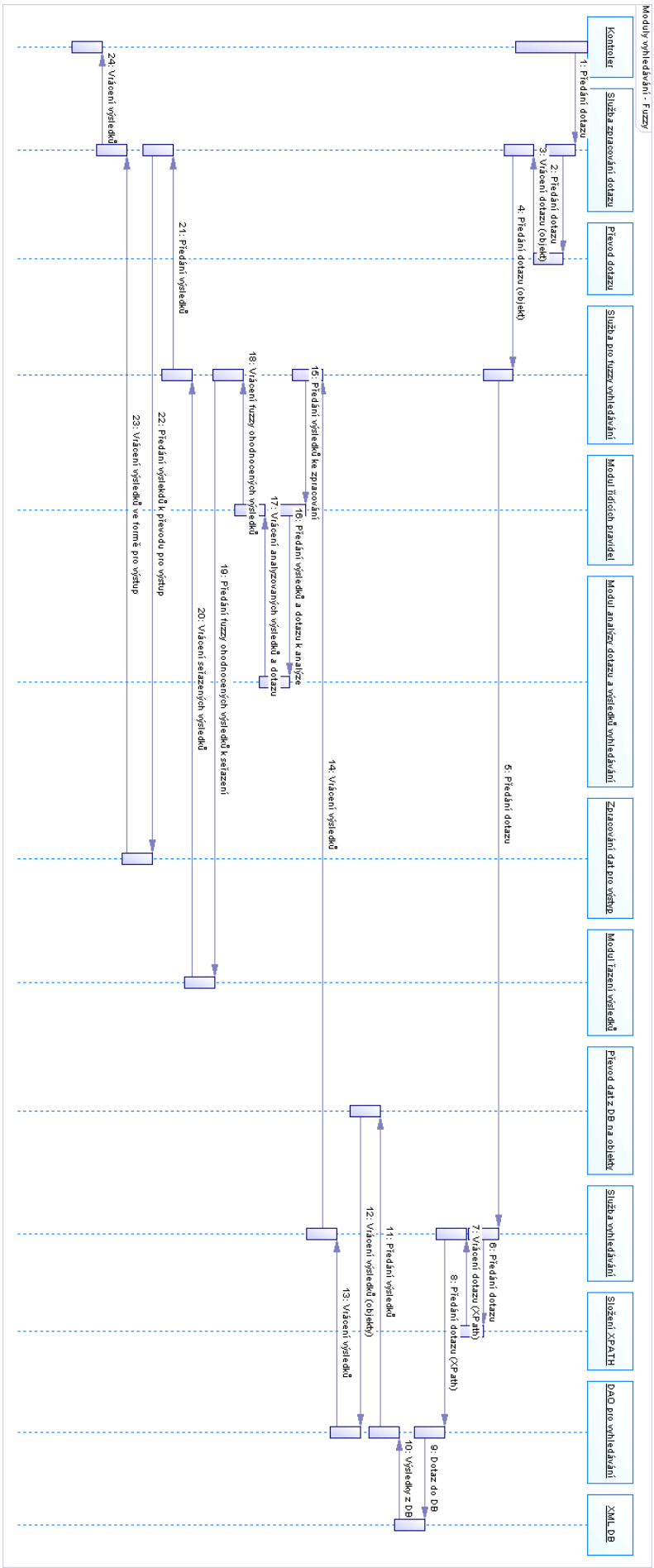
Pravidlo může být plně vyhovující či vyhovující na méně než 100 %. V případném GUI by bylo vhodné nabídnout uživateli pravidla, která plně odpovídají dotazu a v dalším bloku pravidla, která jsou mu doporučována k prozkoumání, protože nevyhovují dotazu na 100 %.

Součástí modulů pro vyhledávání budou i řídící pravidla (viz podkapitola 3.2.1) včetně navazující funkcionality, která budou užita při ohodnocování asociačních pravidel.
Řídící pravidla budou mít dvě složky: *podmínku* a *snížení ohodnocení*.

Proces fuzzy vyhledávání

Popis procesu začíná v momentě obdržení požadavku od vnější služby kontrolerem. Grafické znázornění procesu je na obrázku 4.5.

1. Po příchodu požadavku je tento odchycen kontrolerem, který obsluhuje události spojené s vyhledáváním. Po přijetí požadavku kontroler předá obsažený dotaz službě zpracovávající dotaz, která je součástí servisní vrstvy.
2. Služba pro zpracování dotazu jako první předá dotaz modulu pro převod dotazu.
3. Modul převodu dotazu slouží k převedení dotazu z formy textu do formy objektu. Dotaz ve formě objektu je snáze zpracovatelný a použitelný v dalších fázích vyhledávání. Převedený dotaz je jako objekt předán zpět službě pro zpracování dotazu.
4. Dotaz je ve formě objektu předán službě pro fuzzy vyhledávání.
5. Služba pro fuzzy vyhledávání následně předá dotaz službě pro vyhledávání.
6. Služba pro vyhledávání nejprve předá dotaz ve formě objektu modulu, který slouží k převodu dotazu do jazyka XPath.
7. Dotaz je po převedení modulem do XPath reprezentován opět textově. XPath dotaz je třeba pro získání dat z XML databáze. XPath dotaz je předán zpět službě pro vyhledávání.
8. Služba pro vyhledávání následně předá dotaz jako XPath DAO pro vyhledávání.
9. DAO provede dotaz do XML DB.
10. DAO obdrží výsledky z XML DB ve formě objektu, ve kterém je textově reprezentován výsledek vyhledávání. Každé nalezené asociační pravidlo přichází jako jeden objekt.
11. DAO předá výsledky modulu pro převod výsledků na objekty. Objektová reprezentace bude vhodná pro další zpracování získaných dat.
12. DAO obdrží výsledky převedené na objekty.
13. DAO předá výsledky službě pro vyhledávání.
14. Výsledky vyhledávání reprezentované objekty jsou pro další zpracování předány službě pro fuzzy vyhledávání.
15. Služba pro fuzzy vyhledávání nalezená asociační pravidle ve formě objektů předá modulu řídících pravidel, který provádí ohodnocení jednotlivých asociačních pravidel.
16. Před začátkem zpracování výsledků vyhledávání je provedena analýza dotazu vyhledávání. Ta je provedena pouze jednou před analýzou výsledků vyhledávání.
17. Analyzovaná asociační pravidla i dotaz vyhledávání jsou vráceny modulu řídících pravidel.
    V tomto modulu dojde k porovnání aktuálního prvku výsledku s dotazem. Podle vyhovění původnímu dotazu dochází k ohodnocení aktuálního prvku z výsledků.
18. Ohodnocené prvky jsou navráceny zpět službě pro fuzzy vyhledávání.



Obrázek 4.5: Sekvenční graf – fuzzy vyhledávání

19. Služba pro fuzzy vyhledávání předá ohodnocené výsledky modulu pro řazení fuzzy výsledků.

20. Řadící modul seřadí výsledky podle jejich fuzzy ohodnocení od nejlépe vyhovujícího dotazu sestupně. Seřazené výsledky jsou navráceny službě pro fuzzy vyhledávání.

21. Ohodnocené a seřazené výsledky vyhledávání jsou předány službě pro zpracování dotazu.

22. Tato služba předá obdržená data modulu pro zpracování dat pro výstup.

23. Modul převede výsledky do formy vhodné pro výstup. Ty poté navrátí službě pro zpracování dotazu.

24. Služba pro zpracování dotazu předá výsledky ve vhodné formě kontroleru, který je vrátí jako odpověď vnější službě, která poslala požadavek.

4.5 Návrh modulu pro shlukování

Hlavním účelem modulu pro shlukování je seskupit podobná asociační pravidla získaná vyhledáváním v databázi do shluků, které budou reprezentovány tzv. centroidy (viz podkapitola 3.3.1).

Prvkem je v následujícím textu myšlen výsledek vyhledávání – asociační pravidlo. Shluky se skládají z jednotlivých prvků do shluku patřících a z centroidu, který daný shluk reprezentuje. Centroid obsahuje stejné vlastnosti, jaké jsou zkoumány u prvků – fuzzy ohodnocení. Aby centroid reprezentoval daný shluk, jeho fuzzy ohodnocení musí být průměrem fuzzy ohodnocení prvků daného shluku.

Do daného shluku jsou jednotlivé prvky umisťovány na základě vypočtené podobnosti mezi zkoumaným prvkem a centroidem pro daný shluk. Pro umístění zkoumaného prvku do daného shluku je určena minimální vzdálenost, která pro vložení musí být překročena.

Vlastností, která bude u prvku zkoumána, bude fuzzy ohodnocení podobnosti nalezeného asociačního pravidla a vyhledávacího dotazu. Tato vlastnost je zvolena pro svou vypovídající schopnost a skutečnost, že tuto vlastnost lze snadno získat díky implementaci fuzzy vyhledávání.

Vzhledem k tomu, že ohodnocení asociačního pravidla je vyjádřeno vektorem, lze pro měření vzdálenosti centroidu a aktuálně zkoumaného prvku užít výpočty míry podobnosti dvou vektorů. Modul bude nabízet více možných způsobů pro vypočítání vzdálenosti mezi centroidem a zkoumaným prvkem. Díky faktu, že velikost vektoru vzniká na základě dotazu, který je pro všechny nalezené výsledky stejný, nevznikne zde ani problém s nesourodými vektory[4].

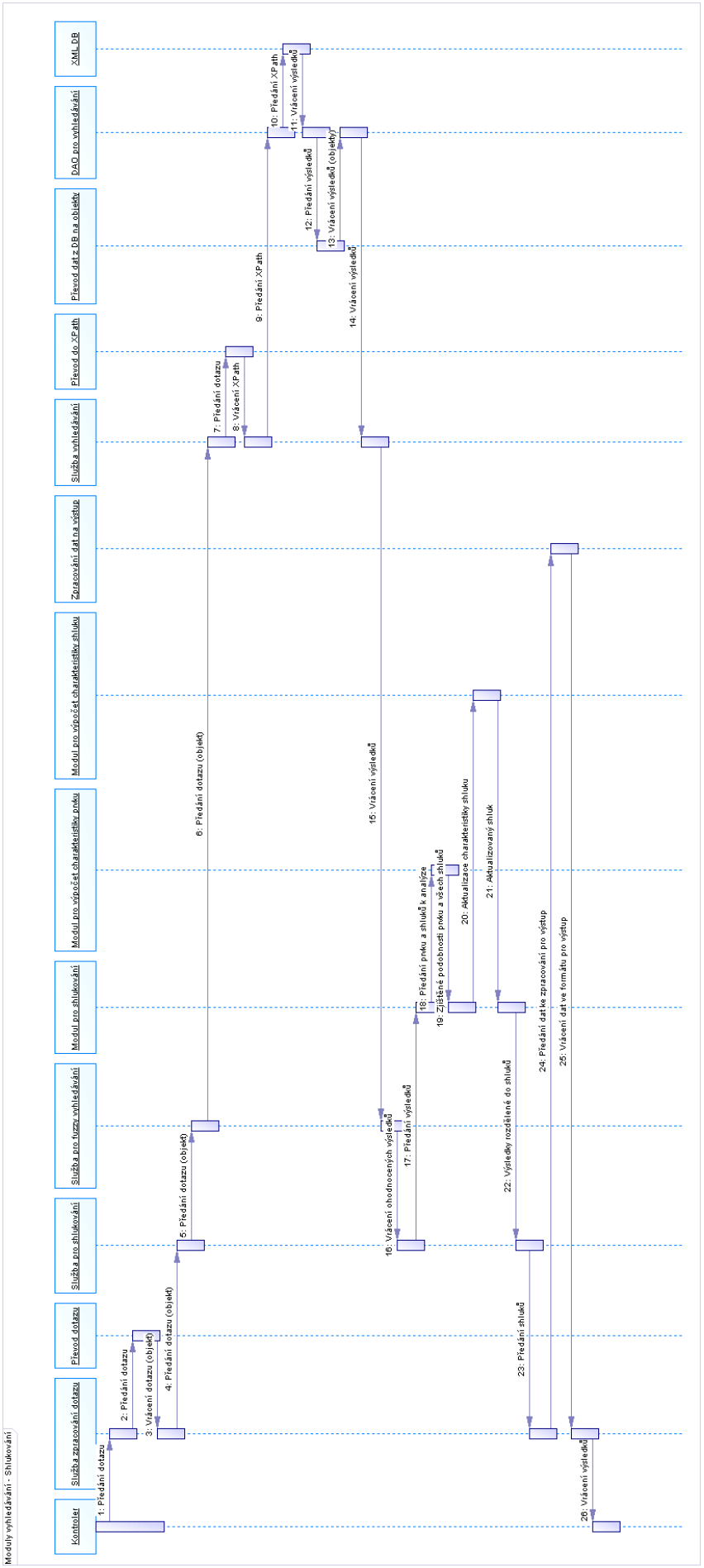
[4]Nesourodými vektory jsou zde myšleny dva vektory, které by měly různý počet prvků, tudíž by nebylo možné je porovnat navrhovaným způsobem.

4.5.1 Proces shlukování

Tato sekce poskytuje popis celého procesu shlukování od přijetí požadavku až po navrácení odpovědi. Popis procesu začíná ve chvíli, kdy je kontrolerem obdržen požadavek od vnější služby. Grafické znázornění procesu je na obrázku 4.6.

1. Po příchodu požadavku je tento odchycen kontrolerem, který obsluhuje události spojené s vyhledáváním. Po přijetí požadavku kontroler předá obsažený dotaz službě pro zpracování dotazu, která je součástí servisní vrstvy.
2. Služba pro zpracování dotazu jako první předá dotaz modulu pro převod dotazu.
3. Modul převodu dotazu slouží k převedení dotazu z formy XML uchované v textovém řetězci do formy objektu. Dotaz ve formě objektu je snáze zpracovatelný a použitelný v dalších fázích vyhledávání. Převedený dotaz je jako objekt předán zpět službě pro vyhledávání.
4. Služba pro zpracování dotazu předá dotaz (reprezentovaný objektem) službě pro shlukování.
5. Služba pro shlukování předá dotaz ve formě objektu službě pro fuzzy vyhledávání. To umožní získat fuzzy ohodnocené výsledky pro další zpracování.
6. Služba pro fuzzy vyhledávání předá dotaz (objekt) službě pro vyhledávání.
7. Služba pro vyhledávání předá dotaz jako objekt modulu pro převod dotazu do XPath formy.
8. Modul XPath převodu vrátí XPath dotaz ve formě textového řetězce.
9. Služba pro vyhledávání předá XPath dotaz DAO pro vyhledávání (objekt pro komunikaci s databází).
10. DAO provede dotaz do XML DB.
11. DAO obdrží výsledky z XML DB ve formě objektu, ve kterém je textově reprezentován výsledek vyhledávání. Každé nalezené asociační pravidlo přichází jako jeden objekt.
12. DAO předá výsledky modulu pro převod výsledků na objekty. Objektová reprezentace bude vhodná pro další zpracování získaných dat. Jednotlivá asociační pravidla jsou převedena na objekty reprezentující výsledek vyhledávání.
13. DAO obdrží výsledky převedené na objekty. Všechny objekty jsou vloženy do obalového objektu sady výsledků.
14. DAO předá výsledky službě pro vyhledávání. Výsledky jsou předány v obalovém objektu obsahujícím všechny nalezené výsledky.
15. Služba pro vyhledávání předá výsledky službě pro fuzzy vyhledávání.
16. Výsledky jsou službou pro fuzzy shlukování ohodnoceny[5]. Výsledky vyhledávání reprezentované objekty jsou pro další zpracování předány službě pro shlukování.
17. Každý jednotlivý prvek ve výsledcích z DB je podroben analýze.

[5]Postup ohodnocení je detailně popsán v sekci o fuzzy vyhledávání



Obrázek 4.6: Sekvenční graf – vyhledávání se shlukováním

18. Již vytvořené shluky a aktuálně zkoumaný prvek jsou předány k analýze podobnosti.
19. Pro každý existující shluk je vrácena míra podobnosti zkoumaného prvku a daného shluku.
20. Prvek je přiřazen do shluku s nejvyšší mírou podobnosti, pokud je tato míra nad stanovenou minimální hranicí. Pokud je nejvyšší míra podobnosti pod touto hranicí, je vytvořen nový shluk. Pro změněný shluk jsou charakteristiky jeho centroidu odeslány k aktualizaci.
21. Shluk s aktualizovaným centroidem je navrácen.
22. Výsledky vyhledávání jsou jako shluky vráceny službě pro shlukování.
23. Shluky jsou službou pro shlukování předány službě pro zpracování dotazu.
24. Získané shluky jsou předány modulu pro zpracování dat na výstup.
25. Data ve formě vhodné pro výstup jsou navrácena.
26. Tato data jsou předána kontroleru, který je přidá do odpovědi.

4.5.2 Doplňující informace k procesu shlukování

Při zpracování prvního prvku dojde k vytvoření nového shluku, který bude obsahovat tento prvek. V tomto případě není třeba tento prvek dále zkoumat a centroid bude utvořen z hodnot daného prvku.

V průběhu zpracovávání nalezených výsledků může dojít k situaci, že prvek změní charakteristiku shluku natolik, že některý prvek či prvky přestanou do daného shluku patřit. Poté by prvky, které nikam nepatří, měly být znovu zpracovány.
V tomto případě by nemělo dojít ke vzniku nekonečné smyčky, situace by v nejhorším případě měla vyústit ke vzniku stejného počtu shluků jako již zpracovaných prvků – každý shluk bude obsahovat pouze jeden prvek. Tento problém bude vznikat především v případě, že je používána hranice příslušnosti do shluku. S takovou hranicí návrh počítá.

4.6 Návrh modulu pro sdružování

Sdružování zajišťuje velmi podobné výstupy jako shlukování[6]. Zatímco moduly pro shlukování zajišťují rozdělení dat do shluků na základě fuzzy ohodnocení jednotlivých prvků, počítání charakteristik shluků a přiřazování prvků do shluků na základě výpočtu podobnosti shluku a prvku, moduly pro sdružování rozdělují výsledky vyhledávání do skupin na základě uživatelem zvoleného pohledu.

Zařazení prvku do skupiny probíhá pouze na základě vstupu od uživatele a analýzy vlastností zkoumaného asociačního pravidla. Uživatel si zvolí, jaký aspekt pravidla bude použit pro rozdělování dat do skupin. Na základě vybraného aspektu je tento prozkoumán u každého analyzovaného pravidla a podle zjištěné hodnoty či vlastnosti vybraného aspektu je asociační pravidlo zařazeno do příslušné skupiny.

[6]Rozdíl mezi shlukováním a sdružováním – viz podkapitola 3.3.

Data jsou následně vrácena rozdělena do skupin, každá skupina je popsána hodnotami či vlastnostmi vybraného aspektu.

4.6.1 Možnosti sdružování nalezených výsledků

Uživatel určuje, jakým způsobem budou data získaná z XML databáze sdružována (rozřazována do skupin). Sdružování do skupin je možné

1. podle názvů atributů obsažených v asociačním pravidle,
2. podle názvů atributů obsažených v asociačním pravidle s reflektováním cedentů,
3. podle kategorií, které obsahuje zvolený atribut,
4. podle počtu atributů v pravidle
5. a podle počtu atributů v pravidle s reflektováním cedentů.

Při prvním způsobu sdružování jsou výsledky zkoumány z pohledu obsažených atributů, podle těch jsou poté rozděleny do skupin. Každá skupina přesně odpovídá obsaženým prvkům a jejich atributům.
Není tudíž možné, aby skupina byla identifikována například pouze podmnožinou atributů, které jsou v jejích prvcích obsaženy. Asociační pravidlo obsahující například atributy s názvy *District, Sex* a *Quality* nemůže patřit do skupiny vymezené pouze atributy *District* a *Sex*. Takové asociační pravidlo bude patřit do skupiny, která je identifikována přesně stejnými atributy, nikoliv méně či více.

Druhý způsob je rozšířením prvního, pracuje také s pozicí atributu v pravidle – bere v potaz, ve kterém cedentu se atribut nachází.

Třetí způsob sdružování se zaměřuje na kategorie vybraného atributu. V tomto případě je jako součást dotazu uveden název atributu, který se bude zkoumat.
Berme v potaz dvě asociační pravidla a fakt, že rozřazování probíhá podle atributu s názvem *District*, který obsahuje názvy měst. První asociační pravidlo bude obsahovat kategorii atributu *Praha*, druhé kategorie *Brno* a *Olomouc*. Rozdělení bude probíhat za stejných podmínek popsaných pro první způsob sdružování – každé z asociačních pravidel bude patřit do jiné skupiny. Kategorie identifikující danou skupinu přesně odpovídají kategoriím obsažených prvků. Kategorie se zkoumají pouze pro vybraný atribut.

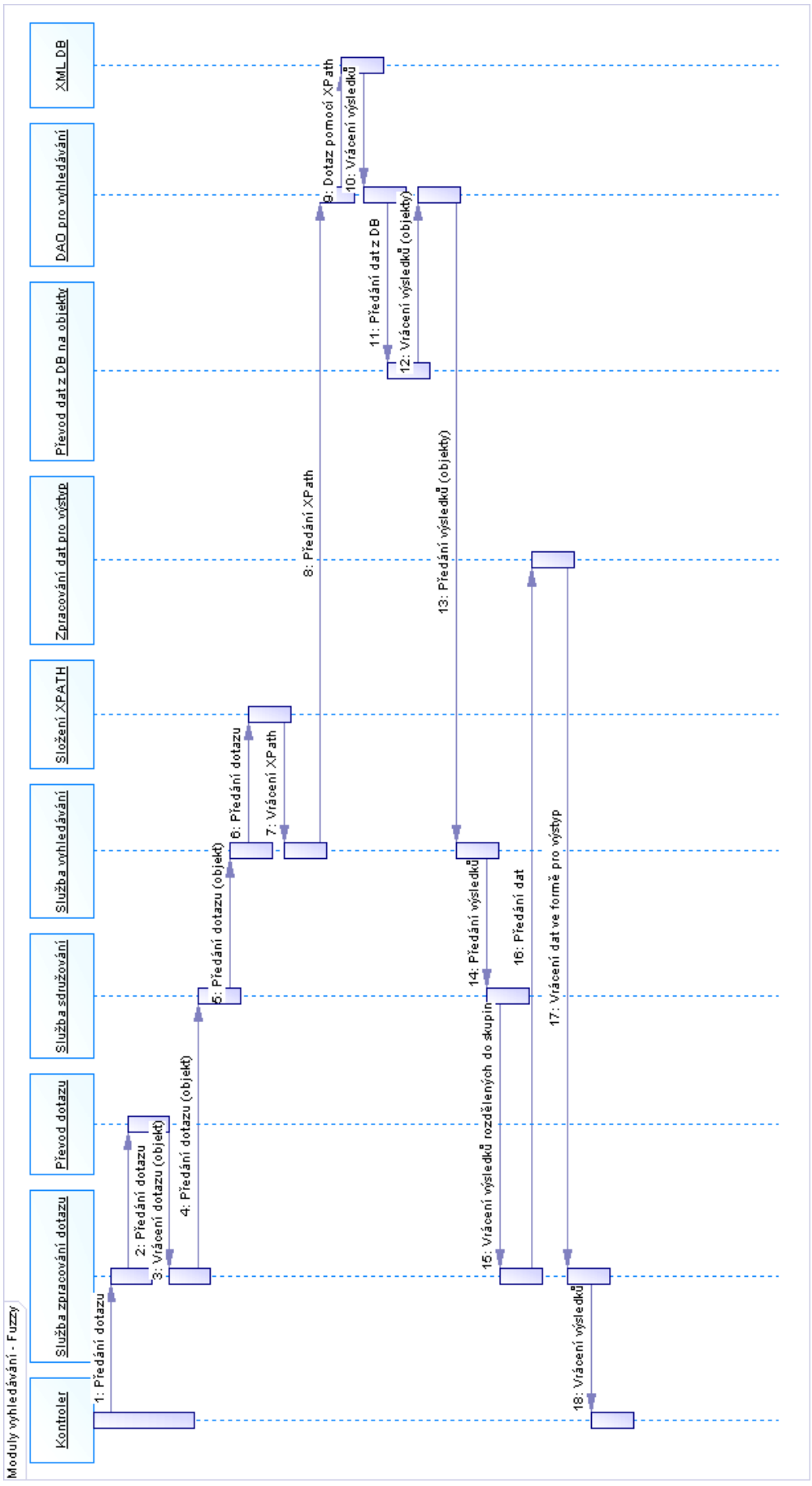
Čtvrtý způsob bere v potaz počet atributů v pravidle. Asociační pravidla jsou tudíž rozdělena do skupin pouze na základě počtu atributů, nikoliv na základě jejich struktury či názvů.

Pátý způsob je téměř totožný se čtvrtým, nicméně zkoumá asociační pravidla z pohledu počtu atributů v jednotlivých cedentech.

4.6.2 Proces sdružování

Popis procesu začíná v momentě přijetí požadavku od vnější služby kontrolerem. Grafické znázornění procesu je na obrázku 4.7.

1. Po příchodu požadavku je tento odchycen kontrolerem, který obsluhuje události spojené s vyhledáváním. Po přijetí požadavku kontroler předá obsažený dotaz službě pro zpracování dotazu, která je součástí servisní vrstvy.
2. Služba pro zpracování dotazu jako první předá dotaz modulu pro převod dotazu.
3. Modul převodu dotazu slouží k převedení dotazu z formy XML uchované v textovém řetězci do formy objektu. Dotaz ve formě objektu je snáze zpracovatelný a použitelný v dalších fázích vyhledávání. Převedený dotaz je jako objekt předán zpět službě pro vyhledávání.
4. Dotaz jako objekt je předán službě pro sdružování.
5. Ta předá dotaz službě vyhledávání pro získání výsledků dotazu.
6. Služba vyhledávání předá dotaz modulu pro převod dotazu do XPath formy.
7. Modul pro XPath převod vrátí převedený dotaz jako textový řetězec.
8. Služba pro vyhledávání následně předá XPath dotaz DAO pro vyhledávání.
9. DAO pro vyhledávání pošle XPath dotaz do XML DB.
10. XML DB vrátí nalezené výsledky ve formě objektu, ve kterém je textově reprezentován výsledek vyhledávání. Každé nalezené asociační pravidlo přichází jako jeden objekt.
11. DAO pro vyhledávání předá nalezené výsledky modulu pro převod výsledků na objekty. Objektová reprezentace bude vhodná pro další zpracování získaných dat. Jednotlivá asociační pravidla jsou převedena na objekty reprezentující výsledek vyhledávání.
12. DAO obdrží výsledky převedené na objekty. Všechny objekty jsou vloženy do obalového objektu sady výsledků.
13. DAO předá výsledky službě pro vyhledávání. Výsledky jsou předány v obalovém objektu obsahujícím všechny nalezené výsledky.
14. Služba pro vyhledávání předá získané výsledky ve formě objektů službě pro sdružování.
15. Služba pro sdružování rozřadí získané výsledky do skupin podle aspektu, který je vybrán uživatelem a je součástí dotazu. Výsledky vyhledávání rozřazené do skupin předá služba pro sdružování službě pro zpracování dotazu.
16. Služba pro zpracování dotazu předá obdržená data modulu pro zpracování dat pro výstup.
17. Data ve formě vhodné pro výstup putují zpět ke službě pro zpracování dotazu.
18. Služba vrátí výsledky kontroleru.



Obrázek 4.7: Sekvenční graf – vyhledávání se sdružováním

4.6.3 Doplňující informace k procesu sdružování

Data mohou být sdružována bez ohledu na to, zda prošla fuzzy ohodnocením či nikoliv. Je to dáno především faktem, že ohodnocení nelze pro sdružování použít. Proto je možné bez problému sdružovat i fuzzy ohodnocené výsledky. V tom případě by se v procesu sdružování objevila ještě služba pro fuzzy vyhledávání, která by se nacházela mezi službou pro zpracování dotazu a službou pro vyhledávání.

4.7 Podpůrné moduly

Kromě popsaných modulů zajišťujících hlavní funkčnost je také třeba mít k dispozici moduly, které budou funkcionalitu podporovat. Takovými jsou moduly určené k transformování, analýze dotazu vyhledávání a výsledku vyhledávání (používaných především ve fuzzy vyhledávání) či moduly pomocné.

Zobrazení napojení podpůrných modulů na hlavní moduly a služby se nachází na obrázku 4.8[7].

4.7.1 Transformace dotazu na objekt

Vyhledávací dotaz přichází do aplikace jako XML ve formě textového řetězce. Pro další zpracování – především převod dotazu na XPath dotaz – je třeba s dotazem pracovat jako s objektem. Možnost zpracování dotazu ve formě objektu přináší především zpřehlednění celého procesu, zjednodušení implementace a možnost jednoduše znovuzískávat data z dotazu, například pro potřeby porovnávání ve fuzzy vyhledávání.

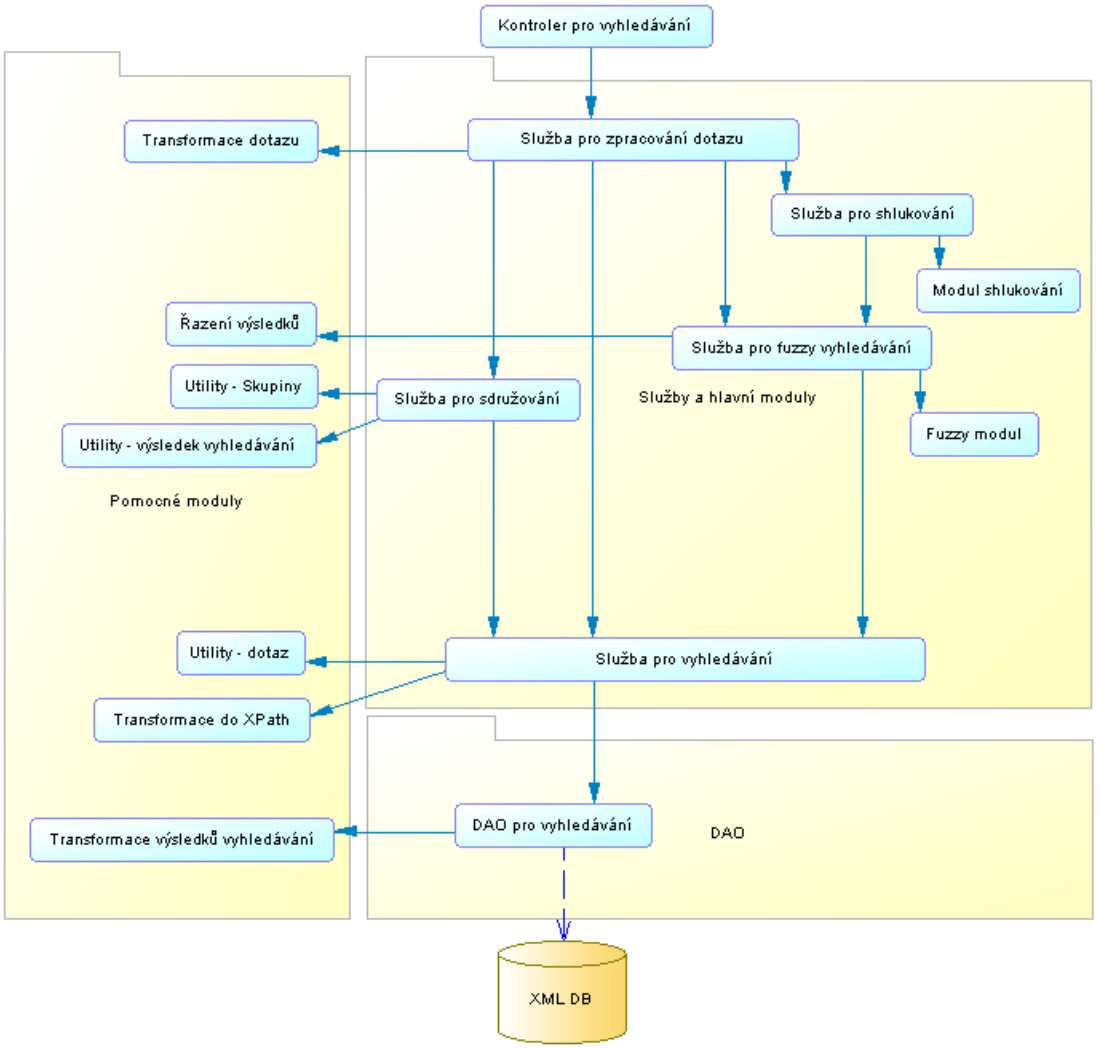
4.7.2 Transformace objektu dotazu na XPath

Pro dotazování do XML databáze je vhodný XPath jazyk. Proto je nutné dotaz, který byl již převeden na objekt, transformovat do XPath. S XPath dotazem je následně nakládáno jako s textovým řetězcem. Objekt dotazu je procházen a postupně transformován do XPath dotazu. Dotaz jako objekt stále zůstává k dispozici pro další zpracování.

4.7.3 Transformace výsledků dotazování na objekty

V případě dotazování pomocí obdrženého XML dotazu převedeného na XPath jsou vráceny výsledky v určité podobě, se kterými je nutné dále pracovat – především fuzzy analýza výsledků a dotazu, shlukování a sdružování. Z tohoto důvodu je nutné, podobně jako v případě dotazu, výsledky vyhledávání převést na Java objekty, aby mohlo k dalšímu zpracování docházet.

[7] Vysvětlení rozdílu mezi službou a modulem se nachází v podkapitole 4.1



Obrázek 4.8: Znázornění objektů – hlavní a pomocné služby a moduly

4.7.4 Transformace pro výstup

Data získaná vyhledáváním, která jsou při průchodu aplikací reprezentována Java objekty, je zapotřebí transformovat zpět do XML formy, která je užita jako výstupní. K dispozici jsou dvě formy výstupu:

- forma kopírující stromovou strukturu, která je užita pro uložení dat v databázi
- a strukturu vhodnou pro zpracování v navazujících systémech (především I:ZI Mineru).

První forma užívá zanoření jednotlivých elementů do sebe, čímž je vytvořena stromová struktura vhodná k prohledávání dat, ale také forma, která je dostatečně čitelná pro člověka.

Podle novější verze definice schématu, které odpovídá druhá výše uvedená forma, jsou jednotlivé druhy elementů sdružovány do skupin – tzn. elementy reprezentující základní booleovské atributy jsou sdruženy v jedné části výstupního formátu, elementy reprezentující odvozené booleovské atributy jsou sdruženy v další části a také elementy reprezentující asociační pravidlo jsou sdruženy v jedné části.

4.7.5 Utility třídy

Utility třídy slouží zejména jako pomocné třídy pro práci s doménovými objekty. Může dojít k situaci, že na různých místech aplikace je třeba s doménovým objektem (či jeho částmi) provádět určité, všude stejné operace. Pro tyto účely je vhodné použít utility třídy, které obsahují pomocné operace pro doménové i jiné objekty.

4.8 Technologické oblasti pro užití při úpravách aplikace

4.8.1 Aplikační framework

Při vývoji webové aplikace za užití Java EE je vhodné použít aplikační framework[8], který může vývoj usnadnit a také snížit objem kódu, který musí vývojář při vytváření aplikace napsat.

Předchozí odstavec ukazuje výhody frameworku a že jeho použití znamená přínos a ulehčení práce. Využití frameworku s sebou také nese ulehčení v podobě možnosti využití funkcionality, kterou framework obsahuje a která by jinak musela být ručně naprogramována a vyvinuta. Také jde o funkcionalitu otestovanou.

Nevýhodou může být zvětšení výsledné velikosti vyvíjené aplikace, protože soubory frameworku je součástí aplikace. Další nevýhodou může být, že některou funkcionalitu fra-

[8]Framework – „*softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovny API, podporu pro návrhové vzory nebo doporučené postupy při vývoji.*“, zdroj: http://cs.wikipedia.org/wiki/Framework

meworku nelze změnit, musí se brát tak, jak je dodávána. Nevýhodou také může být čas nutný ke zvládnutí práce s frameworkem.

4.8.2 Knihovny

Framework nemusí plně obsáhnout podpůrnou funkcionalitu, která je pro vývoj aplikace požadována. Také může implementace dané funkcionality ve frameworku být nedostačující či nevyhovující. Existují proto i samostatné knihovny, které řeší různé oblasti funkcionality.

Výhody využití knihovny jsou velice podobné výhodám frameworku – ulehčení objemu implementace, využití již otestovaných a prověřených funkcionalit.

Mezi nevýhody může patřit zvýšení velikosti aplikace z důvodu nutnosti připojení knihovny, případnou nevýhodou může být i méně vyhovující implementace funkcionality či nutné úpravy stávající aplikace, aby s knihovnou správně spolupracovala.

4.8.3 Srovnání knihovny a frameworku

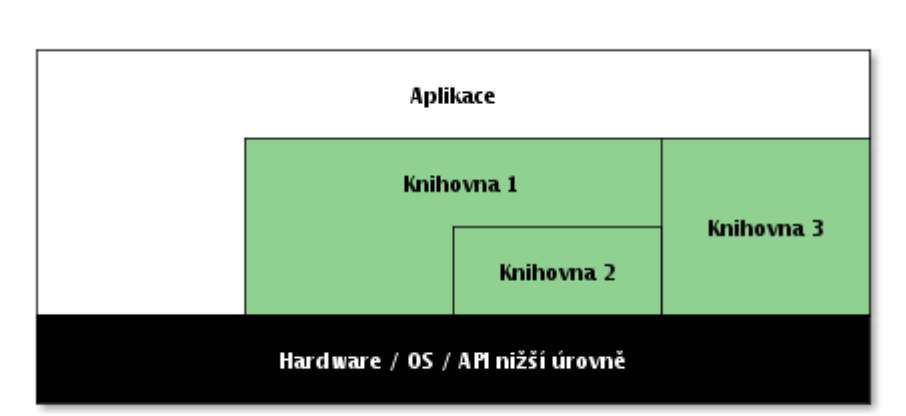
Knihovnu lze definovat jako soubor funkcí zaměřujících se na řešení určité problematiky. Nejedná se o samostatnou aplikaci, cílem je použití knihovny jako součásti vytvářené aplikace. Knihovna by neměla nijak ovlivňovat aplikaci ve smyslu určování, jak bude aplikace fungovat a co bude jak provádět. Působnost knihovny by měla být lokální. Schéma možného užití knihoven je na obrázku 4.9.

Framework se nezaměřuje pouze na jednu určitou problematiku, jedná se spíše o soubor knihoven. Snaží se řešit obecnější problematiku, často se zaměřuje na usnadnění celé implementace aplikace. Framework může aplikaci určovat, jak by co měla dělat. Grafické znázornění užití frameworku je na obrázku 4.10.

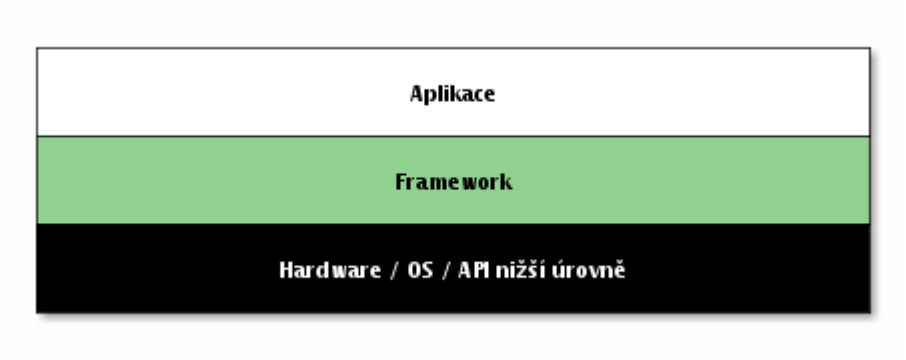
4.8.4 Mapování XML a objektů

Pro účely dalšího zpracování výsledků (asociačních pravidel apod.) získaných z databáze je nutností pracovat s výsledky vyhledávání jako s objekty. Data jsou z databáze získána ve formě XML, proto je potřeba využít technologii, která umožňuje mapování XML dat na Java objekty.

Pro technologii mapování XML na objekty, případně naopak, se používá zkratka OXM – Object-XML Mapping. Existuje mnoho knihoven, které takovou funkcionalitu poskytují pro různé programovací jazyky včetně Javy. Pro potřeby aplikace I:ZI Repository bude využito pouze mapování z XML na objekty a to ve všech výše popsaných případech transformace získaných dat na Java objekty.



Obrázek 4.9: Nákres schématu použití knihovny – Zdroj: `http://majda.cz/zapisnik/265`



Obrázek 4.10: Nákres schématu použití frameworku – Zdroj: `http://majda.cz/zapisnik/265`

Kapitola 5

Implementace

Tato kapitola je zaměřena na popis implementace aplikace I:ZI Repository, změn a vylepšení oproti výchozí aplikaci XQuery search. Podkapitola 5.1 obsahuje informace o použitých technologiích. Podkapitoly 5.2, 5.3, 5.4 a 5.5 jsou zaměřeny na poskytnutí popisu implementace nových funkcionalit či nové implementace již existujících funkcionalit. Podkapitola 5.6 pak obsahuje informace o převodu nestrukturovaných dat ve formě textových řetězců na data strukturovaná reprezentovaná Java objekty. Poslední podkapitola 5.7 se zaměřuje na popis logování použitého pro aplikaci.

Popis implementace se zaměřuje na informování o použitých implementačních řešeních. Uvádí technologie využité při implementaci, řešení problémů vzniklých v předchozích částech procesu vývoje aplikace. Detailně popisuje důležité algoritmy a části aplikace.

5.1 Použité technologie

5.1.1 Databáze

Pro ukládání dat je použita XML databáze Berkeley[1], která patří do rodiny Berkeley databází[2].

Jedná se o tzv. nativní XML databázi, což znamená, že data ukládá jako XML. Nejedná se tudíž pouze o nástavbu například na relační databázi, ale databázi přímo určenou pro uchovávání XML dat.

Tato databáze byla zvolena především z důvodu využití v předchozí verzi aplikace a z toho plynoucích zkušeností s vývojem aplikací používajících právě tuto databázi. Dalším důvodem je dostupnost databáze zdarma, je vyvíjena jako open-source software. Důležitým faktem při výběru byla také aktivní činnost na fóru zabývajícím se podporou a řešením dotazů a problémů spojených s Berkeley XML databází.

[1] Internetová stránka s informacemi o Berkeley XML DB – `http://www.oracle.com/us/products/database/berkeley-db/xml/overview/index.html`

[2] Internetová stránka s informacemi o Berkeley DB produktech – `http://www.oracle.com/us/products/database/berkeley-db/overview/index.html`

Databáze používá tzv. kontejnery. Kontejner by se dal přirovnat svou funkcí k jedné databázi z pohledu relačních databází. Kontejner obsahuje jednotlivé dokumenty, což jsou již konkrétní ukládaná data ve formě XML.

K datům lze přistupovat transakčně, což s sebou nese některé výhody jako je možnost konkurenčního přístupu k datům či udržování konzistentnosti dat v databázi. Při spuštění aplikace se vytvoří objekt, který přístup do databáze řídí. Před každým přístupem do databáze se otevře transakce, v rámci které jsou všechny operace prováděny. Po ukončení je transakce uzavřena.

5.1.2 Framework

Pro usnadnění a zjednodušení implementace byl použit Spring Framework[3]. Spring je jedním z nejznámějších frameworků pro jazyk Java, konkrétně pro vývoj aplikací na platformě Java EE.

Spring framework usnadňuje vývoj Java EE aplikací především v následujících aspektech[4] (z pohledu vývoje enterprise aplikací):

- Pomoc při odstranění těsných programových vazeb jednotlivých POJO[5] objektů a vrstev za pomoci návrhového vzoru Inversion of Control[6].
- Možnost volby implementace (EJB[7], POJO) business vrstvy pro aplikační architekturu a ne naopak (tedy aby architektura předepisovala implementaci).
- Řešení různých aplikačních domén bez nutnosti použití EJB, například transakční zpracování, podpora pro remoting business vrstvy formou webových služeb či RMI[8].
- Podpora implementace komponent pro přístup k datům, ať již formou přímého JDBC[9] či ORM[10] technologií a nástrojů.
- Odstranění závislosti na roztroušených konfiguracích a pracného dohledávání jejich významu.
- Abstrakce vedoucí ke zjednodušenému používání dalších části Java EE.
- Správa a konfigurační management business komponent.

[3]Internetové stránky s informacemi o Spring Framework – `http://www.springsource.org/spring-framework`

[4]Převzato z Interval.cz -Spring Framework – představení J2EE lightweight kontejneru – `http://interval.cz/clanky/spring-framework-predstaveni-j2ee-lightweight-kontejneru/`

[5]POJO – Plain Old Java Object, jedná se o klasický Java objekt, který neslouží k žádným speciálním účelům spojeným například s EJB apod.

[6]Inversion Of Control – obrácené řízení, návrhový vzor, který umožňuje uvolnit vztahy mezi těsně svázanými objekty

[7]EJB – Enterprise Java Beans, serverové komponenty, které umožňují modulární vývoj podnikových aplikací

[8]RMI – Remote method invocation, Java technologie umožňující volání metod na jednom virtuálním stroji z druhého

[9]JDBC – Java Database Connectivity, Java API definující jednotné rozhraní pro přístup k relačním databázím

[10]ORM – Object-relational mapping, objektově relační mapování, zajišťuje automatickou konverzi mezi daty v relační databázi a objekty v aplikaci

Spring Framework byl vybrán především díky zkušenostem s vývojem aplikací na tomto frameworku. Dalším důvodem byla jeho profesionálnost, dostupnost informací a rad na různých internetových fórech a také jeho široké možnosti použití a usnadnění vývoje.

5.1.3 Knihovny

K zajištění některých částí funkcionality bylo třeba využít knihoven, které jsou vytvářeny za účelem usnadnění dané činnosti, kterou poskytují. Mimo knihoven Spring Frameworku a Berkeley XML databáze byly použity tyto knihovny

- Castor XML,
- Quartz,
- Log4j.

Castor XML Castor[11] od ExoLab Group je open-source knihovna, která zajišťuje především mapování XML na Java objekty a naopak. Další oblastí, která nebyla v aplikaci využita, je funkcionalita zaměřená na mapování XML na relační databáze. Pro účely aplikace byla použita verze 1.3.2.

Quartz Knihovna Quartz scheduler[12] od Terracotta, Inc. slouží ke spouštění akcí v nastaveném časovém intervalu. Jedná se o open-source projekt, který umožňuje využít funkcionalitu automatického spouštění činností v jakémkoli projektu postaveném na Java EE či Java SE. Quartz je vydáván pod Apache 2.0 license[13].
V aplikace je použit pro odstraňování starých záznamů příchozích dotazů a odchozích výsledků vyhledávání. Byla použita verze 1.8.6.

Log4j Log4j[14] od Apache Software Foundation je logovací knihovna. Slouží k ukládání informací vzniklých při běhu aplikace. Jedná se o část většího open-source projektu Apache logging services[15], je rovněž vydáván pod Apache 2.0 license.
V aplikaci je použit pro ukládání záznamů o činnosti aplikace, které mohou posloužit pro zjišťování chyb vzniklých při běhu aplikace. Byla použita verze 1.2.16.

Mimo vyjmenovaných byly použity další knihovny, které již zmíněné knihovny potřebují pro svou správnou funkčnost. Tyto knihovny nicméně není třeba dále představovat, z hlediska hlavní funkčnosti aplikace není třeba se s nimi detailněji zabývat.

[11] Internetové stránky s informacemi o Castor XML – `http://www.castor.org/index.html`
[12] Internetové stránky s informacemi o Quartz Scheduler – `http://quartz-scheduler.org/`
[13] Internetové stránky s informacemi o Apache 2.0 license – `http://www.apache.org/licenses/LICENSE-2.0`
[14] Internetové stránky s informacemi o log4j – `http://logging.apache.org/log4j/1.2/`
[15] Internetové stránky s informacemi o Apache logging services – `http://logging.apache.org/`

5.2 Základní vyhledávání

Základní vyhledávání stále zůstává hlavní funkcí aplikace. Základním hledáním je z pohledu způsobu užití aplikace myšleno vyhledávání bez dalšího zpracování výsledků.

Z pohledu samotné aplikace je vyhledávání klíčovou funkcionalitou, která umožňuje získávat data z XML databáze na základě dotazu. Bez funkcionality vyhledávání v XML databázi by aplikace jako taková neměla data k jejich dalšímu zpracování.

Především z důvodu své důležitosti a také případné zpětné kompatibility s původní aplikací (především pro ostatní služby napojené na původní aplikaci) musela být tato funkcionalita zachována z vnějšího pohledu beze změn.

Změna práce s daty

Z důvodu výše popsané potřeby zachování zpětné kompatibility nedošlo u vyhledávání z pohledu přijímaných a poskytovaných dat k velkým změnám. Značné změny se ovšem udály uvnitř aplikace.

Hlavní změna se týká práce s objekty – přijímaným dotazem vyhledávání, získanými výsledky z databáze apod. V původní aplikaci byla tato data při průchodu aplikací reprezentována pouze textově. Kvůli tomu nemohlo docházet k jejich širší analýze, případně bylo zpracování takových dat velice náročné na implementaci, což s sebou neslo také snazší vytváření chyb v kódu.

Tyto nevýhody byly odstraněny převedením dat na objekty. Tím se zpracovávaná data převedla z pohledu aplikace z formy nestrukturované na formu z větší části strukturovanou.

Objektově jsou nyní reprezentována všechna data, která aplikace používá – příchozí dotaz, data získaná z databáze a transformované objekty pro vnitřní použití v aplikaci. Všechna vyjmenovaná data lze poměrně snadno převést na objekty a jejich vlastnosti, protože struktura dat je předem známa a neměla by se od předpokladů odchylovat. V případě, že k odchýlení dojde, lze to považovat za chybu v datech.

Přímý dotaz

Jedinou výjimkou ze zpracovávaných dat, která není převáděna na objekt, je přímý dotaz. Přímý dotaz je převeden na objekt jako jeden celek bez dalšího rozkládání či strukturování – to je nutné, aby data mohla projít aplikací.

Nejedná se o dotaz v XML struktuře, který je určen pro komunikaci mezi aplikací a ostatními službami. Přímý dotaz slouží především k nahodilému získávání informací z databáze, je určen k účelům administrování aplikace.

Přímý dotaz může být napsán v XPath nebo XQuery jazyce. Tento dotaz není nijak aplikací analyzován ani zpracováván. Tak, jak ho aplikace obdrží, je odeslán do XML databáze.

Výsledky vyhledávání tímto dotazem taktéž nejsou nijak zpracovány a ve formě obdržené z databáze jsou vráceny na výstup. Data jsou pouze zabalena do XML elementu reprezentujícího odpověď od aplikace.

Výhody přístupu

Mezi hlavní východy převádění dat na objekty lze zmínit usnadnění nakládání s daty v aplikaci. Díky lepší strukturovanosti dat lze snadněji provádět úpravy již existující implementace, hledat chyby v existující implementaci či implementovat nové funkce. Tato výhoda je významná z pohledu budoucích úprav aplikaci či přidávání nových funkcí.

Další výhodou je zpřehlednění kódu aplikace. Ta umožní lepší orientaci v kódu, která je přínosná pro vývojáře aplikace a to i pro takového, který vidí aplikaci poprvé či se s ní začíná seznamovat.

Poslední výhodou zmíněnou v této části je lepší provázanost struktury aplikace a faktického pohledu na příchozí data, provázanost s objekty reálného světa a chápání příchozích dat.

Nevýhody přístupu

Mezí nevýhodami lze uvést počáteční časovou investici. Před samotnou implementací změn je potřeba vybrat vhodný způsob, jak budou změny implementovány, případně vybrat podpůrnou knihovnu, která vývoj usnadní.

Další nevýhodou je nutnost vytvoření množství nových doménových objektů, za pomoci kterých budou data v aplikaci reprezentována. Především složitější struktury zpracovávaných dat, jako v tomto případě, s sebou nesou potřebu vytvářet velký počet objektů, dalo by se říci podobjektů, ze kterých se skládají objekty nadřazené.

Vytvoření někdy velkého množství objektů s sebou také nese menší přehlednost aplikace a nárůst kódu, který nevykonává žádnou přidanou hodnotu – většinou se jedná o metody pro nastavování a získávání proměnných objektů.

5.3 Fuzzy vyhledávání

Úkolem fuzzy vyhledávání je získání výsledků z databáze podle dotazu. Tyto výsledky jsou následně ohodnoceny z pohledu podobnosti výsledku a dotazu a také z pohledu vlastností získaného výsledku vyhledávání. Každý výsledek vyhledávání je poté ohodnocen vektorem, který vyjadřuje míru podobnosti výsledku a dotazu a ohodnocení vlastností výsledku.

5.3.1 Algoritmus fuzzy ohodnocení

Popis algoritmu začíná ve chvíli, kdy kontroler předá požadavek službě pro fuzzy vyhledávání. Parametry požadavku jsou příchozí dotaz jako objekt a objekt nastavení vyhledávání.

Metoda ZiskejFuzzyVysledkyPomociDotazu

Vstup: Objekt příchozího dotazu, Objekt nastavení vyhledávání
Výstup: Seřazené ohodnocené výsledky

```
1  pokud dotaz je null pak
2  |  vrať null
3  konec podmínky
4  získej List[a] výsledků voláním služby pro vyhledávání
5  převeď objekt příchozího dotazu na objekt vhodný pro analýzu
6  pokud výsledky nejsou null pak
7  |  pro každý výsledek proveď
8  |  |  pokud výsledek není null pak
9  |  |  |  získej Data Description pomocí služby pro Data Description
10 |  |  |  převeď výsledek vyhledávání na objekt vhodný pro analýzu
11 |  |  |  získej vektor popisující vyhovění aktuálního výsledku příchozímu dotazu
   |  |  |  pomocí metody Ohodnot()          /* Popis metody níže */
12 |  |  |  nastav aktuálnímu výsledku vyhovění dotazu
13 |  |  konec podmínky
14 |  konec cyklu
15 konec podmínky
16 seřaď ohodnocené výsledky
17 vrať seřazené výsledky
```

[a] List je datová struktura v Javě, umožňuje uchovávat libovolné množství objektů stejného typu.

Metoda Ohodnot

Vstup: Objekt výsledku vyhledávání (forma pro analýzu), Objekt dotazu (forma pro analýzu)

Výstup: Dvourozměrný vektor s ohodnocením výsledku

1 nastav výchozí podobnost objektů na 100 %

2 proveď analýzu výsledku vyhledávání a ulož do nového objektu

3 proveď analýzu dotazu a ulož do nového objektu

4 sniž míru podobnosti objektů o penalizaci získanou na základě počtu základních booleovských atributů obou objektů

5 sniž míru podobnosti objektů o penalizaci získanou na základě zajímavosti určené anotací

6 získej vektor podobností pro jednotlivé základní booleovské atributy v antecedentech obou objektů

7 získej vektor podobností pro jednotlivé základní booleovské atributy v consequentech obou objektů

8 získej vektor podobností pro jednotlivé základní booleovské atributy v condition cedentech obou objektů

9 slož dvourozměrný vektor z vektorů podobností pro jednotlivé cedenty a z celkového ohodnocení výsledku

10 vrať finální vektor

5.3.2 Sestavení vektoru ohodnocení

Výsledkem ohodnocení nalezeného výsledku vyhledávání je dvourozměrný vektor. Tento vektor se skládá z několika částí:

- **vektor ohodnocení antecedentu** – vzniká na základě porovnání konkrétnosti jednotlivých základních booleovských atributů v antecedentech, případné disjunktnosti základních booleovských atributů (pouze pro výsledek vyhledávání) a na základě počtu chybějících/přebývajících kategorií pro základní booleovské atributy v nalezeném výsledku vyhledávání oproti dotazu,
- **vektor ohodnocení consequentu** – stejné jako předchozí bod ovšem pro consequent,
- **vektor ohodnocení condition** – stejné jako předchozí bod ovšem pro condition,
- **ohodnocení vlastností výsledku vyhledávání** – vzniká na základě absolutního rozdílu počtu základních booleovských atributů v dotazu a výsledku vyhledávání, dále podle ohodnocení zajímavosti/nezajímavosti uživatelem (pokud je k dispozici).

5.3.3 Vlastnosti asociačního pravidla využité při fuzzy hodnocení

Pro fuzzy ohodnocení podobnosti nalezeného asociačního pravidla a dotazu byly zvoleny vlastnosti, které jsou snadno získatelné z uložených či přijatých dat. Také bylo důležité co nejvíce využít vlastností, které jsou k dispozici v obou zkoumaných objektech, vzhledem k tomu, že má docházet ke zjištění podobnosti obou objektů.

Pro fuzzy ohodnocení nalezeného asociačního pravidla bylo mimo vlastností nacházejících se v dotazu a asociačním pravidle využito i vlastnosti, která je součástí pouze asociačního pravidla – ohodnocení zajímavosti či nezajímavosti asociačního pravidla uživatelem.

Pro získání ohodnocení výsledku vyhledávání tedy slouží následující aspekty

- počet základních booleovských atributů,
- počet kategorií pro každý základní booleovský atribut,
- konkrétnost základních booleovských atributů,
- disjunktnost základních booleovských atributů (pouze asociační pravidlo),
- určená zajímavost/nezajímavost získaná od uživatele (pouze asociační pravidlo).

Pro každý aspekt musí být v konfiguračním souboru určena penalizace za odchylky vlastností asociačního pravidla od dotazu.

Získané hodnoty jsou ve výsledku uloženy do dvourozměrného vektoru. Jednotlivé hodnoty, které jsou ve vektoru uloženy, vznikají odečítáním penalizací získaným podle výše uvedených aspektů od výchozí 100% podobnosti.

Počet základních booleovských atributů Pokud má asociační pravidlo více či méně (v závislosti na druhu vyhledávání) základních booleovských atributů než dotaz, je to považováno za odchylku od dotazu a penalizováno. Penalizace v tomto případě zohledňuje absolutní rozdíl mezi počty základních booleovských atributů v dotazu a asociačním pravidle, jako řídící se užívá hodnota pro dotaz. Rozdíl je poté vynásoben hodnotou penalizace.

Počet kategorií základních booleovských atributů Podobně jako počet základních booleovských atributů je zkoumán i počet kategorií každého základního booleovského atributu. Lze určit rozdílné penalizace za chybějící i přebývající kategorie. Penalizace je načítána za každou chybějící/přebývající kategorii.

Konkrétnost základních booleovských atributů Konkrétnost je určována na základě dat získaných z Data description[16] elementu obou objektů (dotazu i asociačního pravidla). Data description obsahuje informace o datech použitých při data miningu. Nejdůležitějšími daty, která z pohledu fuzzy ohodnocení Data description obsahuje, jsou kategorie pro všechny dostupné základní booleovské atributy. Na základě těchto informací je pomocí následujícího vzorce určen poměr konkrétnosti použitých kategorií pro základní booleovský atribut v dotazu a v asociačním pravidle[17].

[16]Data Description = popis dat použitých pro vytváření hypotéz v rámci dataminingové úlohy systémem LISp-Miner – více podkap. 4.2.2

[17]Příklad: asociační pravidlo obsahuje ve zkoumaném základním booleovském atributu dvě kategorie (např. Praha, Brno). Celkem je k dispozici dle Data description deset kategorií (deset měst ČR). Základní booleovský atribut v dotazu obsahuje tři kategorie (např. Praha, Brno, Plzeň) taktéž z deseti možných kategorií. V tom případě je výsledný poměr $\frac{\frac{2}{10}}{\frac{3}{10}} \doteq 0,66667$. To znamená, že kategorie v základním booleovském atributu v asociačním pravidle jsou konkrétnější než ty v dotazu, proto nebude aplikována penalizace.

Vzorec pro určení poměru konkrétnosti základního booleovského atributu v nalezeném výsledku a v dotazu:

$$\frac{\frac{A}{B}}{\frac{C}{D}}$$

pro
A ... počet kategorií použitých v základním booleovském atributu ve výsledku vyhledávání,
B ... počet kategorií celkem k dispozici pro daný atribut ve výsledku vyhledávání,
C ... počet kategorií použitých v základním booleovském atributu v dotazu,
D ... počet kategorií celkem k dispozici pro daný atribut ve dotazu.

Pomocí tohoto vzorce lze určit, zda je z pohledu použitých kategorií základní booleovský atribut v asociačním pravidle méně konkrétní než ten v dotazu. To nastane v případě, že výsledná hodnota vypočítaná pomocí vzorce je větší než 1. V tomto případě je hodnota násobena hodnotu penalizace.

Disjunktnost základních booleovských atributů Tento aspekt se již nezaměřuje na určování podobnosti mezi asociačním pravidlem a dotazem, nicméně přináší do fuzzy hodnocení asociačního pravidla zaměření na strukturu a vlastnosti pravidla samotného. V tomto případě se jedná o snižování fuzzy hodnocení asociačního pravidla na základě spojek základních booleovských atributů[18]. Disjunktní spojení základních booleovských atributů je považováno za méně žádoucí než spojení konjuktní. Je to především z důvodu předpokladu, že uživatele budou spíše zajímat pravidla konkrétnější, než pravidla nekonkrétní.
Pokud má základní booleovský atribut disjunktní spojku, je penalizován.

Určená zajímavost/nezajímavost získaná od uživatele Součástí ukládaných informací o asociačním pravidle může také být uživatelovo označení zajímavosti či nezajímavosti pravidla (pokud je k dispozici). Pokud je taková informace k dispozici, měla by se promítnout do fuzzy hodnocení asociačního pravidla. Z toho důvodu je asociační pravidlo označené jako nezajímavé penalizováno.

5.4 Shlukování

5.4.1 Algoritmus shlukování

Popis algoritmu začíná ve chvíli, kdy kontroler předá požadavek službě pro shlukování. Parametry požadavku jsou příchozí dotaz jako objekt a objekt nastavení vyhledávání. Nejprve dochází k získání výsledků fuzzy vyhledávání. Poté je volána metoda *ShlukujVysledkyVyhledavani*. Při tomto prvním volání metody je jako parametr *List shluků* předáván prázdný,

[18] Spojky jsou fakticky vyjádřeny v odvozených booleovských atributech, nicméně se vztahují zejména k základním booleovským atributům, jelikož vyjadřují způsob jejich vztahu – konjunkce či disjunkce.

nově vytvořený List shluků.

Metoda ShlukujVysledkyVyhledavani

Vstup: List výsledků fuzzy vyhledávání, Objekt nastavení vyhledávání, List shluků
Výstup: Výsledky fuzzy vyhledávání rozdělené do shluků

```
1  pokud List výsledků vyhledávání je null pak
2  |  vrať prázdný List shluků
3  konec podmínky
4  vytvoř proměnou limit, která bude obsahovat limit pro příslušnost do shluku
5  pokud limit příslušnosti je součástí nastavení vyhledávání z dotazu pak
6  |  nastav proměnné limit získanou hodnotu
7  jinak
8  |  nastav proměnné limit hodnotu 0.9
9  konec podmínky
10 vytvoř proměnou formula, která bude obsahovat zvolený vzorec pro výpočet vzdálenosti
   zkoumaného výsledku vyhledávání od shluků
11 pokud volba vzorce pro výpočet je součástí nastavení vyhledávání z dotazu pak
12 |  nastav proměnné formula získanou hodnotu
13 konec podmínky
14 pro všechny výsledky vyhledávání proveď
15 |  pokud List shluků je prázdný pak
16 |  |  vytvoř nový shluk z aktuálního výsledky vyhledávání, centroid nového shluku má
   |  |  stejné vlastnosti jako aktuální výsledek vyhledávání
17 |  |  vlož nově vytvořený shluk do Listu shluků
18 |  jinak
19 |  |  získej pro aktuální výsledek vyhledávání podobnost s každým z existujících shluků
20 |  |  zjisti shluk s nejvyšší podobnosti k aktuálnímu výsledku vyhledávání
21 |  |  pokud podobnost nejpodobnějšího shluku a aktuálního výsledku vyhledávání < limit
   |  |  pak
22 |  |  |  vytvoř nový shluk z aktuálního výsledku vyhledávání a vlož ho do Listu shluků
23 |  |  jinak
24 |  |  |  do vybraného shluku vlož aktuální výsledek vyhledávání jako nový prvek
25 |  |  |  aktualizuj vlastnosti centroidu vybraného shluku
26 |  |  konec podmínky
27 |  konec podmínky
28 konec cyklu
29 vytvoř prázdný List, který bude obsahovat prvky shluků k přepočítání
30 pro všechny existující shluky proveď
31 |  pro všechny prvky aktuálního shluku proveď
32 |  |  zjisti podobnost prvku a centroidu shluku
33 |  |  pokud podobnost < limit pak
34 |  |  |  vyjmi aktuální prvek ze shluku
35 |  |  |  přidej aktuální prvek do Listu pro znovu-zpracování
36 |  |  konec podmínky
37 |  konec cyklu
38 konec cyklu
39 pokud List s prvky pro znovu-zpracování není prázdný pak
40 |  zavolej metodu ShlukujVysledkyVyhledavani()
41 konec podmínky
42 vrať List shluků
```

5.4.2 Použité míry pro výpočet podobnosti shluku a prvku

Při zjišťování, zda aktuálně zkoumaný výsledek vyhledávání může či nemůže patřit do některého ze shluků, se využívají koeficienty podobnosti shluku a výsledku vyhledávání. Pro tyto účely bylo použito dvou vzorců, které podle doc. Strossy byly v praxi ověřeny jako úspěšné (uvedeno dle [19], kap. 2.2.3):

- **Kosínová míra podobnosti** vyjadřuje kosinus odchylky dvou zkoumaných vektorů. Míra nezávisí na velikosti vektorů, protože úhel mezi nimi je stále stejný. Maximální podobnost ($\sigma = 1$) nastane ve chvíli, kdy vektory mají stejný směr (například jeden je násobkem druhého). Kosínovou míru podobnosti lze vypočítat na základě vzorce

$$\sigma(\overrightarrow{u}, \overrightarrow{v}) = \frac{\overrightarrow{u}}{|\overrightarrow{u}|} \cdot \frac{\overrightarrow{v}}{|\overrightarrow{v}|}.$$

  Pro použití v aplikaci byl vzorec přepsán jako

$$\sigma(\overrightarrow{u}, \overrightarrow{v}) = \frac{\sum_{i=1}^{n} u_i \times v_i}{\sqrt{\sum_{i=1}^{n} (u_i)^2} \times \sqrt{\sum_{i=1}^{n} (v_i)^2}}.$$

- **Míra (symetrického) překrytí** se vypočítá pomocí vzorce

$$\sigma(\overrightarrow{u}, \overrightarrow{v}) = \frac{\sum_{i=1}^{n} \min(u_i, v_i)}{\min(\sum_{i=1}^{n} u_i, \sum_{i=1}^{n} v_i)}.$$

  Míra překrytí je podobná Jaccardově míře podobnosti[19], vypočtená hodnota reprezentuje míru překrytí dvou objektů, v tomto případě vektorů.

Vzhledem k tomu, že ohodnocení výsledků vyhledávání je uloženo jako vektor, lze bez problémů použít právě výše uvedené vzorce pro výpočet podobnosti.

V nastavení vyhledávání, které je součástí příchozího dotazu, lze určit v rámci parametrů zvolené analýzy, kterou míru pro výpočet použít. Lze také nastavit hranici pro příslušnost do shluku. Pokud jsou parametry vynechány, použije se jako výchozí kosínová míra a hranice pro příslušnost do shluku je nastavena na 0,9.

5.5 Sdružování

Sdružování se od shlukování liší způsobem vkládání do skupin (u shlukování do shluků). Shlukování využívá pro přidělování do shluků výpočty na základě fuzzy ohodnocení asociačních pravidel nalezených vyhledáváním. Sdružování nepoužívá žádné výpočty, do skupin zařazuje asociační pravidla na základě uživatelem zvolené vlastnosti asociačního pravidla, které je součástí vyhledávacího dotazu.

[19] Jaccardova míra podobnosti měří podobnost dvou objektů, je určena velikostí průniku objektů dělenou velikostí sjednocení objektů. neboli

$$\sigma(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

Sdružování probíhá na základě (viz kap. 4.6)

- vybrané kategorie, kterou obsahuje základní booleovský atribut,
- základních booleovských atributů v asociačním pravidle obsažených,
- základních booleovských atributů v asociačním pravidle obsažených s rozlišením cedentů,
- délky asociačního pravidla (počtu základních booleovských atributů),
- délky asociačního pravidla s rozlišením cedentů.

Především pro účely sdružování vznikla utility[20] třída pro výsledky vyhledávání. Metody této třídy využívají i jiné objekty, v případě služby pro sdružování je ovšem jejich využití velice významné. Díky využití této utility třídy bylo možné zkrátit, zjednodušit a zpřehlednit kód služby pro sdružování.

Pro usnadnění práce a možnost znovupoužití implementovaných operací vznikla také utility třída pro skupiny.

Algoritmus sdružování

Popis algoritmu začíná ve chvíli přijetí požadavku na sdružování výsledků vyhledávání službou pro sdružování. Popis je obecný, zaměřený na kostru všech algoritmů pro jednotlivé

[20] Jedná se o pomocnou třídu, která velice často nevyžaduje vlastní instanci (má pouze statické metody). Slouží k usnadnění a možnosti znovu-použití kódu, který pracuje v tomto případě s objektem výsledku vyhledávání. Nabízí metody jako například *Získej z výsledku vyhledávání všechny základní booleovské atributy* nebo *Získej všechny kategorie pro základní booleovský atribut určený názvem.*

druhy sdružování, které jsou si až na detaily velice podobné.

Metoda RozdelVysledkyDoSkupin

Vstup: List výsledků fuzzy vyhledávání, Objekt nastavení vyhledávání
Výstup: Výsledky vyhledávání rozdělené do skupin

```
1  pokud List výsledků vyhledávání je null pak
2  |  vrať null
3  konec podmínky
4  získej druh sdružování z nastavení vyhledávání
5  pokud druh sdružování není k dispozici pak
6  |  vrať null
7  konec podmínky
8  podle typu sdružování z nastavení vyber odpovídající metodu
9  vytvoř nový List skupin
10 pro každý výsledek vyhledávání proveď
11 |  získej vlastnosti výsledku vyhledávání potřebnou pro aktuální druh sdružování
12 |  získej skupinu pro získané vlastnosti pokud existuje, ulož do proměnné group
13 |  pokud group je null pak
   |  |  /* odpovídající skupina nenalezena                              */
14 |  |  vytvoř novou skupinu
15 |  |  vlož aktuální výsledek vyhledávání do nové skupiny
16 |  |  nastav nové skupině popis podle vlastností aktuálního výsledku vyhledávání
17 |  |  vlož novou skupinu do Listu skupin
18 |  jinak
   |  |  /* odpovídající skupina nalezena                                */
19 |  |  vlož aktuální výsledek vyhledávání do nalezené skupiny
20 |  konec podmínky
21 konec cyklu
22 vrať List skupin
```

5.6 Převod dat na objekty

Pro zlepšení přehlednosti, logičnosti, upravitelnosti a opravitelnosti nakládání s přijatými objekty (dotazy, výsledky vyhledávání) bylo zapotřebí data přijímaná jako XML ve formě textových řetězců či podobných typů transformovat na Java objekty. K tomuto účelu posloužila knihovna OXM[21] ze Spring framework a hlavně Castor XML knihovna[22].

Mapování XML elementů a atributů na Java objekty a jejich atributy je určováno pomocí mapovacích XML souborů. Příklad mapování základního booleovského atributu reprezentovaného elementem BBA:

```
<class name="xquerysearch.domain.result.BBA">
        <map-to xml="DBA" />
        <field name="id" type="string">
                <bind-xml name="id" node="attribute" />
        </field>
```

[21] Knihovna OXM je součástí Spring frameworku a poskytuje prostředky pro mapování Java objektů a XML dat.

[22] Více informací v kapitole 5.1.3

```
6         <field name="transformationDictionary" type="xquerysearch.domain.
              result.TransformationDictionary">
7                 <bind-xml name="TransformationDictionary" node="element" />
8         </field>
9         <field name="dataDictionary" type="xquerysearch.domain.result.
              DataDictionary">
10                <bind-xml name="DataDictionary" node="element" />
11        </field>
12 </class>
```

Mapovat lze XML elementy i atributy, na datové typy, které jsou součástí distribuce Java prostředí i na vlastní objekty. Pro vlastní objekt je třeba uvést způsob mapování XML dat na jeho proměnné či atributy.

Jak je vidět z ukázky kódu, XML data lze mapovat i na vlastní objekty. Z tohoto důvodu jsou všechny základní doménové objekty, které jsou používány při mapování, rozloženy do hierarchické struktury. Mimo dalších vlastností lze v mapovacím souboru snadno určit, zda se jedná o skupinu objektů (List, Set atp.) či o jednotlivé objekty.

Pro každý druh mapování existuje objekt, který má jako jeden z atributů nastaveny mapovací soubory. Mapování je rozděleno do několika souborů, ze kterých se skládá mapování pro jeden hlavní doménový objekt. Toto rozdělení bylo provedeno především z důvodu znovupoužitelnosti kódu.

Pro účely mapování byla vytvořena třída, jejíž metoda pro mapování přijímá dva parametry – objekt mapování (popsaný v předchozím odstavci) a data k mapování jako textový řetězec. Za využití generiky slouží tato třída pro převedení vstupních dat na jakýkoliv zvolený doménový objekt.

5.7 Logování

V aplikace jsou použity dva druhy logování – standardní logování běhu aplikace a vytvořené logování příchozích dotazů a odchozích výsledků vyhledávání.

5.7.1 Informace z běhu aplikace

Standardní logování slouží k uchování informací o běhu aplikace – především je vhodné pro odhalování chyb.

Mimo dalších informací ukládá i výpisy vniklých výjimek při běhu aplikace. Dále je vhodné ukládat informativní výpisy z některých klíčových částí aplikace či výpisy varování vzniklých při nechtěném chování aplikace.

Java jako taková poskytuje základní možnosti logování, nicméně se často využívají knihovny přímo na logování zaměřené. V tomto případě byla použita knihovna log4j. Tato knihovna umožňuje široké možnosti nastavení – například

- z kterých částí aplikace se budou výpisy ukládat,

- cesta k souboru, do kterého budou výpisy ukládány,
- způsob obsluhy logovacího souboru (např. maximální velikost – vytvoření nového souboru, počet uchovávaných souborů, vytváření nového logovacího souboru na denní bázi),
- stupeň ukládaných výpisů (INFO – pouze informativní výpisy – používáno na produkční úrovni, DEBUG – rozsáhlé výpisy vhodné pro vývoj aplikace atp.),
- a další.

Díky použití konfiguračních souborů lze vcelku snadno nastavit chování logování pro vývoj aplikace či pro verzi, která je již nasazena do ostrého provozu.

5.7.2 Dotazy a výsledky vyhledávání

Přímo pro účely aplikace byl vyvinut systém asynchroního logování příchozích dotazů a odchozích výsledků vyhledávání.

Toto logování spočívá v ukládání jednotlivých objektů do samostatných souborů. Pro tyto soubory je vymezena složka, cesta k ní je uvedena v konfiguračním souboru.

Názvy jednotlivých souborů se skládají z tzv. timestamp[23] a textové přípony (součást názvu logovacího souboru), zda se jedná o dotaz či výsledek. Přípona je nastavitelná v konfiguračním souboru. Timestamp zaručuje unikátní označení souboru. Pro soubory jednoho výkonu aplikace (dotaz i výsledek) je použita stejná timestamp pro snazší párování souborů.

Soubory lze ukládat do souborů různých typů. Vzhledem k tomu, že data dotazu i výsledku mají formu XML, byl zvolen xml soubor. Přípona souboru je nastavitelná pomocí konfiguračního souboru.

Práce se soubory může být časově objemná činnost, což by ovšem mělo dopad na rychlost vyhledávání. Z tohoto důvodu bylo toto logování vyjmuto z hlavního vlákna[24], které zpracovává dotaz, a bylo vloženo do samostatného vlákna. Vytváření logovacích souborů proto nezdržuje hlavní proces dotazování, běží paralelně.

Odstraňování starých logovacích souborů

Logovací soubory obsahující dotazy a výsledky vyhledávání snadno dosahují velikosti několika kilobytů. V případě, že by služba byla využívána ve velké míře, popřípadě více službami, docházelo by ke vzniku velkého množství logovacích souborů, které by zabíraly více a více místa. Bylo by tudíž nutné ručně tato data pročišťovat, aby nedošlo k situaci, že na serveru, na kterém je aplikace provozována, dojde k vyčerpání volného místa na disku či discích.

[23] Počet milisekund, které uběhly od 1.1.1970

[24] *„Vlákna představují způsob, jak v rámci jednoho procesu provádět více činností paralelně (nebo – u jednoprocesorového stroje – pseudoparalelně). V rámci každého z vláken je vykonáván kód nezávisle na ostatních vláknech.“*, převzato z Linuxsoft.cz – Java (20) – vlákna – `http://www.linuxsoft.cz/article.php?id_article=1006`

Z toho důvodu je součástí implementace funkcionalita, která za využití Quartz Scheduleru odmazává staré logovací soubory. Pomocí nastavení v konfiguračním souboru lze určit, které soubory jsou považovány za staré. Určení hranice pro vymazání je možné v řádu minut, programově je ovšem omezena na minimálně 5 minut, tzn. že nelze odstranit soubory mladší, než 5 minut.

V případě, že jsou nalezeny soubory starší než zvolená hranice, dojde k jejich odstranění. Součástí nastavení je také tzv. cron expression[25], která určuje, jak často bude kontrola logovacích souborů probíhat.

[25]Cron-expressions se používají k ovládání spouštění vybraných činností. Jedná se o textový řetězec, který má sedm částí, které popisují detaily časování. Tyto části jsou odděleny mezerami a reprezentují sekundy, minuty, hodiny, den v měsíci, měsíc, den v týdnu a volitelně rok. Více na Quartz Enterprise Job Scheduler 1.x Tutorial – Lesson 6: CronTrigger – `http://quartz-scheduler.org/documentation/quartz-1.x/tutorials/TutorialLesson06`

Kapitola 6

Testování

Tato kapitola za zabývá návrhem testovacích scénářů, které mají za úkol ověřit správnou funkčnost upravených a nově přidaných funkcí aplikace – podkapitola 6.1. Druhým úkolem této sekce je navrhnout testovací scénáře pro ověření výkonnosti a stability aplikace – podkapitola 6.2.

Testování bude probíhat na lokálním stroji – notebooku. Testování funkčnosti bude probíhat ručně přes webové administrační rozhraní. Testování výkonnosti bude probíhat pomocí k tomu účelu vytvořené testovací aplikace (rovněž v jazyce Java), vyhledávání bude probíhat pomocí vyhledávacího dotazu ze scénáře č.1.

6.1 Testování správné funkčnosti aplikace

Tato sekce se zaměřuje na otestování funkcionality, která byla v aplikaci upravena či do aplikace nově přidána. Účelem testů je vyzkoušet, zda chování aplikace odpovídá očekáváním. Testují se i nepříznivé scénáře.

Scénář 1 – základní vyhledávání

- **Testovaná vlastnost:** Základní vyhledávání bez použití nastavení vyhledávání ve vyhledávacím dotazu.
- **Očekávané chování:** Vrácení výsledků podle zadání v dotazu.

Scénář 2 – Sdružování podle počtu atributů

- **Testovaná vlastnost:** Vyhledávání se sdružováním podle počtu atributů ve výsledku vyhledávání.
- **Očekávané chování:** Vrácení výsledků podle zadání v dotaze rozřazených podle počtu atributů.

Scénář 3 – Sdružování podle počtu atributů na cedent

- **Testovaná vlastnost:** Vyhledávání se sdružováním podle počtu atributů na cedent ve výsledku vyhledávání.
- **Očekávané chování:** Vrácení výsledků podle zadání v dotaze rozřazených podle počtu atributů na cedent.

Scénář 4 – Sdružování podle atributů

- **Testovaná vlastnost:** Vyhledávání se sdružováním podle atributů ve výsledku vyhledávání.
- **Očekávané chování:** Vrácení výsledků podle zadání v dotaze rozřazených podle atributů.

Scénář 5 – Sdružování podle atributů na cedent

- **Testovaná vlastnost:** Vyhledávání se sdružováním podle atributů na cedent ve výsledku vyhledávání.
- **Očekávané chování:** Vrácení výsledků podle zadání v dotaze rozřazených podle atributů na cedent.

Scénář 6 – Sdružování podle kategorií atributu

- **Testovaná vlastnost:** Vyhledávání se sdružováním výsledků vyhledávání podle kategorií atributu *District*.
- **Očekávané chování:** Vrácení výsledků podle zadání v dotaze rozřazených podle kategorií atributu *District*.

Scénář 7 – Sdružování podle kategorií atributu

- **Testovaná vlastnost:** Vyhledávání se sdružováním výsledků vyhledávání podle kategorií atributu *Sex*.
- **Očekávané chování:** Vrácení výsledků podle zadání v dotaze rozřazených podle kategorií atributu *Sex*.

Scénář 8 – Sdružování bez parametrů

- **Testovaná vlastnost:** Vyhledávání se sdružováním bez specifikace parametrů sdružování.
- **Očekávané chování:** Vrácení výstupu bez výsledků.

Scénář 9 – Sdružování bez cíle

- **Testovaná vlastnost:** Vyhledávání se sdružováním bez specifikace cíle vyhledávání – asociační pravidla / zadání dataminingové úlohy.
- **Očekávané chování:** Vrácení výstupu bez změny oproti uvedení správného cíle.

Scénář 10 – Sdružování – cíl zadání dataminingové úlohy

- **Testovaná vlastnost:** Vyhledávání nastavené stejně jako ve scénáři č.2, cíl vyhledávání zadání dataminingové úlohy.
- **Očekávané chování:** Vrácení výstupu bez změny oproti scénáři č.2.

Scénář 11 – Fuzzy vyhledávání – cíl asociační pravidla

- **Testovaná vlastnost:** Vyhledávání s fuzzy ohodnocením výsledků – cíl vyhledávání asociační pravidla.
- **Očekávané chování:** Vrácení fuzzy ohodnocených výsledků vyhledávání.

Scénář 12 – Fuzzy vyhledávání – cíl zadání dataminingové úlohy

- **Testovaná vlastnost:** Vyhledávání s fuzzy ohodnocením výsledků – cíl vyhledávání zadání dataminingové úlohy.
- **Očekávané chování:** Vrácení fuzzy ohodnocených výsledků vyhledávání – ohodnocení pro zadání dataminingové úlohy.

Scénář 13 – Fuzzy vyhledávání – cíl nezadán

- **Testovaná vlastnost:** Vyhledávání s fuzzy ohodnocením výsledků – cíl vyhledávání nezadán.
- **Očekávané chování:** Vrácení výstupu bez změny oproti scénáři č.11.

Scénář 14 – Vyhledávání se shlukováním – s parametry

- **Testovaná vlastnost:** Vyhledávání se shlukováním, nastavení shlukování parametry.
- **Očekávané chování:** Vrácení výstupu ve shlucích dle parametrů.

Scénář 15 – Vyhledávání se shlukováním – bez parametrů

- **Testovaná vlastnost:** Vyhledávání se shlukováním, bez nastavení shlukování parametry.
- **Očekávané chování:** Vrácení výstupu ve shlucích dle výchozích hodnot.

Scénář 16 – Základní vyhledávání – cíl zadání datamining ové úlohy

- **Testovaná vlastnost:** Základní vyhledávání – cíl vyhledávání zadání dataminingové úlohy.
- **Očekávané chování:** Vrácení výsledků podle zadání v dotaze.

Scénář 17 – Vyhledávání s užitím hybridního dotazu

- **Testovaná vlastnost:** Vyhledávání s hybridním vyhledávacím dotazem – cíl asociační pravidla + cíl zadání dataminingové úlohy.
- **Očekávané chování:** Vrácení výsledků podle zadání v dotaze.

Scénář 18 – Fuzzy vyhledávání – cíl asociační pravidla, kratší dotaz

- **Testovaná vlastnost:** Vyhledávání s fuzzy ohodnocením výsledků – kratší druh vyhledávání.
- **Očekávané chování:** Vrácení fuzzy ohodnocených výsledků podle zadání v dotaze.

Scénář 19 – Shlukování – bez parametrů, kratší dotaz

- **Testovaná vlastnost:** Vyhledávání se shlukováním výsledků – kratší druh vyhledávání.
- **Očekávané chování:** Vrácení výsledků podle zadání v dotaze rozřazených do shluků.

Scénář 20 – Základní vyhledávání – kratší dotaz

- **Testovaná vlastnost:** Základní vyhledávání – kratší druh vyhledávání.
- **Očekávané chování:** Vrácení výsledků podle zadání v dotaze.

6.2 Testování výkonnosti a stability aplikace

Tato podkapitola obsahuje návrhy testování aplikace z pohledu výkonnosti a stability. Účelem těchto testů je ověřit výkonnost aplikace pod zátěží dimenzovanou na předpokládaný rozsah použití. Druhým účelem je ověření stability aplikace pod zvolenou zátěží.

Scénář 21 – Základní vyhledávání – 2 zdroje po 100 dotazech

- **Testovaná vlastnost:** Základní vyhledávání – paralelní dotazování 2 zdrojů – každý pošle 100 dotazů.
- **Očekávané chování:** Odpověď aplikace bez výrazného nárůstu času nutného na zpracování.

Scénář 22 – Základní vyhledávání – 3 zdroje po 100 dotazech

- **Testovaná vlastnost:** Základní vyhledávání – paralelní dotazování 3 zdrojů – každý pošle 100 dotazů.
- **Očekávané chování:** Odpověď aplikace bez výrazného nárůstu času nutného na zpracování.

Scénář 23 – Základní vyhledávání – 2 zdroje po 1000 dotazech

- **Testovaná vlastnost:** Základní vyhledávání – paralelní dotazování 2 zdrojů – každý pošle 1000 dotazů.
- **Očekávané chování:** Odpověď aplikace bez výrazného nárůstu času nutného na zpracování.

Scénář 24 – Základní vyhledávání – 3 zdroje po 1000 dotazech

- **Testovaná vlastnost:** Základní vyhledávání – paralelní dotazování 3 zdrojů – každý pošle 1000 dotazů.
- **Očekávané chování:** Odpověď aplikace bez výrazného nárůstu času nutného na zpracování.

Scénář 25 – Základní vyhledávání – velký počet výsledků

- **Testovaná vlastnost:** Základní vyhledávání – dotaz, podle kterého bude vyhledáno velké množství výsledků.
- **Očekávané chování:** Odpověď aplikace bez výrazného nárůstu času nutného na zpracování.

6.3 Výsledky testování

Výsledky testů potvrdily správnou funkcionalitu změn a vylepšení aplikace i její stabilitu.

Všechny testy pro ověření správně funkčnosti byly úspěšné. Každý test byl proveden třikrát z důvodu ověření neměnnosti výsledků vyhledávání.

Výkonnostní testy ukázaly, že aplikace je stabilní. Bez problémů odolala všem testům, především posílání 1000 dotazů ze tří zdrojů paralelně (každý zdroj poslal 1000 dotazů) ukázalo, že aplikace by měla odolat náporu při ostrém nasazení.

Testovací scénáře 21-24 byly prováděny s omezením výsledků vyhledávání na 100. Časy se lišily v závislosti na počtu zdrojů, které posílaly dotazy. V případě dvou zdrojů se čas vyřízení jednoho vyhledávacího dotazu pohyboval okolo 3 sekund (rozdílu mezi posíláním 100 dotazů a 1000 dotazů je okolo 0,7s). V případě tří zdrojů čas narostl na přibližně 4-5 sekund (rozdíl mezi posíláním 100 a 1000 dotazů je přibližně 1s).

Testovací scénář 25 měl za úkol prověřit stabilitu a čas na zpracování v případě, že není omezen počet výsledků vyhledávání. V případě použitého dotazu to bylo 414 výsledků vyhledávání. V tomto případě došlo k navýšení času potřebného pro odpověď aplikace na přibližně 27 sekund. Je nutné podotknout, že v případě tohoto scénáře byly vraceny velké objemy dat, které zajisté měly vliv na čas odpovědi aplikace.

Výsledky testů, především těch výkonnostních, je nutné uvádět v kontextu počítačové sestavy použité pro testování. Použitým strojem byl notebook (stáří 5 let, oproti původní konfiguraci upgradovaná RAM paměť a pevný disk), 4 GB RAM paměti, dvou-jádrový procesor Intel Core2 Duo T7100 s rychlostí 1,8 GHz, operační systém Windows 7 Professional 64-bit, ostatní parametry nejsou z pohledu testování důležité. Při testování bylo pozorováno maximální vytížení procesoru (systém při testování nebyl zbytečně vytěžován pro dosažení co nejlepších výsledků) – z toho lze usuzovat, že při provozování aplikace na serveru s výkonným procesorem, bude vyhledávání rychlejší, než dokazují zde uvedené testy.

Informace o testovaní jsou uvedeny v příloze č. 1 na str. 102.

Závěr

V úvodu byly stanoveny následující cíle, kterých má být v rámci vypracování diplomové práce dosaženo:

- Provedení úprav aplikace se zaměřením na snadnost budoucích úprav a vylepšení a také odstraňování případných chyb stávající implementace
- Umožnění obecnějšího zadání vyhledávání v databázi úloh DZD
- Zavedení fuzzy přístupů do vyhledávání v databázi úloh DZD
- Umožnění shlukování výsledků vyhledávání

První cíl byl splněn kompletním přepracováním výchozí aplikace XQuery search, čímž vznikla aplikace I:ZI Repository. Z původní aplikace prakticky nezůstal žádný kód, původní funkcionalita byla zachována v rámci nového kódu, který splňuje potřebně vlastnosti pro použití v nové struktuře aplikace.

Nové členění aplikace je uvedeno v podkapitole 4.1. Umožňuje výrazně snadnější zavádění vylepšení aplikace a výhodou je také lepší možnost například výměny úložiště a celé vrstvy pro komunikaci s ním. Nové členění také aplikaci zpřehlednilo a usnadnilo tak hledání případných implementačních chyb.

Přepsání aplikace s sebou také přineslo změnu práce s daty a to především s vyhledávacím dotazem a s výsledky vyhledávání. S těmito dvěma objekty bylo doposud zacházeno jako s nestrukturovanými daty v podobě textového řetězce – fakticky to byla data ve formě XML. To byla výrazná překážka pro vylepšování aplikace. Proto bylo zavedeno mapování XML dat na Java objekty. To s sebou přineslo markantní vylepšení strukturovanosti dat. Díky tomuto mapování lze s daty v aplikaci zacházet jako s Java objekty, což usnadnilo zavádění změn a vylepšení.

Po výše popsaných úpravách aplikace přišla na řadu vylepšení aplikace. Prvním z nich je zobecnění vyhledávacího dotazu. Toho bylo dosaženo úpravou stávajícího formátu vyhledávacího dotazu a především umožnění vyhledávat podle zadání dataminingové úlohy. Aktuálně tedy lze zadávat vyhledávací dotaz na prohledání výsledků dataminingových úloh, na prohledání zadání dataminingových úloh (a vrácení také jejích výsledků) či dotaz hybridní. V hybridním typu dotazu lze specifikovat požadavky na výsledky vyhledávání z pohledu zadání dataminingové úlohy i z pohledu výsledků dataminingové úlohy.

Ze zavedení mapování XML dat na Java objekty těží také další vylepšení aplikace – fuzzy vyhledávání. Tento způsob vyhledávání vrací tzv. fuzzy ohodnocené výsledky. Fuzzy ohod-

nocení se skládá ze dvou druhů ohodnocení – prvním je podobnost výsledku vyhledávání a vyhledávacího dotazu, druhým je ohodnocení vlastností výsledku vyhledávání, které nebyly využity v prvním druhu ohodnocení a jsou relevantní. Toto ohodnocení tudíž určuje, jak moc je výsledek vyhledávání podobný vyhledávacímu dotazu, případně zda neobsahuje vlastnosti snižující jeho relevantnost.

Bez mapování XML dat by se neobešlo ani poslední vylepšení aplikace stanovené v úvodu práce – shlukování výsledků vyhledávání. Tato funkčnost byla pro potřeby práce a aplikace rozdělena na dvě oddělené části – shlukování a sdružování. Sdružování rozřazuje výsledky vyhledávání do skupin podle kritéria, které je vyhledávacího dotazu. Shlukování řadí výsledky vyhledávání do shluků na základě jejich fuzzy ohodnocení. Pro shlukování byla užita metoda centroidů, výsledek vyhledávání je přiřazován do shluku na základě podobnosti výsledku vyhledávání a centroidu. Pro přiřazení do shluku je zvolena hranice minimální podobnosti shluku a výsledku vyhledávání (ta může být součástí vyhledávacího dotazu).

Cílů vytyčených v úvodu práce bylo tedy dosaženo.

Mimo výše popsaných úprav a vylepšení aplikace bylo také vylepšeno logování aplikace. Mimo běžného zaznamenávání činnosti aplikace bylo přidáno logování příchozích vyhledávacích dotazů a výsledků vyhledávání, které aplikace vrací. Toto logování probíhá asynchronně – je vyjmuto z hlavního vlákna průchodu aplikací, aby nedocházelo k zpomalení odpovědi aplikace. Vlastní logovací soubor má každý příchozí vyhledávací dotaz i každá odchozí odpověď aplikace (obsahující nalezené výsledky vyhledávání). Aplikace také umožňuje pravidelné mazání starých logovacích souborů (periodicita mazání i maximální stáří logovacích souborů lze nastavit).

Vypracování této práce mi poskytlo několik velice cenných zkušeností. Hlavní z nich je zpracování obsáhlého tématu a vypracování textu, který o tématu pojednává. Mezi další zkušenosti patří využití technologií, na kterých je aplikace postavena a přepracování již existující aplikace od základu, zachování její funkcionality a přidání funkcionalit nových.

Možnosti vylepšení a práce v budoucnu

Jednou z možností vylepšení je úprava shlukování. Současný způsob určování podobnosti centroidu a výsledku vyhledávání dostatečně nereflektuje fakt, že oba objekty jsou si většinou poměrně dost podobné. Vychází to především ze skutečnosti, že výsledky vyhledávání jsou hodně podobné vyhledávacímu dotazu a většinou jsou si výsledky vyhledávání navzájem podobné. Z tohoto důvodu mají vysokou podobnost mezi sebou i s centroidy jednotlivých shluků. Tento problém by mohl být vyřešen upravením stávajícího způsobu měření podobnosti centroidu a výsledku vyhledávání, případně vyvinutím nového způsobu poměřování podobnosti, který by dostatečně reflektoval skutečnost popsanou v tomto odstavci.

Další možností pro vylepšení aplikace se týká fuzzy vyhledávání. Pro fuzzy hodnocení výsledků vyhledávání je k dispozici několik aspektů. To ovšem neplatí pro fuzzy hodnocení zadání dataminingové úlohy v případě, že je vyhledávání specifikováno právě pro zadání dataminingové úlohy. Tento problém by mohl být vyřešen vymyšlení nových vhodných aspektů pro fuzzy hodnocení zadání dataminingové úlohy.

Další návrh na budoucí práci na aplikaci se již netýká nedostatků dosavadní implementace. Z pohledu struktury aplikace by bylo vhodné oddělit samotné API pro vyhledávání a práci

s XML úložištěm a administrační aplikaci. Takové rozdělení by lépe reflektovalo fakt, že vyvíjená aplikace má být REST API pro práci s XML úložištěm. Oddělení administračního modulu (a také jeho vylepšení a rozšíření) by také mohlo poskytnout prostor pro zavedení zabezpečení a omezení přístupu k ovládání aplikace.

Součástí budoucí práce by také mělo být rozšíření chybových hlášek odesílaných v případě problémů v rámci odpovědi aplikace, stejně tak optimalizace některých operací pro dosažení rychlejšího času potřebného na zpracování vyhledávacího dotazu a vrácení výsledků vyhledávání.

Použitá literatura

[1] AN, A.; KHAN, S.; HUANG, X.: Objective and Subjective Algorithms for Grouping Association Rules. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, Washington, DC, USA: IEEE Computer Society, 2003, ISBN 0-7695-1978-4, s. 477–.
URL `http://dl.acm.org/citation.cfm?id=951949.952077`

[2] BABUŠKA, R.: Fuzzy and neural control. DISC Course Lecture Notes (November 2009). 2009.
URL `http://www.dcsc.tudelft.nl/~disc_fnc/transp/fncontrol.pdf`

[3] BERKA, P.: *Dobývání znalostí z databází*. Academia, 2003, ISBN 9788020010629.
URL `http://books.google.cz/books?id=tGvFAAAACAAJ`

[4] BERKA, P.: *Inteligentní systémy*. Vysoká škola ekonomická v Praze. Fakulta informatiky a statistiky, Nakladatelství Oeconomica, 2008, ISBN 9788024514369.
URL `http://books.google.cz/books?id=OPLucQAACAAJ`

[5] BURDA, M.: Získávání znalostí z databází - Asociační pravidla. 2004.
URL `http://www.fit.vutbr.cz/study/courses/ZZD/public/seminar0304/GUHA-text.pdf`

[6] FAYYAD, U. M.; PIATETSKY-SHAPIRO, G.; SMYTH, P.; aj. (editoři): *Advances in knowledge discovery and data mining*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996, ISBN 0-262-56097-6.

[7] HALL, P. A. V.; DOWLING, G. R.: Approximate String Matching. *ACM Comput. Surv.*, ročník 12, č. 4, 1980: s. 381–402.

[8] HELM, L.: *Fuzzy Association Rules - An Implementation in R*. Diplomová práce, Vienna University of Economics and Business Administration, 2007.

[9] HONG, T.-P.; LEE, Y.-C.: An Overview of Mining Fuzzy Association Rules. In *Fuzzy Sets and Their Extensions: Representation, Aggregation and Models*, *Studies in Fuzziness and Soft Computing*, ročník 220, editace H. BUSTINCE; F. HERRERA; J. MONTERO, Springer Berlin / Heidelberg, 2008, ISBN 978-3-540-73722-3, s. 397–410.
URL `http://dx.doi.org/10.1007/978-3-540-73723-0_20`

[10] KEJKULA, M.: *Zpracování asociačních pravidel metodou vícekriteriálního shlukování*. Dizertační práce, VŠE-FIS, 2009.

[11] KLIR, G.; YUAN, B.: *Fuzzy sets and fuzzy logic: theory and applications*. Prentice Hall PTR, 1995, ISBN 9780131011717.
URL `http://books.google.cz/books?id=AOhQAAAAMAAJ`

[12] LÖSTER, T.: *Hodnocení výsledků metod shlukové analýzy.* Dizertační práce, Vysoká škola ekonomická v Praze. Fakulta informatiky a statistiky, 2011.

[13] NAVARRO, G.: A guided tour to approximate string matching. *ACM Comput. Surv.*, ročník 33, March 2001: s. 31–88, ISSN 0360-0300, doi:http://doi.acm.org/10.1145/375360.375365.
URL `http://doi.acm.org/10.1145/375360.375365`

[14] OLIVEIRA, J.; PEDRYCZ, W.: *Advances in fuzzy clustering and its applications.* Wiley, 2007, ISBN 9780470027608.
URL `http://books.google.cz/books?id=KAzHSQMY5P4C`

[15] RAUCH, J.: Asociační pravidla a matematická logika. In *Znalosti 2004*, editace V. SNÁŠEL, VŠB TU Ostrava, 2004, ISBN 80-248-0456-5, s. 114–125.

[16] RAUCH, J.: *Systém LISp-Miner.* Vysoká škola ekonomická v Praze, fakulta informatiky a statistiky, 2005, stručný popis určený pro posluchače kurzů Metod zpracování informací.

[17] RAUCH, J.; ŠIMŮNEK, M.: An Alternative Approach to Mining Association Rules. In *Foundations of Data Mining and knowledge Discovery*, *Studies in Computational Intelligence*, ročník 6, editace T. Y. Lin; S. Ohsuga; C.-J. Liau; X. Hu; S. Tsumoto, Springer, 2005, ISBN 978-3-540-26257-2, s. 211–231.
URL `http://dblp.uni-trier.de/db/series/sci/sci6.html#RauchS05`

[18] SAHAR, S.: Interestingness via what is not interesting. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '99, New York, NY, USA: ACM, 1999, ISBN 1-58113-143-7, s. 332–336, doi:10.1145/312129.312272.
URL `http://doi.acm.org/10.1145/312129.312272`

[19] STROSSA, P.: *Zpracování informačních fondů: Algoritmizace a automatizace zpracování textovỳch informací.* číslo č. 2 in Zpracování inform. fondů, Vysoká škola ekonomická, 1994, ISBN 9788070791974.
URL `http://books.google.cz/books?id=Wy_eAAAACAAJ`

[20] VEJNAROVÁ, J.; JIRKŮ, P.: *Formální logika - neformální výklad základů formální logiky.* Vysoká škola ekonomická v Praze, 2000, ISBN 80-245-0054-X.

[21] VERNICA, R.; LI, C.: Efficient top-k algorithms for fuzzy search in string collections. In *Proceedings of the First International Workshop on Keyword Search on Structured Data*, KEYS '09, New York, NY, USA: ACM, 2009, ISBN 978-1-60558-570-3, s. 9–14, doi:http://doi.acm.org/10.1145/1557670.1557677.
URL `http://doi.acm.org/10.1145/1557670.1557677`

[22] ZADEH, L.; KLIR, G.; YUAN, B.: *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers.* Advances in fuzzy systems, World Scientific, 1996, ISBN 9789810224219.
URL `http://books.google.cz/books?id=wu0dMiIHwJkC`

[23] ŠIMŮNEK, M.: *Systém LISp-Miner: akademickỳ systém pro dobỳvání znalostí z databází : historie vỳvoje a popis ovládání.* Vysoká škola ekonomická v Praze. Fakulta informatiky a statistiky, Nakladatelství Oeconomica, 2010, ISBN 9788024516998.
URL `http://books.google.cz/books?id=RdjnZwEACAAJ`

Přílohy

Součástí diplomové práce je mimo níže uvedených příloh i přiložené CD. Toto CD obsahuje zdrojové kódy aplikace I:ZI Repository, zdrojové soubory Latex verze diplomové práce a nákresy, které jsou v diplomové práci použity. Struktura přiloženého CD je následující:

```
CD
| - aplikace     // Kompletní zdrojové soubory aplikace I:ZI Repository
| - diagramy     // Diagramy použité v diplomové práci
| - prace-latex  // Zdrojové soubory Latex verze diplomové práce
```

Příloha č. 1 – Výsledky testů

Testy pro ověření funkcionality

Číslo testu	Pokus 1	Pokus 2	Pokus 3
1	OK	OK	OK
2	OK	OK	OK
3	OK	OK	OK
4	OK	OK	OK
5	OK	OK	OK
6	OK	OK	OK
7	OK	OK	OK
8	OK	OK	OK
9	OK	OK	OK
10	OK	OK	OK
11	OK	OK	OK
12	OK	OK	OK
13	OK	OK	OK
14	OK	OK	OK
15	OK	OK	OK
16	OK	OK	OK
17	OK	OK	OK
18	OK	OK	OK
19	OK	OK	OK
20	OK	OK	OK

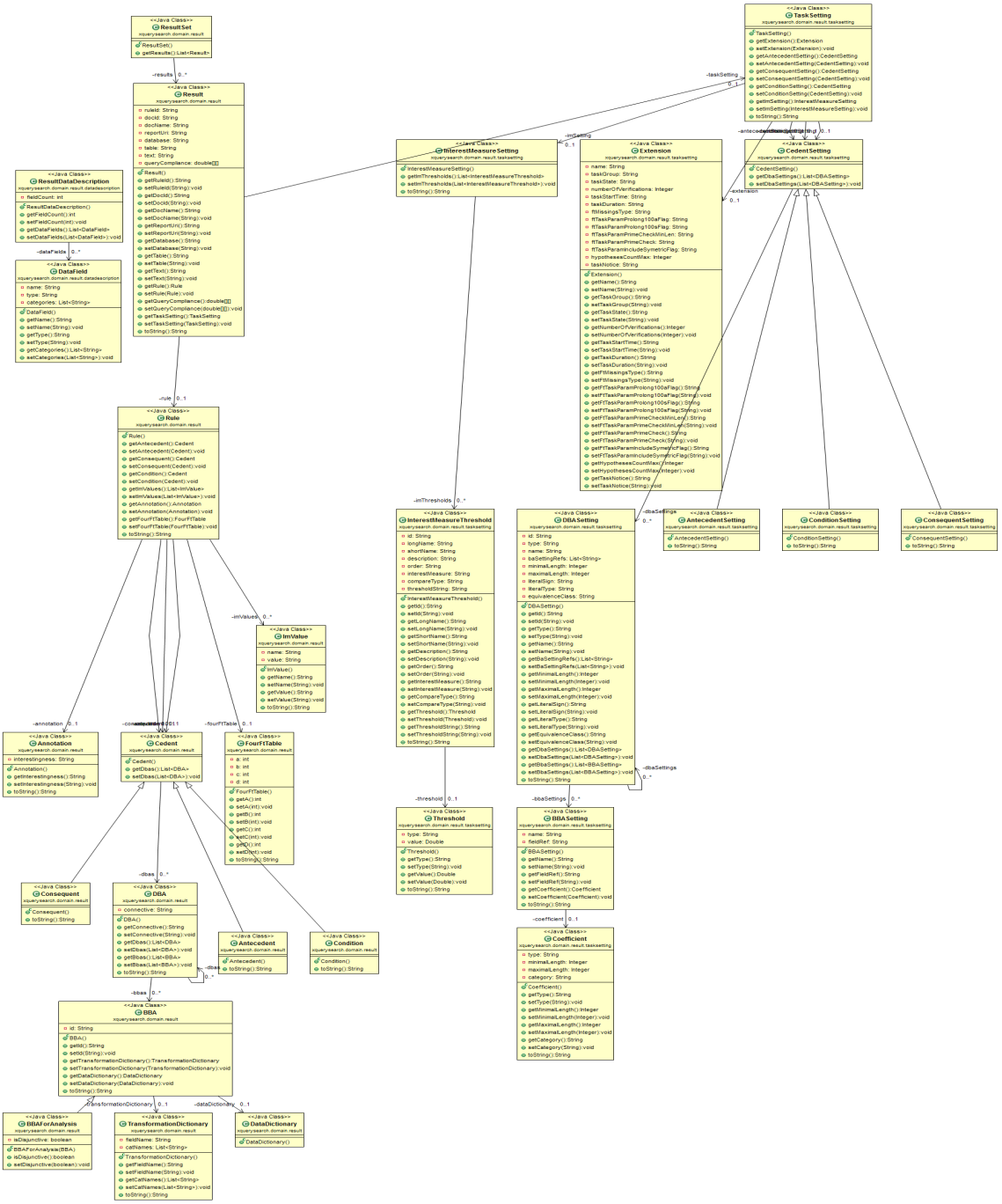
Výsledky testů pro ověření funkcionality

Testy pro ověření výkonnosti a stability

Číslo testu	Pokus 1			Pokus 2			Pokus 3		
	Avg (ms)	Min (ms)	Max (ms)	Avg (ms)	Min (ms)	Max (ms)	Avg (ms)	Min (ms)	Max (ms)
21	3346,5	2707	4435	2741,5	2466	3841	2899,5	2467	5657
22	4122	2621	5965	4032,3	2481	5725	4191,7	2544	6337
23	3246,5	2450	6784	3378,2	2387	7058	3505,3	2507	7003
24	5098,3	2538	12282	5158,7	2308	9534	5305,1	2489	8730
25	26582,7	19906	33160	27002,5	20150	32768	26785,3	19856	32509

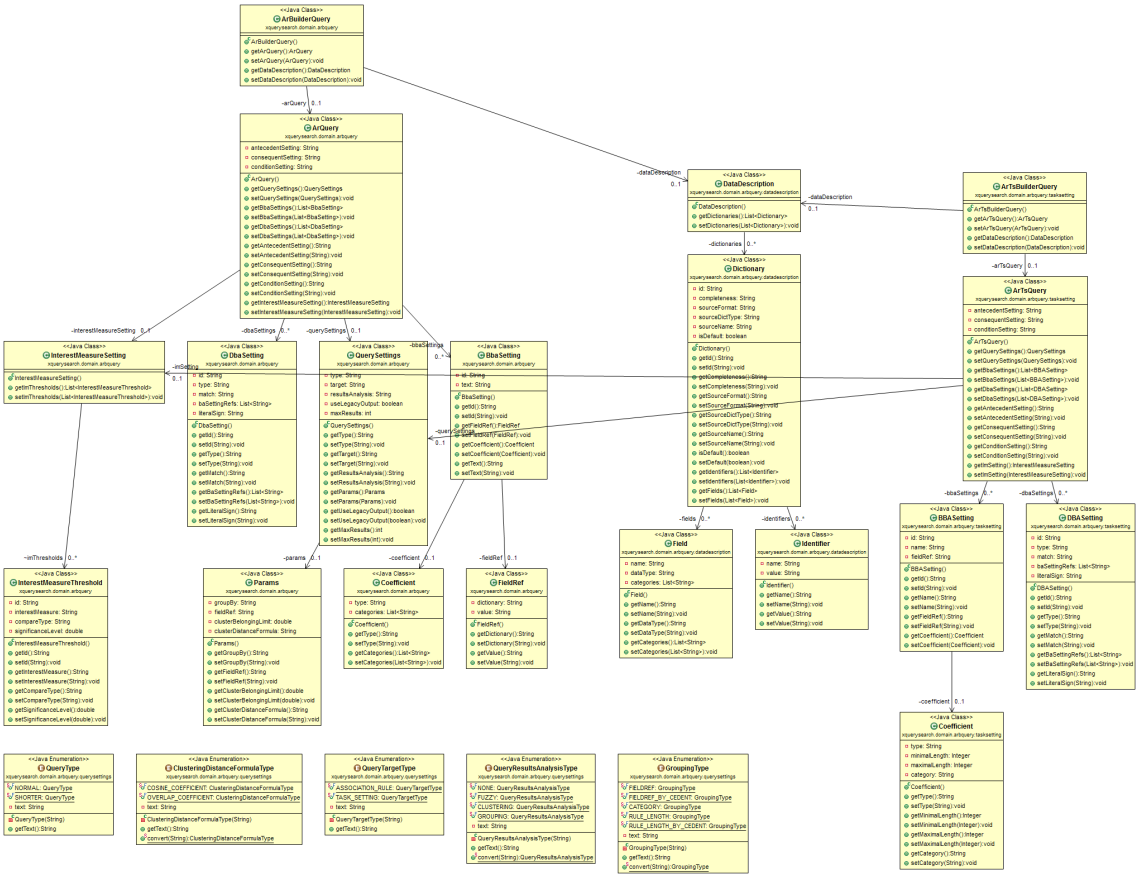
Výsledky testů pro ověření výkonnosti a stability

Příloha č. 2 – Class diagram pro výsledek vyhledávání



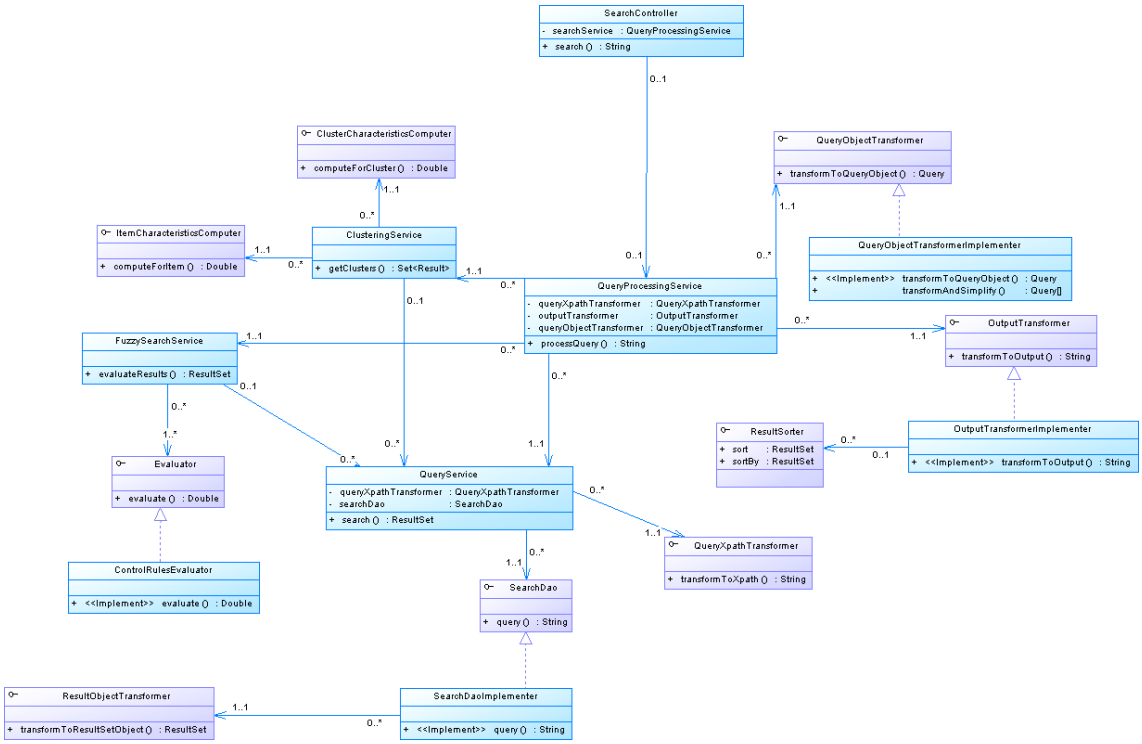
Class diagram pro výsledek vyhledávání – vygenerovaný na základě implementace

Příloha č. 3 – Class diagram pro vyhledávací dotaz



Class diagram pro vyhledávací dotaz – vygenerovaný na základě implementace

Příloha č. 4 – Návrh struktury tříd



Návrh struktury tříd aplikace (pouze pro hlavní funkcionalitu) – class diagram

Příloha č. 5 – Vyhledávací dotaz – prohledávání výsledků dataminingových úloh

```
<?xml version="1.0"?>
<arb:ARBuilder xmlns:arb="http://keg.vse.cz/ns/arbuilder0_2"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://keg.vse.cz/ns/arbuilder0_2 http://sewebar.vse.cz/schemas/
          ARBuilder0_2.xsd">

     <!-- DataDescription zkraceno - zobrazeno pouze nekolik atributu pro ukazku -->
     <DataDescription xmlns:ar="http://keg.vse.cz/ns/arbuilder0_2"
          xmlns:dd="http://keg.vse.cz/ns/datadescription0_2"
          xmlns:guha="http://keg.vse.cz/ns/GUHA0.1rev1">
          <Dictionary completeness="All" sourceFormat="PMML" sourceDictType="
               TransformationDictionary"
               id="2" default="true" sourceName="Loans">
               <Identifier name="dataset">Loans</Identifier>
               <Identifier name="metabase">LMM-VUBEXl1t0k2xnJnXFwuyHg</Identifier>
               <Field dataType="string">
                    <Name>Age</Name>
                    <Category>&lt;21;31)</Category>
                    <Category>&lt;31;41)</Category>
                    <Category>&lt;41;51)</Category>
                    <Category>&lt;51;61)</Category>
                    <Category>&lt;61;71)</Category>
               </Field>
               <Field dataType="string">
                    <Name>Salary</Name>
                    <Category>low</Category>
                    <Category>avg</Category>
                    <Category>high</Category>
               </Field>
               <Field dataType="string">
                    <Name>Sex</Name>
                    <Category>F</Category>
                    <Category>M</Category>
               </Field>
               <Field dataType="string">
                    <Name>Quality</Name>
                    <Category>good</Category>
                    <Category>bad</Category>
               </Field>
          </Dictionary>
     </DataDescription>
     <ARQuery>
               <QuerySettings>
               <Type>Normal</Type> <!-- Shorter, Normal -->
               <LegacyOutput>false</LegacyOutput>
               <MaxResults>10</MaxResults>
               <Target>AssociationRule</Target> <!-- AssociationRule, TaskSettings -->
               <ResultsAnalysis>Grouping</ResultsAnalysis> <!-- None, Fuzzy, Clustering, Grouping --
                    >
               <Params>
                    <GroupBy>RuleLength</GroupBy>
                    <FieldRef>District</FieldRef>
               </Params>
          </QuerySettings>
          <BBASettings>
               <BBASetting id="3">
                    <Text>District</Text>
                    <FieldRef dictionary="TransformationDictionary">District</FieldRef>
                    <Coefficient>
                         <Type>At least one from listed</Type>
                         <Category>Benesov</Category>
                    </Coefficient>
               </BBASetting>
               <BBASetting id="6">
                    <Text>Quality</Text>
                    <FieldRef dictionary="TransformationDictionary">Quality</FieldRef>
                    <Coefficient>
                         <Type>At least one from listed</Type>
                         <Category>good</Category>
                    </Coefficient>
               </BBASetting>
          </BBASettings>
          <DBASettings>
               <DBASetting id="1" type="AnyConnective" match="traverseOnly">
                    <BASettingRef>2</BASettingRef>
               </DBASetting>
               <DBASetting id="2" type="Literal">
                    <BASettingRef>3</BASettingRef>
                    <LiteralSign>Both</LiteralSign>
               </DBASetting>
               <DBASetting id="4" type="AnyConnective" match="traverseOnly">
                    <BASettingRef>5</BASettingRef>
               </DBASetting>
               <DBASetting id="5" type="Literal">
                    <BASettingRef>6</BASettingRef>
                    <LiteralSign>Both</LiteralSign>
               </DBASetting>
          </DBASettings>
          <AntecedentSetting>1</AntecedentSetting>
          <ConsequentSetting>4</ConsequentSetting>
          <InterestMeasureSetting>
               <InterestMeasureThreshold id="1">
                    <InterestMeasure>FUI</InterestMeasure>
```

```
91                      <CompareType>Greater than or equal</CompareType>
92                      <SignificanceLevel>0.7</SignificanceLevel>
93                  </InterestMeasureThreshold>
94              </InterestMeasureSetting>
95          </ARQuery>
96      </arb:ARBuilder>
```

Příloha č. 6 – Vyhledávací dotaz – prohledávání zadání data-miningových úloh

```
<?xml version="1.0"?>
<arb:ARBuilder xmlns:arb="http://keg.vse.cz/ns/arbuilder0_2"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://keg.vse.cz/ns/arbuilder0_2 http://sewebar.vse.cz/schemas/
        ARBuilder0_2.xsd">

    <!-- DataDescription zkraceno - zobrazeno pouze nekolik atributu pro ukazku -->
    <DataDescription xmlns:ar="http://keg.vse.cz/ns/arbuilder0_2"
        xmlns:dd="http://keg.vse.cz/ns/datadescription0_2"
        xmlns:guha="http://keg.vse.cz/ns/GUHA0.1rev1">
        <Dictionary completeness="All" sourceFormat="PMML" sourceDictType="
            TransformationDictionary"
            id="2" default="true" sourceName="Loans">
            <Identifier name="dataset">Loans</Identifier>
            <Identifier name="metabase">LMM-VUBEXll1t0k2xnJnXFwuyHg</Identifier>
            <Field dataType="string">
                <Name>Age</Name>
                <Category>&amp;lt;21;31)</Category>
                <Category>&amp;lt;31;41)</Category>
                <Category>&amp;lt;41;51)</Category>
                <Category>&amp;lt;51;61)</Category>
                <Category>&amp;lt;61;71)</Category>
            </Field>
            <Field dataType="string">
                <Name>Salary</Name>
                <Category>low</Category>
                <Category>avg</Category>
                <Category>high</Category>
            </Field>
            <Field dataType="string">
                <Name>Sex</Name>
                <Category>F</Category>
                <Category>M</Category>
            </Field>
            <Field dataType="string">
                <Name>Quality</Name>
                <Category>good</Category>
                <Category>bad</Category>
            </Field>
        </Dictionary>
    </DataDescription>
    <ARQuery>
        <QuerySettings>
            <Type>Normal</Type> <!-- Shorter, Normal -->
            <Target>TaskSetting</Target> <!-- AssociationRule, TaskSetting -->
            <ResultsAnalysis>None</ResultsAnalysis> <!-- None, Fuzzy, Clustering, Grouping -->
        </QuerySettings>
        <BBASettings>
            <BBASetting id="3">
                <Name>Age (subset), 1 - 1</Name>
                <FieldRef>Age</FieldRef>
                <Coefficient>
                    <Type>Subset</Type>
                    <MinimalLength>1</MinimalLength>
                    <MaximalLength>1</MaximalLength>
                </Coefficient>
            </BBASetting>
            <BBASetting id="6">
                <Name>Quality (subset), 1 - 1</Name>
                <FieldRef>Quality</FieldRef>
                <Coefficient>
                    <Type>Subset</Type>
                    <MinimalLength>1</MinimalLength>
                    <MaximalLength>1</MaximalLength>
                </Coefficient>
            </BBASetting>
        </BBASettings>
        <DBASettings>
            <DBASetting id="1" type="Any" match="traverseOnly">
                <BASettingRef>2</BASettingRef>
            </DBASetting>
            <DBASetting id="2" type="Literal">
                <BASettingRef>3</BASettingRef>
                <LiteralSign>Both</LiteralSign>
            </DBASetting>
            <DBASetting id="4" type="Any" match="traverseOnly">
                <BASettingRef>5</BASettingRef>
            </DBASetting>
            <DBASetting id="5" type="Literal">
                <BASettingRef>6</BASettingRef>
                <LiteralSign>Both</LiteralSign>
            </DBASetting>
        </DBASettings>
        <AntecedentSetting>1</AntecedentSetting>
        <ConsequentSetting>4</ConsequentSetting>
        <InterestMeasureSetting>
            <InterestMeasureThreshold id="1">
                <InterestMeasure>Any Interest Measure</InterestMeasure>
            </InterestMeasureThreshold>
        </InterestMeasureSetting>
    </ARQuery>
</arb:ARBuilder>
```

Příloha č. 7 – Vyhledávací dotaz – hybridní dotaz

```
<?xml version="1.0"?>
<arb:ARBuilder xmlns:arb="http://keg.vse.cz/ns/arbuilder0_2"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://keg.vse.cz/ns/arbuilder0_2 http://sewebar.vse.cz/schemas/
            ARBuilder0_2.xsd">

    <!-- DataDescription zkraceno - zobrazeno pouze nekolik atributu pro ukazku -->
    <DataDescription xmlns:ar="http://keg.vse.cz/ns/arbuilder0_2"
            xmlns:dd="http://keg.vse.cz/ns/datadescription0_2"
            xmlns:guha="http://keg.vse.cz/ns/GUHA0.1rev1">
        <Dictionary completeness="All" sourceFormat="PMML" sourceDictType="
            TransformationDictionary"
                id="2" default="true" sourceName="Loans">
            <Identifier name="dataset">Loans</Identifier>
            <Identifier name="metabase">LMM-VUBEXl1t0k2xnJnXFwuyHg</Identifier>
            <Field dataType="string">
                <Name>Age</Name>
                <Category>&lt;21;31)</Category>
                <Category>&lt;31;41)</Category>
                <Category>&lt;41;51)</Category>
                <Category>&lt;51;61)</Category>
                <Category>&lt;61;71)</Category>
            </Field>
            <Field dataType="string">
                <Name>Sex</Name>
                <Category>F</Category>
                <Category>M</Category>
            </Field>
            <Field dataType="string">
                <Name>Quality</Name>
                <Category>good</Category>
                <Category>bad</Category>
            </Field>
        </Dictionary>
    </DataDescription>
    <HybridQuery>
        <TaskSetting>
            <ARQuery>
                <QuerySettings>
                    <Type>Normal</Type> <!-- Shorter, Normal -->
                </QuerySettings>
                <BBASettings>
                    <BBASetting id="3">
                        <Name>Age (subset), 1 - 1</Name>
                        <FieldRef>Age</FieldRef>
                        <Coefficient>
                            <Type>Subset</Type>
                            <MinimalLength>1</MinimalLength>
                            <MaximalLength>1</MaximalLength>
                        </Coefficient>
                    </BBASetting>
                    <BBASetting id="6">
                        <Name>Quality (subset), 1 - 1</Name>
                        <FieldRef>Quality</FieldRef>
                        <Coefficient>
                            <Type>Subset</Type>
                            <MinimalLength>1</MinimalLength>
                            <MaximalLength>1</MaximalLength>
                        </Coefficient>
                    </BBASetting>
                </BBASettings>
                <DBASettings>
                    <DBASetting id="1" type="Any" match="traverseOnly">
                        <BASettingRef>2</BASettingRef>
                    </DBASetting>
                    <DBASetting id="2" type="Literal">
                        <BASettingRef>3</BASettingRef>
                        <LiteralSign>Both</LiteralSign>
                    </DBASetting>
                    <DBASetting id="4" type="Any" match="traverseOnly">
                        <BASettingRef>5</BASettingRef>
                    </DBASetting>
                    <DBASetting id="5" type="Literal">
                        <BASettingRef>6</BASettingRef>
                        <LiteralSign>Both</LiteralSign>
                    </DBASetting>
                </DBASettings>
                <AntecedentSetting>1</AntecedentSetting>
                <ConsequentSetting>4</ConsequentSetting>
                <InterestMeasureSetting>
                    <InterestMeasureThreshold id="1">
                        <InterestMeasure>Any Interest Measure</InterestMeasure>
                    </InterestMeasureThreshold>
                </InterestMeasureSetting>
            </ARQuery>
        </TaskSetting>
        <AssociationRule>
            <ARQuery>
                <QuerySettings>
                    <Type>Normal</Type> <!-- Shorter, Normal -->
                </QuerySettings>
                <BBASettings>
                    <BBASetting id="3">
                        <Text>Age</Text>
                        <FieldRef dictionary="TransformationDictionary">Age</FieldRef>
                        <Coefficient>
```

```
                                <Type>At least one from listed</Type>
                                <Category>&lt;21;31)</Category>
                                <Category>&lt;31;41)</Category>
                                <Category>&lt;41;51)</Category>
                                <Category>&lt;51;61)</Category>
                                <Category>&lt;61;71)</Category>
                            </Coefficient>
                        </BBASetting>
                        <BBASetting id="6">
                            <Text>Quality</Text>
                            <FieldRef dictionary="TransformationDictionary">Quality</FieldRef>
                            <Coefficient>
                                <Type>At least one from listed</Type>
                                <Category>good</Category>
                            </Coefficient>
                        </BBASetting>
                    </BBASettings>
                    <DBASettings>
                        <DBASetting id="1" type="Any" match="traverseOnly">
                            <BASettingRef>2</BASettingRef>
                        </DBASetting>
                        <DBASetting id="2" type="Literal">
                            <BASettingRef>3</BASettingRef>
                            <LiteralSign>Both</LiteralSign>
                        </DBASetting>
                        <DBASetting id="4" type="Any" match="traverseOnly">
                            <BASettingRef>5</BASettingRef>
                        </DBASetting>
                        <DBASetting id="5" type="Literal">
                            <BASettingRef>6</BASettingRef>
                            <LiteralSign>Both</LiteralSign>
                        </DBASetting>
                    </DBASettings>
                    <AntecedentSetting>1</AntecedentSetting>
                    <ConsequentSetting>4</ConsequentSetting>
                    <InterestMeasureSetting>
                        <InterestMeasureThreshold id="1">
                            <InterestMeasure>Any Interest Measure</InterestMeasure>
                        </InterestMeasureThreshold>
                    </InterestMeasureSetting>
                </ARQuery>
            </AssociationRule>
        </HybridQuery>
    </arb:ARBuilder>
```