Mendelova zemědělská a lesnická univerzita v Brně
Provozně ekonomická fakulta

Fulltextové vyhledávání v rozsáhlém informačním systému

Diplomová práce

Vedoucí práce:
RNDr. Ing. Milan Šorm, Ph.D.

Bc. Miroslav Prachař

Brno 2008

Rád bych poděkoval vedoucímu diplomové práce RNDr. Ing. Milanu Šormovi, Ph.D. za jeho cenné rady, připomínky a nápady. Dále děkuji kolegům z vývojového týmu UIS MZLU v Brně za pomoc a rady při tvorbě této práce.

Prohlašuji, že jsem tuto diplomovou práci vyřešil samostatně s použitím literatury, kterou uvádím v seznamu. Při jejím zpracování jsem vycházel z rad, připomínek, názorů a prací zaměstnanců Ústavu pro informační systém MZLU v Brně. Řešení využívá prostředků Univerzitního informačního systému, který vznikl prací desítek členů vývojového týmu.

V Brně dne 16. května 2008 .....

Abstract

Prachař, M. Fulltext searching in a large information system. Diploma thesis. Brno, 2008.

This diploma thesis deals with possibilities of fulltext searching in the University Information System of Mendel University of Agriculture and Forestry in Brno. It analyses the initial stage of searching and describes the current technologies. The thesis includes the design and implementation of a separate fulltext technology using RDBMS Oracle and other instruments. Also a comparison with other available solutions is presented.

Abstrakt

Prachař, M. Fulltextové vyhledávání v rozsáhlém informačním systému. Diplomová práce. Brno, 2008.

Diplomová práce se zabývá možnostmi nasazení fulltextového vyhledávání v Univerzitním informačním systému Mendelovy zemědělské a lesnické univerzity v Brně. Analyzuje výchozí stav vyhledávání a popisuje současné technologické možnosti. Součástí práce je návrh a implementace vlastní fulltextové technologie s využitím RDBMS Oracle a dalších nástrojů. Práce zahrnuje také srovnání s jinými dostupnými řešeními.

Obsah

1 Úvod a cíl práce

1.1 Úvod do problematiky

Informační systémy se v posledních letech staly nedílnou součástí většiny organizací od malých soukromých provozů až po rozsáhlé státní instituce. Chod takových institucí bez integrovaného a automatizovaného výpočetního systému si dnes lze jen stěží představit. Je to dáno povahou obecně celého hospodářského prostředí, kde jedním z nejdůležitějších faktorů jsou přesné a včasné informace. Význam informací není zanedbatelný ani v oblasti školních aktivit a aktivit ve vzdělávání. Jejich získávání a vyhledávání má usnadňovat právě integrovaný informační systém.

Možnost kvalitního vyhledávání informací patří v současné době mezi základní požadavky uživatelsky přívětivého a moderního prostředí jakéhokoliv elektronického systému, sloužícího k poskytování informačních hodnot svým uživatelům. Dvojnásob to platí pro rozsáhlé a robustní informační systémy s obsáhlými databázemi poskytující komplexní služby. U takových systémů roste riziko špatné orientace při práci s nimi a mohou vznikat problémy se zachováním celkové přehlednosti. Stávají se pak hůře použitelnými a nedosáhne se při jejich nasazení takové efektivity, jaké by se dosáhnout mohlo. V očích potenciálních zákazníků mohou díky tomu ztrácet na atraktivitě a oslabuje se tím jejich konkurenceschopnost, i když třeba jsou vystaveny na kvalitním jádře a mají implementovány funkcionality efektivně řešící úkoly z dané oblasti.

Při nástupu elektronických informačních systémů běžně stačilo k zachování jejich přehlednosti systém logicky rozčlenit do několika oddělených sekcí. Zorientovat se a získat potřebné informace přitom nepředstavovalo žádný velký problém a nebylo tudíž potřeba zvláštních požadavků na funkcionalitu vyhledávání. První větší systémy pro zpracování dat a poskytování informací z 60. let minulého století neobsahovaly ani zlomek funkcí dnešních systémů. S rozvojem informačních technologií, které se od svého nástupu stále rychleji rozvíjí, se vytvořily a stále vytvářejí podmínky pro vznik rozsáhlých informačních systémů, tak jak je dnes známe. Současným trendem je vytváření integrovaných informačních systémů, které pokrývají oblast činnosti celé organizace a splňují jejich strategické cíle.

K usnadnění orientace v dnešních rozsáhlých informačních systémech významně přispívá právě možnost vyhledávání. A platí zde přímá úměra – čím kvalitnější možnosti vyhledávání systém umožňuje, tím snadnější je orientace v něm. I u velmi rozsáhlých systémů se tak může usnadnit jejich používání a přispět k zachování přehlednosti. Systém s kvalitním vyhledáváním je schopen rychle poskytnout uživateli potřebné a správné informace a jeho použití se stává celkově efektivnější.

S nárůstem objemů dat v databázích informačních systémů a také s rozvojem Internetu začala vznikat potřeba nástrojů umožňujících i z takových rozsáhlých dat efektivně a rychle získávat relevantní informace. Objevily se tak fulltextové technologie podporující prohledávání rozsáhlých datových zdrojů a poskytující další možnosti při zpracování textových dokumentů. Moderní fulltextové technologie disponují

množstvím vyspělých funkcí, které poskytují široké možnosti specifikace hledané informace a umožňují získávání přesnějších a relevantnějších výsledků vyhledávání.

Možnost fulltextového vyhledávání by tedy neměla chybět u žádného informačního systému, uchovávajícího nějaká textová data. Takovým systémem je i Univerzitní informační systém vyvíjený na Mendelově zemědělské a lesnické univerzitě v Brně, který představuje komplexní integrovaný systém pro podporu činnosti univerzity. V době psaní této práce je systém dále používán čtyřmi univerzitami v ČR a na Slovensku. Systém je vystavěn nad RDBMS (*Relational Database Management System*) firmy Oracle a uchovává a zpracovává data převážně textové povahy, včetně rozsáhlých záznamů a dokumentů.

Vyhledávání je v systému implementováno od samého počátku jeho vývoje. Pro oblast rozsáhlejších textových dat je dále vhodné nasazení fulltextových technologií, které umožní uživatelům lépe získávat relevantní informace z těchto dat.

1.2 Cíl práce

Tato práce si klade za cíl identifikovat a analyzovat oblasti Univerzitního informačního systému vyvíjeného na MZLU v Brně (dále jen UIS), které si žádají nasazení fulltextové technologie a následně tuto technologii vhodnými nástroji implementovat a nasadit do reálného provozu systému.

V rámci práce bude zapotřebí důkladně poznat aspekty fulltextových technologií na teoretické úrovni, analyzovat dostupné prostředky připadající v úvahu při implementaci, dále pak posoudit jejich vhodnost nasazení z hlediska nejen technického, ale také z hlediska ekonomického. Na základě takové analýzy pak může proběhnout posouzení a volba vhodné technologie a poté samotná implementace. Závěr práce bude věnovaný zhodnocení navrženého řešení, porovnání s ostatními možnými řešeními, posouzení přínosu po zavedení do provozu a také diskuzi o možných dalších rozšířeních implementovaného řešení.

2 Základní pojmy a principy

2.1 Data

Pojem data bývá používán velmi často a to v různých kontextech a významech. Obecně bychom mohli říci, že jsou to reprezentanti údajů a hodnot v určitém dohodnutém tvaru, který je vhodný pro zpracování, komunikaci a interpretaci lidskými bytostmi i automatizovanými výpočetními prostředky. Data jsou vyjádřena fyzickým nosičem, ať už jde o inkoust a papír, elektrické signály, či elektromagnetické záření [15]. Pro účely této práce budeme pod pojmem data rozumět záznamy a dokumenty (zejména textové) uložené v databázi UIS.

2.2 Informace

Informace je obsah zprávy definovaný jako záporný dvojkový logaritmus její pravděpodobnosti a vyjadřuje se v bitech. Ve své podstatě jde o velikost snížení entropie (neznalosti) příjemce zprávy [17]. Člověk informacím přisuzuje konkrétní význam a příjem informací uspokojuje jednu z významných potřeb člověka, a to potřebu poznávání nebo také informační potřebu. Informace vznikají z dat až v okamžiku jejich užití. Informace tak představuje vypovídací schopnost dat, vzniká zpracováním dat a je cílem tohoto zpracování [13].

2.3 Indexování

Pod pojmem indexování si představme proceduru, která zajistí podmínky pro realizaci efektivního vyhledávání. Pro efektivní vyhledávání nad množinou dat je potřeba data určitým způsobem předpřipravit a přetransformovat do vhodné podoby. Vzniká tak požadavek na vytvoření speciálních datových struktur (tzv. *indexních struktur*) nad vlastními daty.

Indexování tedy budeme chápat jako proces vytvoření a naplnění indexní datové struktury takovým způsobem, aby bylo maximálně efektivně využito vlastností databázových indexů při vyhledávání [12].

2.4 Databázový index

Databázový index je standardní databázový objekt dostupný ve většině moderních SŘBD, je uložen fyzicky na disku jako jiné databázové objekty. Jeho primárním úkolem je urychlení některých prováděných operací, jako například získávání záznamů z tabulky nebo spojování dvou nebo více tabulek ve složitějších SQL dotazech. Obsahuje určitým způsobem setříděný seznam hodnot a u každé hodnoty uchovává odkaz na místo, kde se daný záznam skutečně nachází (zpravidla řádek tabulky). Lze jej definovat nad jedním nebo více sloupci databázové tabulky.

Implementace indexu je záležitostí konkrétního databázového systému. Moderní databázové systémy však nejčastěji používají stromovou strukturu indexu, zejména

pak jeho tzv. *B-Tree* variantu. Poněkud zjednodušeně můžeme říci, že stromová struktura indexu redukuje časovou složitost elementární vyhledávací operace z lineární (bez použití indexu) přibližně na logaritmickou (s použitím indexu). Podrobněji se touto problematikou zabývá např. [12]. Indexy se stromovou strukturou můžou používat různé ukládací techniky, podle nichž rozlišujeme „normální“ index, index s reverzními klíči, nebo tzv. function based index. Vedle indexů se stromovou strukturou existují i další typy indexů využívajících jinou strukturu než stromovu, např. bitmapový index nebo doménový index.

V databázovém systému Oracle lze dále definovat i kompozitní indexy. Jedná se o velmi užitečný typ indexu, u kterého lze do jednoho indexu zahrnout („vnořit“) více sloupců v definovaném pořadí. Více o indexech pojednává dokumentace systému Oracle [23].

2.4.1 Vlastnosti indexů

Pro účely této práce nás budou nejvíce zajímat účinky indexů v relačním databázovém systému. Vhodně vytvořený index může přinést značné urychlení při získávání výsledků SQL příkazů, zejména těch dotazovacích. Pokud budou hodnoty ve sloupci tabulky dostatečně selektivní[1], tak vyhledání řádku tabulky aplikováním podmínky na tento sloupec bude při použití indexu mnohem rychlejší. Jestli se index skutečně pro vyhledání daných záznamů v tabulce použije, závisí i na dalších faktorech, jimiž jsou zejména charakter a objem konkrétních dat a rozhodnutí optimalizátoru databázového systému, který posuzuje více hledisek pro rozhodnutí postupu při získávání dat[2].

Indexy ovšem mohou způsobit i výkonnostní ztrátu, pokud jsou nevhodně založeny. Údržba indexu totiž stojí systém nějakou režii a může tudíž prodlužovat dobu potřebnou pro modifikaci dat DML příkazy. Zpravidla toto prodloužení bývá zanedbatelné, nicméně v případě např. většího množství (nadbytečně) vytvořených indexů nad tabulkou se zpomalení operace již může viditelně projevit.

Při vytváření indexů je proto potřeba postupovat vždy uvážlivě. Snahou by mělo být zakládat spíše méně indexů, ale zato je cíleně a efektivně využívat. Doporučuje se nezakládat indexy nad malými tabulkami [7] a nad sloupci s malou selektivní schopností (typicky například sloupec určující pohlaví osoby nebo sloupec s hodnotami typu boolean).

[1] Literatura, např. [7], uvádí, že pokud příkaz z dané tabulky vybírá méně než cca 10 % všech záznamů, bude velmi pravděpodobně jejich vyhledání nejrychlejší právě přes index.

[2] Např. v databázovém systému Oracle je implementován tzv. *cost based* optimalizátor, který posuzuje celkové náklady – čas procesoru, počet bloků čtených z disku, celkový čas zpracování atd. – pro jednotlivé možnosti zpracování dotazu a vybírá tu nejmíň nákladnou. Je efektivnější než jeho předchůdce (tzv. *rule based* optimalizátor), který se rozhodoval jen na základě předdefinovaných pravidel. Více o práci optimalizátoru pojednává [6] a [23].

2.5 Inverzní index

Fulltextové vyhledávání pro svoji uspokojivou činnost zpravidla potřebuje speciální druh indexu. Jedná se o tzv. *inverzní indexy*, jejichž konstrukce je postavena na odlišné filozofii než klasické indexy. U těchto indexů je hlavní myšlenkou extrahovat ze zdrojových dokumentů jisté fragmenty textu zvané *tokeny* a k těmto fragmentům pak přiřadit seznam dokumentů, v nichž se vyskytují.

2.6 Fulltextové vyhledávání

Fulltextové vyhledávání nebo jen krátce fulltext bychom mohli definovat jako metodu vyhledávání uvnitř zpravidla rozsáhlých textových souborů nebo v sadě takových souborů. Hybnou silou vzniku a vývoje fulltextu byl nárůst objemu dat publikovaných prostřednictvím sítě internet, kde vznikla potřeba efektivně hledat nebo lokalizovat webové stránky. V první fázi toto řešily tzv. katalogy, kam uživatelé ukládali informace o svých stránkách. Jelikož ale tyto katalogy přestaly požadavkům uživatelů dostačovat, vznikly fulltexty. Softwarové stroje procházely jednotlivé HTML stránky, stahovaly je a transformovaly je do vhodné podoby tak, aby bylo umožněno efektivní vyhledávání v obsazích stránek.

V dnešní době se fulltext uplatňuje všude tam, kde se vyskytují rozsáhlejší textové dokumenty. Některé údaje o dokumentu (např. název, autor nebo třeba klíčová slova) jsou obvykle uvedeny jednak uvnitř samotného dokumentu, ale také i vně dokumentu jako pomocné informace o dokumentu sloužící k jeho vyhledání. Nazývejme tyto pomocné informace jako metadata. Fulltext umožňuje vyhledávat dokumenty nejenom podle jejich metadat, což v mnoha případech nedostačuje, ale prochází samotný obsah dokumentu a porovnává jej se zadaným výrazem.

Textové soubory v nějaké dokumentové sadě obsahují texty týkající se nejrůznějších témat a cílem fulltextu je zprostředkovat nalezení takového textu nebo dokumentu, jehož obsah co nejvíce koresponduje se zadaným vyhledávacím dotazem, tedy hledaným slovem nebo slovním spojením.

2.7 Možnosti získávání informace z textových databází

Pro vyhledávání dokumentů je potřeba stanovit jakým způsobem bude reprezentován dotaz, jak bude reprezentován dokument a jeho index a stanovit pravidla, kdy dokument dotazu vyhoví, případně jak moc je dokument pro dotaz relevantní. Myšlenkové principy, na nichž jsou postaveny vyhledávací stroje lze popsat pomocí modelů. Modelů existuje několik a významně modifikují implementaci vyhledávacích mechanismů [10]. Zmiňme tedy z našeho pohledu ty nejzajímavější z nich. Jedná se zejména o *boolský model*, *vektorový model* a *rozšířený boolský model* vyhledávání[3].

[3]Mezi další modely patří např. pravděpodobnostní model, fuzzy model, neuronový model, latentní sémantický model, min-max model (MMM), Paice model nebo různé modifikace Beyesovských sítí.

V současnosti je tendence, aby daný fulltextový vyhledávač nepracoval výhradně podle jednoho z modelů, ale naopak je snahou hledat mezi jednotlivými modely fúze tak, aby se potlačila negativa a využily výhodné vlastnosti modelů. Zaměřme se nyní na zmiňované tři základní modely, tak jsou popsány v [1], [10], [4] a [3].

2.7.1 Boolský model

Tento model je jeden z nejstarších vůbec (50. léta 20. stol) a v minulosti býval hojně používán v knihovnických informačních systémech. Na dotaz vrací jako výsledky ty dokumenty, které obsahují slova z dotazu. V základní variantě neumožňuje stanovování relevance, pracuje se zde pouze s hodnotami 0 (dokument nevyhovuje dotazu) nebo 1 (dokument vyhovuje dotazu) [10]. Dotazy jde u tohoto modelu vyjádřit logickými výrazy:

- t_a **AND** t_b – v dokumentu se vyskytují závoveň oba termy
- t_a **OR** t_b – v dokumentu se vyskytuje alespoň jeden z termů
- **NOT** t – v dokument se daný term nevyskytuje

Fulltextový dotaz tak může mít například tyto podoby:

„univerzitní AND systém“,
„informace OR data“,
„(univerzitní OR informační) AND systém“
„univerzitní AND NOT informační“.

Boolský model může být obohacen o další prvky, například o zástupné znaky v termech – „univerzit*“ zastupuje termy „univerzita“, „univerzitní“ atd. Dále sem patří různé proximitivní omezení, nebo-li specifikace vzájemné polohy dvou termů v dokumentu – ve stanoveném pořadí za sebou nebo blízko sebe. V neposlední řadě lze do fulltextového dotazu zakomponovat také omezení dle atributů dokumentu – faktografických údajů, typicky např. podmínka „AND rok_vydání >= 2000“.

Výhodou tohoto modelu je bezesporu jeho jednoduchost, nevýhodou pak potenciální existence takových dotazů, kterým vyhoví buďto značně malá, nebo naopak rozsáhlá množina dokumentů. To je důsledkem příliš hrubého rozdělení dokumentů, u nichž se rozlišuje pouze zda dotazu vyhoví nebo nevyhoví. Model nedisponuje nástroji pro uspořádání vyhovujících dokumentů podle míry relevance vůči položenému dotazu. Omezením tohoto modelu je i skutečnost, že všechny termy v dotazu i v identifikaci dokumentu jsou chápány jako stejně důležité.

2.7.2 Vektorový model

Vektorový model je cca o 20 let mladší než boolský a snaží se odstranit nebo alespoň minimalizovat nevýhody boolského modelu. Umožňuje především jemnější výpočet relevance dokumentu vzhledem k dotazu. Myšlenka, ze které model vychází, je opět

velice jednoduchá. Je založena na tom, že každý text (textový záznam či dotaz) je reprezentován bodem v n-rozměrném souřadnicovém systému. Tento bod představuje zároveň i vektor začínající v počátku souřadnic [10].

Vektorový model tedy chápe dotaz i dokument jako vektor $(w_1, \ldots, w_m)$. Číslo m představuje počet všech termů vyskytujících se v sadě dokumentů. Hodnoty $w_1, \ldots, w_m$ jsou z intervalu $\langle 0, 1\rangle$. Složka w_i vyjadřuje významnost i-tého termu pro dokument či dotaz. Nulová nebo nule blízká hodnota znamená term nevýznamný nebo málo významný. Složky blízké jedné pak vyjadřují důležité termy [1].

Vzhledem ke skutečnosti, že dotaz i dokument jsou reprezentovány pomocí vektorů v prostoru $\langle 0, 1\rangle^m$, nabízí se měřit relevanci dokumentu pro daný dotaz jako podobnost příslušných dvou vektorů.

Základní úvaha vychází ze známého faktu, že skalární součin dvou vektorů je největší, pokud mají tyto vektory stejný směr a jako nulový vychází, pokud mají vektory směr opačný. Tohoto jevu lze využít pro vlastní kalkulaci podobnosti. Funkce, která dotazu a dokumentu přiřadí hodnotu jejich podobnosti, se nazývá *podobnostní funkce*.

Nechť q je dotaz reprezentovaný vektorem termů, q_j je j-tá složka vektoru dotazu, d_i je i-tý dokument a $w_{i,j}$ je j-tá složka jeho vektoru (tj. významnost j-tého termu v i-tém dokumentu). Podobnostní funkci pak označme jako $sim(q, d_i)$ [1]. V případě využití vlastností zmiňovaného skalárního součinu by podobnostní funkce měla následující tvar [1]:

$$sim(\vec{q}, \vec{d_i}) = \sum_{j=1}^{m} q_j \cdot w_{i,j}.$$

Uživatelské dotazy jsou obvykle reprezentovány množinou termů, přičemž počet termů je typicky velmi malé číslo. Vektor dotazu tak má téměř všechny složky nulové, pouze pro uvedené termy jsou příslušné složky rovny jedné. Tento přístup lze zobecnit tím, že pro každý term v dotazu bude umožněno uvést jeho váhu [1].

Pro efektivní kalkulaci vah složek vektorů dokumentů v kolekci lze dle [1] vektory spočítat jako:

$$w_{i,j} = TF_{i,j} \cdot IDF_j,$$

kde $w_{i,j}$ je opět j-tá složka vektoru i-tého dokumentu. $TF_{i,j}$ přestavuje frekvenci (počet výskytů) j-tého termu v i-tém dokumentu a IDF_j je tzv. *inverzní frekvence* j-tého termu v sadě dokumentů. Inverzní frekvence termu dle [1] vychází ze vztahu

$$IDF_j = \log \tfrac{n}{DF_j}.$$

Hodnota DF_j vyjadřuje počet dokumentů obsahujících j-tý term, n je pak počet všech dokumentů v kolekci. Inverzní frekvence reprezentuje důležitost termu pro indexaci v rámci celé kolekce dokumentů. Jejím hlavním smyslem je posílit vliv málo používaných termů. U těchto termů se dá totiž předpokládat, že mohou vystihnout charakter dokumentu [1].

Existuje řada možností, jak vektorový model dále modifikovat či rozšiřovat. Jednou z vlastností skalárního součinu například je, že pro dva dlouhé vektory (přiřazené delším dokumentům) vychází jeho hodnota větší než pro dva kratší vektory, přestože v obou případech je směr vektorů shodný. Bezdůvodně by tak byly při výpočtu podobnosti zvýhodněny dlouhé vektory (dokumenty) před krátkými. Pro eliminaci tohoto jevu je proto vhodné vektory normalizovat na jednotkovou délku. Dále podobnostní funkce může mít i komplikovanější tvar než prostý skalární součin. Může být využita např. Kosinová míra, Jaccardova míra, Diceova míra a další, více viz [4].

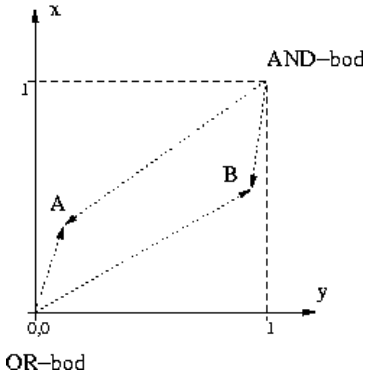
Vektorový model nabízí oproti boolskému modelu jemnější rozlišení míry relevance dokumentů. Ta nabývá hodnota zpravidla z intervalu $\langle 0, 1\rangle$. Čím vyšší číslo je dokumentu přiřazeno, tím více je dokument relevantní vůči vyhledávacímu dotazu.

2.7.3 Rozšířený boolský model

Tento model byl poprvé představen už v roce 1983 (Salton, Fox, Wo) a jeho hlavním cílem bylo přidat do klasického boolského modelu jemnější funkci podobnosti dotazů a dokumentů [11]. Rozšířený boolský model si klade za cíl umožnit při dotazech s typicky boolskými operátory (AND, OR) práci nejen s hodnotami relevance true a false (resp. 1 a 0) jako v případě boolského modelu. Po vyčíslení relevance dokumentu vůči dotazu tak může být výsledek nikoliv jen jedna z hodnot 0 nebo 1, ale může to být i nenulová kladná hodnota z intervalu $\langle 0, 1\rangle$.

Opět vyjdeme z faktu, že dokument d_j je reprezentován jako vektor se složkami $(w_1, \ldots, w_m)$. Pro jednoduchost si představme, že nepracujeme s mnoha slovy (termy), ale jen se dvěma ($m = 2$). Díky tomu veškeré vektory můžeme uvažovat ve dvourozměrném prostoru a lze je tak názorně graficky vyjádřit.

Představme si nyní dvě možnosti. Máme dotaz typu „x AND y“ a dotaz „x OR y“, kde x a y jsou naše jediné dva termy. Uvažujeme-li dotaz s AND, pak je pro nás lepší dokument, který je blíže bodu (1,1) („AND-bod“ z obr. 1). V případě OR dotazu je lepší ten dokument, který je dále od bodu (0,0) [11].



Obr. 1: Princip Rozšířeného boolského modelu

Předpokládejme, že dokument d (respektive jeho vektor) má souřadnice (x, y) a je tedy reprezentován bodem v dvourozměrném prostoru. Potom podobnost dokumentu a dotazu typu „x OR y“ vyhodnocujeme jako vzdálenost bodu d od OR-bodu. Aby tato podobnost vycházela vždy mezi hodnotami 0 až 1, nepoužijeme pro výpočet jen prostou Pythagorovu větu, ale provedeme i potřebnou normalizaci vzdálenosti faktorem $\sqrt{2}$. Dostáváme tak tento tvar podobnostní funkce podle [11]:

$$sim(x\ OR\ y, d) = \sqrt{\frac{x^2+y^2}{2}}.$$

V případě dotazu „x AND y“ postupujeme obdobně, ale vzdálenost, která nás zajímá je měřena od AND-bodu:

$$sim(x\ AND\ y, d) = 1 - \sqrt{\frac{(1-x)^2+(1-y)^2}{2}}$$

V případě, že hodnoty x i hodnoty y nabývají pouze boolských hodnot (tj. 0 nebo 1), pak podobnosti mohou nabývat tří hodnot.

Model jde dále rozvinout, a sice zakomponováním tzv. *p-normy*. V p-normě se namísto druhých mocnin a odmocnin používají jejich ekvivalenty o základu p. Tento parametr p pak musí být nějakým způsobem stanoven před samotnou kalkulací podobnosti. Nabízí se buďto uživatelská volba parametru nebo jeho pevné nastavení v systému. Tvar podobnostní funkce bychom po zavedení p-normy modifikovali takto:

$$sim(x\ OR\ y, d) = \sqrt[p]{\frac{x^p+y^p}{2}}.$$

resp.

$$sim(x\ AND\ y, d) = 1 - \sqrt[p]{\frac{(1-x)^p+(1-y)^p}{2}}$$

Doposud jsme přepokládali existenci pouze dvou termů v dotazu i v dokumentech. Tuto značně omezující podmínku nyní opusťme a předpokládejme existenci obecně m termů. Na základě tohoto předpokladu dostaneme dle [3] zobecněný tvar podobnostní funkce:

$$sim(OR, d) = \sqrt[p]{\frac{\sum_{j=1}^{m} w_{i,j}^p}{m}}.$$

resp.

$$sim(AND, d) = 1 - \sqrt[p]{\frac{\sum_{j=1}^{m} (1-w_{i,j})^p}{m}}$$

Relevance dokumentu se tak odvozuje v případě dotazu typu OR ze vzdálenosti od nulového dokumentu $\vec{d_F} = (0, 0, \ldots, 0)$ a v případě dotazu typu AND ze vzdálenosti od jedničkového dokumentu $\vec{d_F} = (1, 1, \ldots, 1)$

Připusťme ještě možnost specifikovat i v dotazu pro každý výskyt termu t_j také jeho váhu $q_j \in \langle 0, 1 \rangle$. Dotaz takového typu by mohl mít např. podobu „informační(0,8) AND systém“. Pokud se v dotazu u termu váha explicitně neuvede, bere se jako rovna jedné (doposud jsme uvažovali u všech termů v dotazu vždy váhu rovnu jedné). Zakomponováním možnosti stanovení váhy termu v dotazu dostaneme nejobecnější tvar podobnostní funkce [3]:

$$sim(OR, d) = \sqrt[p]{\frac{\sum_{j=1}^{m} q_j^p \cdot w_{i,j}^p}{\sum_{j=1}^{m} q_j}}.$$

resp.

$$sim(AND, d) = 1 - \sqrt[p]{\frac{\sum_{j=1}^{m} q_j^p \cdot (1 - w_{i,j})^p}{\sum_{j=1}^{m} q_j}}$$

V případě kombinovaných dotazů typu „(x AND y) OR z“ se při vyčíslování podobnosti aplikuje postupné substituční dosazování. V tomto případě tedy dojde nejprve k vyčíslení závorky formulí pro AND a tato hodnota se pak bude uvažovat v OR formuli.

Rozšířený boolský model se snaží o skloubení dvou výše popisovaných modelů se zachováním výhod z obou modelů. Z boolského modelu si zachovává možnost specifikace logických vazeb mezi termy v dotazu, z vektorového modelu přidává možnost určení míry relevance dokumentu vzhledem k položenému dotazu.

Parametr p může nabývat v podstatě libovolné kladné reálné hodnoty. Bude-li $p = 1$, pak tyto formule degradují na formule prostého vektorového modelu. V tom případě bude dokonce irelevantní, zda se kalkuluje AND nebo OR varianta dotazu, protože výsledek by byl v obou případech stejný. Naopak v případě, kdy se p blíží nekonečnu ($p \to \infty$), přechází formule na modely blízké klasickému boolskému modelu. Literatura [4] uvádí, že pro $p = 2$ model vykazuje lepší výsledky než model vektorový.

3 Analýza vyhledávání v UIS

3.1 Možnosti vyhledávání v UIS

Funkcionalita vyhledávání je v UIS implementována od samého počátku jeho vývoje. Původně se jednalo o jednoduchou funkcionalitu dohledávání zásadních objektů vyskytujících se v UIS, tedy např. uživatelů, předmětů nebo pracovišť. Vhodným indexováním se dosáhlo možnosti pro každý objekt stanovit množinu údajů a z nich pak klíčů, které po zadání vedou k vrácení objektu ve výsledkové sadě.

Tato funkcionalita tvořila efektivnější alternativu k rozbalovacím seznamům, které díky relativní rozsáhlosti dat nelze dost dobře použít. Práce s rozbalovacími seznamy (nebo-li s tzv. *popupy*) je totiž v případě řádově stovek až tisíců položek značně neefektivní. Nelézt konkrétní položku (např. jméno uživatele) v tak rozsáhlém seznamu je téměř nadlidský úkol a pro práci s informačním systémem by tedy popupy byly v případě těchto objektů nepoužitelné.

Postupným vývojem se popisovaná funkcionalita rozšiřovala na další oblasti vznikající v UIS (publikace, dokumentový server, e-přihlášky atd.) a stala se jádrem pro vyhledávací a portálové aplikace jako Katalog předmětů, Souborný katalog knihovny, Vyhledávání v DS, Vyhledávání lidí, Tematické vyhledávání, a další. Výborně se také v systému uplatní u nově implementované funkcionality tzv. našeptávání[4] založené na technologii AJAX.

3.2 Současný mechanismus indexování

Indexování, které toto rychlé vyhledávání umožňuje, je řešeno na databázové vrstvě za použití PL/SQL procedur a funkcí zapouzdřených do PL/SQL balíku `PKG_INDEX`. Při procesu indexování je potřeba nejprve data převést do výsledného vyhledávaného formátu, což v případě UIS představují řetězce velkých písmen bez diakritiky. Dále je potřeba ukládat všechny možné klíče, které po zadání uživatelem mají vést k nalezení indexované hodnoty (záznamu). Jedině tímto způsobem je možné dosáhnout pozdějšího vyhledávání pomocí operátoru „=“, který umožňuje nejefektivnější použití databázového indexu [5].

Pro názornost si představme, že chceme najít osobu, jejíž příjmení je „Novák“. Aby dohledání bylo efektivní, snadné a uživatelsky přívětivé, je potřeba umožnit osobu nalézt nejenom přesným zadáním řetězce „Novák“, ale i zadáním nějakého podřetězce vytvořeného z výsledného hledaného řetězce. Dále je potřeba umožnit zadávat vzorek bez ohledu na diakritiku a velikost písmen. Pro smysluplné využití indexování bylo v UIS dále stanoveno, že délka takového podřetězce musí být minimálně tři znaky. Při indexování například všech osob s příjmením „Novák“ se proto musí zajistit uložení všech tří a víceznakových podřetězců, které lze z tohoto

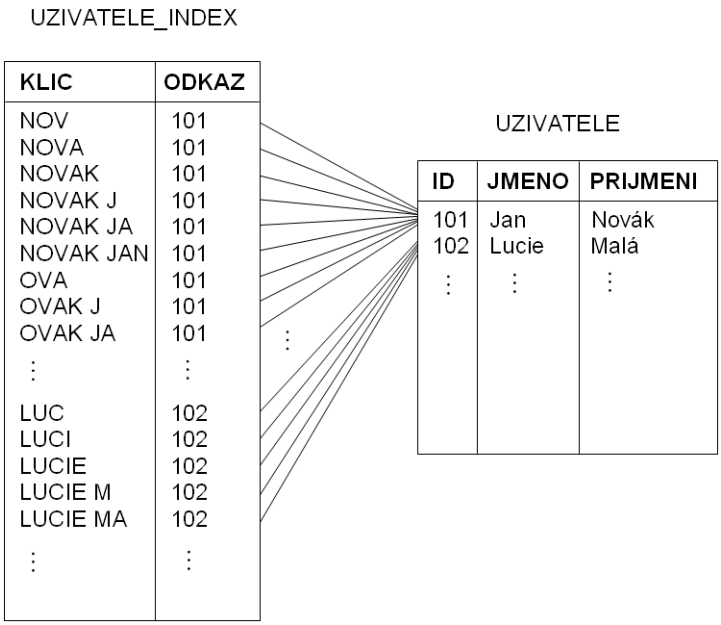
[4]Našeptáváním rozumíme rychlé zobrazení možných vyhovujících záznamů po zadání již prvních několika znaků hledaného řetězce a bez nutnosti opakovaného znovunačítání stránky stiskem tlačítka.

příjmení vytvořit. Z příjmení „Novák“ tedy obdržíme sadu těchto řetězců ve tvaru: NOVAK, NOVA, NOV, OVAK, OVA, a VAK.

Na aplikační úrovni informačního systému je pak zajištěna konverze vyhledávaného řetězce zadaného uživatelem do formátu, který je použit pro ukládání klíčů v databázi. Uživatel tedy může řetězec resp. podřetězec zadávat v různých formátech – s diakritikou i bez ní, může použít malá písmena nebo velká. V každém případě je zajištěna možnost porovnávat zadaný vzorek operátorem „=“ přímo s hodnotami v databázi.

3.2.1 Indexní struktury

Indexní struktura, do které jsou takto vzniklé klíče ukládány, je ve své podstatě velmi jednoduchá. Je tvořena databázovou tabulkou se dvěma základními sloupci `KLIC` a `ODKAZ`. Do sloupce `KLIC` jsou ukládány všechny kombinace řetězců a podřetězců hledané hodnoty, tedy klíče vedoucí po zadání k nalezení záznamu. Sloupec `ODKAZ` je pak cizí klíč s hodnotami odkazujícími na sloupec `ID` v prohledávané (indexované) tabulce. Sloupec `ID` je jednoznačný identifikátor záznamu (primární klíč) v indexované tabulce.



Obr. 2: Princip indexování v UIS

Na výše uvedeném obrázku je názorně naznačen princip indexovacích tabulek v UIS. Tabulka `UZIVATELE` je podkladová tabulka s vlastními daty, tedy v tomto případě s uživateli. Druhá tabulka s názvem `UZIVATELE_INDEX` představuje indexovací tabulku nad tabulku s uživateli.

Nad oběma sloupci v tabulce `UZIVATELE_INDEX` je pak v pořadí (KLIC, ODKAZ) definován kompozitní databázový index zajišťující kýženou rychlost při hle-

dání daného záznamu. Při nalezení hodnoty klíče totiž může databázový stroj zjistit hodnotu odkazu (odpovídající ID hledaného záznamu) přímo z indexu a nemusí číst podkladovou indexovanou tabulku s vlastními daty.

Potřebné klíče mohou být získávány z více sloupců, a to nejenom z vlastní indexované tabulky, ale také ze sloupců jiných tabulek, vztahujících se k záznamům tabulky indexované. K jedné indexované tabulce tak může být definován jeden nebo více zdrojů klíčů. Toto se označuje termínem *zdrojová projekce* [5]. Pro uživatele může takovou zdrojovou projekci představovat například jeho ID, zřetězení sloupců se jménem a příjmením, číslo karty, login nebo také jeho přezdívka apod.

Při konstrukci klíčů je dále k dispozici několik metod, kterými lze ze zdrojové projekce pro jeden záznam základní indexované tabulky získat množinu 1 až n klíčů sloužících k dohledání záznamu. V současnosti jsou v UIS podporovány následující metody:

- indexování celých údajů
- plné podřetězcové indexování
- indexování podřetězců od začátku slova
- indexování jednotlivých slov

Celý systém indexování pak lze konfigurovat za pomoci speciální tabulky, v níž jsou uloženy definice zdrojových projekcí společně s údajem o metodě indexování pro každou indexovanou tabulku.

3.2.2 Aktualizace indexních struktur

Aby vyhledávací funkce neustále korespondovala se skutečným stavem dat v databázi a byla stále aktuální, je potřeba udržovat aktuální také data v indexovacích strukturách. Musí být tedy zajištěno jejich přebudovávání při změně stávajících záznamů a doplňovat při vkládání záznamů nových. Dále je také potřeba zajistit odstranění příslušných souvisejících klíčů při vymazání záznamu.

Indexování probíhá automatizovaně a zcela transparentně. Můžeme tak hovořit o indexaci „on-commit“, tzn. současně při manipulaci s daty ve zdrojové tabulce proběhne i jejich indexování. Vše zajišťují speciální procedury a funkce na databázové úrovni, které jsou spouštěny v tělech triggerů reagujících na DML příkazy nad zdrojovými tabulkami.

Je zřejmé, že tak může docházet ke zpomalení DML operací, nicméně výrazně se toto zpomalení projeví teprve při jednorázové změně většího množství záznamů v tabulce. K takovým zásahům však dochází spíše nárazově a nepravidelně, obvykle v souvislosti s údržbou a vývojem systému. Mnohem častěji dochází k jednorázovým změnám jednoho nebo několika málo záznamů. Takové operace jsou rutinně prováděny přes aplikační rozhraní informačního systému, které je pro drtivou většinu uživatelů jediným možným způsobem, jak provádět změny v databázi [12]. Při běžném provozu je tedy zpomalení DML operací nad indexovanými tabulky zanedbatelné.

3.3 Analýza subsystémů UIS pro nasazení fulltextu

Podstatnou vlastností tohoto principu indexování vyhledávání je, že se indexují jen relativně krátké řetězce (popř. číselné údaje) charakterizující objekt. Tedy pro uživatele ID, jméno,příjmení, u předmětů jeho název, kód, u publikace název, ISBN apod. Po technické stránce je tento mechanismus navržen pro indexování číselných údajů a řetězců do délky 4000 znaků (v databázovém systému Oracle jde o datové typy NUMBER a VARCHAR2). Výše popisovaný mechanismus indexování lze tedy nasadit jen na data uložená ve sloupcích těchto dvou typů (nutno podotknout, že se jedná o dva zdaleka nejpoužívanější datové typy v databázi UIS).

Z tohoto pohledu bychom mohli říci, že se ve své podstatě jedná o indexování metadat. V případě vyhledávání lidí, předmětů, pracovišť a dalších subjektů charakterizovaných zpravidla relativně krátkým názvem, případně dalšími údaji, je uvedený způsob indexování vhodný a dostačující. V těchto případech je totiž více či méně irelevantní rozlišovat metadata záznamů od vlastních dat.

Existují ovšem oblasti UIS, kde lze množinu metadat zřetelně rozlišit od vlastního obsahu záznamu a kde přestává stačit indexovat pouze tato metadata. Typickým příkladem může být Dokumentový server (dále jen DS), který slouží pro ukládání a zveřejňování dokumentů různých typů a formátů, obvykle však jde o dokumenty textové povahy. U těchto dokumentů lze jako metadata označit název dokumentu, jeho krátký popis nebo kdy a kým byl dokument vložen. Tato data lze efektivně indexovat a dokumenty podle nich vyhledávat. Vedle toho vzniká ovšem potřeba umět prohledávat i samotný obsah dokumentu, na což ovšem současný mechanismus není připraven a ani za tímto účelem nebyl navržen. Podobně třeba v diskuzích je potřeba umět indexovat a prohledávat obsahy vlastních příspěvků, a nejen jejich název, k jaké diskuzi se vztahují, kdo je jejich autorem a kdy byly vloženy.

Pro ukládání takových záznamů slouží v UIS datové typy třídy LOB (Large Object), které nabízí databázový systém Oracle. Jedná se konkrétně o datový typ Binary Large Object (BLOB) pro ukládání binárních souborů a typ Character Large Object (CLOB) pro rozsáhlé textové záznamy, které mohou přesáhnout délku 4000 znaků. Datový typ BLOB dovoluje ukládat nejrůznější multimediální soubory (video, audio, obrázky), z našeho pohledu však je nejzajímavější možnost ukládání souborů s textovými daty, jako např. .doc nebo .pdf. V těchto formátech jsou typicky ukládány dokumenty v DS nebo přílohy e-mailů. Do sloupců typu CLOB jsou pak ukládány rozsáhlejší čistě textové záznamy (případně záznamy typu HTML, resp. XML). Tento typ nachází uplatnění např. pro ukládání příspěvků u zmiňovaných diskuzí, překladových řetězců, požadavků v helpdesku, těl e-mailů apod.

Obecně můžeme tedy říci, u všech oblastí UIS, kde se vyskytují (textové) záznamy typu LOB nebo dlouhé záznamy typu VARCHAR2, může potenciálně být žádoucí nasazení fulltextové technologie.

3.4 Požadavky na funkcionalitu fulltextu

Pokusme se nyní specifikovat, co všechno by měla nasazovaná fulltextová technologie splňovat a co se od ní z hlediska zefektivnění práce s UIS očekává. Definované požadavky na funkcionalitu fulltextového vyhledávání sehrají také důležitou roli při výběru technologie, která se pak při vlastní implementaci do UIS použije.

3.4.1 Indexování rozsáhlých textových záznamů

Z předchozí kapitoly vyplývá jeden ze základních požadavků kladených na fulltextovou technologii. Musí být schopna vedle čísel a krátkých řetězců podporovat i objekty typu LOB, tedy musí umět indexovat a prohledávat vedle metadat i vlastní obsahy dokumentů.

Není přitom striktně vyžadováno, aby aktualizace fulltextového indexu proběhla ihned po vložení nebo změně záznamu, což by s sebou mohlo přinést i výrazné zpomalení této operace, zvlášť v případě manipulace s rozsáhlejším textovým dokumentem. Indexace dokumentu tak může proběhnout s určitým definovaným zpožděním.

3.4.2 Podpora fulltextových operátorů

Dále vzhledem k faktu, že indexované obsahy dokumenty bývají zpravidla obsáhlejší, je zapotřebí, aby technologie podporovala poněkud širší možnosti konstrukce fulltextových dotazů. Platí totiž úměra, že čím rozsáhlejší data jsou prohledávány, tím obtížnější je získávání informací z těchto dat. Při vyhledávání by tedy mělo být umožněno použití takových dotazů, kterými lze nalézt co nejvíce relevantní dokumenty.

Pro tyto účely slouží speciální operátory, umožňující zápis takového fulltextového dotazu, kterým dokážeme přesněji vyjádřit, co hledáme. Je žádoucí, aby určitou množinu těchto operátorů podporovala i technologie nasazovaná do UIS.

Základní operátory

V první řadě je důležitá podpora základních logických operátorů, jimiž jsou notoricky známé spojky AND, OR a NOT. Vedle těchto ryze boolských operátorů je žádoucí podpora některých speciálních fulltextových operátorů, mezi něž řadíme NEAR a PHRASE.

Binárním operátorem NEAR říkáme, že dané dva termy se v textu mají vyskytovat blízko sebe. Toto vágní zadání musí být nějakým způsobem přetransformováno do algoritmizovatelné podoby. V nejjednodušším případě se jedná o stanovení hodnoty vyjadřující o kolik slov maximálně mohou být slova z dotazu v textu od sebe vzdálena, aby text (záznam, dokument) operátoru NEAR vyhověl.

Operátorem PHRASE pak říkáme, že příslušná slova se v textu mají vyskytovat vedle sebe. Slouží pro vyhledání textu obsahujícího přesnou posloupnost slov, tedy frázi. Doplňujícím prvkem k tomuto operátoru by mohla být podmínka, že stanovená

posloupnost slov tvoří celý text. Jinak řečeno jde o hledání výhradně jen celého textu specifikovaného v dotazu, tedy text, který sice danou posloupnost slov obsahuje, ale zároveň obsahuje ještě další text, dotazu už nevyhoví. Tato funkcionalita může být velmi užitečná, typicky např. při vyhledávání v překladových řetězcích.

Pokročilejší operátory a funkce
Z pokročilejších fulltextových operátorů je žádoucí operátor pravostranného rozšíření, reprezentovaný obvykle znakem „∗“ uváděným na konci termu v dotazu. To znamená, že např. dotazu „univerz∗“ by vyhověly všechny záznamy obsahující slovo univerz*ita*, univerz*itní*, univerz*ální*, atd. Jedná se tedy o jistou formu podřetězcového vyhledávání. Užitečné by mohly být i jeho analogické modifikace, tedy operátor levostranného rozšíření (uváděný na začátku termu) a operátor oboustranného rozšíření (uváděný uprostřed termu).

Dalším pokročilým operátorem je operátor STEM reprezentovaný zpravidla znakem „$“ uváděným vždy na začátku termu v dotazu. Při aplikaci tohoto operátoru se pro term provede jeho lexikální analýza (skloňování podstatných jmen, časování sloves atd.), pomocí níž se vygenerují další tvary, přes které se bude vyhledávat. Např. dotazu „$jde“ by tak vyhověly i texty obsahující některý z tvarů *jdu, jdeme, jdete, šli, nešlo, jít, jdouce, ...*.

Vyspělé fulltextové technologie nabízí i funkci kontroly pravopisu, tzv. *spelling*. U termů (slov) z uživatelského dotazu se zkontroluje, zda existují ve slovníku – tedy jestli např. slovo neobsahuje překlep. Pokud dané slovo ve slovníku chybí, uživatel zadávající dotaz může být na tuto skutečnost upozorněn, navíc mu může být nabídnuta alternativa správného (nebo jiného) slova.

3.4.3 Integrace s UIS

Při nasazování fulltextového vyhledávání je nutné dbát na relativní snadnost zavádění pro jednotlivé oblasti UIS. Funkcionalita musí být jednoduše a jednotně použitelná pro aplikační programátory při jejím zapracovávání do aplikací.

Právní systém UIS
Vyhledávací technologie musí dále umět dobře spolupracovat s právním subsystémem UIS. Tento subsystém představuje velmi propracovaný soubor uživatelských rolí a práv spolu s přesně definovanými pravidly pro jejich přidělování. Více o implicitním přidělování nároků pojednává [6]. Zobrazování výsledků vyhledávání musí tedy reflektovat vztahy uživatelů k jednotlivým oblastem UIS a v nich pak k jednotlivým záznamům.

Uživatelská přívětivost
Samozřejmý je požadavek na vhodné grafické rozhraní pro zadávání dotazů v souladu se současným stylem ovládání aplikací UIS. Také prezentace výsledků uživateli by měla korespondovat se zavedenými konvencemi UIS a navíc by měla nabídnout prostředky vhodné pro zobrazování větších textových záznamů. Jedná se zejména

o tzv. *highlighting*, čili zvýraznění hledaných termů z dotazu v nalezených záznamech, a dále také o možnost tvorby náhledů, tedy vhodným způsobem zkrácených původních textů pro rychlejší orientaci v nalezené sadě záznamů.

Uživatel by měl dále mít možnost do určité míry ovlivnit způsob zobrazení nalezených výsledků. Zejména by měla být dostupná možnost seřazení výsledků podle různých kritérií, např. podle abecedy, podle logických celků nebo podle relevance. Jen tak může být dosaženo rychlé a efektivní získávání informací z dat rozsáhlého informačního systému.

Pro komfortní a „user friendly“ vyhledávaní je dobré uživateli umožnit zadávat dotazy nezávisle na velikosti písmen, tedy umožnit, aby zadávání dotazu bylo tzv. case insensitive.

Dotaz by mělo být možné dále zadávat s diakritikou i bez ní, resp. nechat uživateli na výběr, zda se diakritika má brát při vyhledávání v úvahu nebo zda se má vyhledávat nezávisle na diakritice.

V neposlední řadě je také kladen důraz na přijatelné doby odezvy při vyhledávání, a to maximálně v řádech sekund.

4 Technologické možnosti

Následující kapitoly budou věnovány analýze a popisu dostupných technologií, které by mohly přicházet v úvahu při implementaci fulltextového vyhledávání do UIS. Tyto možnosti vyplynuly z rozboru funkcionalit a technických řešení jednak samotných produktů, dále pak i z analýzy současného technologického stavu UIS, podmínek při jeho vývoji a v neposlední řadě i ekonomických faktorů, zejména tedy pořizovacích nákladů a nákladů na provoz.

Všechny níže diskutované produkty byly k dispozici k otestování a k vyzkoušení jejich funkčnosti nad reálnými daty UIS. Jen za tohoto předpokladu lze totiž uskutečnit kvalitní analýzu, na jejímž základě je možné vyslovit relevantní závěr vedoucí k volbě nejvíce optimálního řešení.

Kromě komerčně dostupných „hotových“ produktů je diskutována také možnost vývoje vlastní fulltextové technologie, což také představuje jednu z možných cest při implementaci fulltextového vyhledávání do UIS. Je zapotřebí se tedy zabývat i touto možností, i když by se na první pohled mohla zdát jako méně schůdné řešení.

4.1 Google Search Appliance

Google Search Appliance představuje komplexní nástroj pro vyhledávání v datech dané společnosti. Používá se v rámci počítačového prostoru společnosti, tedy za firewallem, a umožňuje vyhledávat a zobrazovat informace uložené v intranetu, na webových a souborových serverech společnosti, v systémech pro správu obsahu, v databázích a dalších zdrojích. Umožňuje vyhledávat informace uložené ve více než 220 různých souborových formátech a ve více než 109 jazycích. Jedná se o hardwarové řešení s integrovaným vyhledávacím mechanismem – předinstalovaným vyhledávačem Google – určeným k indexování dokumentů na podnikových serverech. Následuje popis charakteristických vlastností produktu. Detailní informace je možno nalézt v [20] a [24].

4.1.1 Vlastnosti

Díky komplexní funkci Google One Box for Enterprise lze v Google Search Appliance snadno, bezpečně a v reálném čase získávat informace z firemních aplikací – ze zaměstnaneckého adresáře, kalendářů, CRM, ERP nebo BI systémů, a to všechno prostřednictvím jednoduchého a vesměs důvěrně známého rozhraní Google.

Google Search Appliance je určena pro účely vyhledávání vysoce relevantních výsledků, prohledávání rozsáhlé sady dokumentů, pro účely získávání bezpečných výsledků nebo prohledávání dat uložených v aplikacích vyvinutých třetími stranami (ERP, CRM, BI nebo další systémy). Google Search Appliance lze použít i mimo firewall, například na webových stránkách a prezentacích, a umožnit jejich návštěvníkům rychle vyhledat požadovanou informaci na webu společnosti.

Nabízí svým uživatelům řadu jednoduché a intuitivní ovládání a vyhledávání podle řady jednotlivých vyhledávácích kritérií, podle kterých hledá nejvhodnější dokumenty. Následuje výčet významných vlastností z uživatelského pohledu.

- **Dynamické přehledy stránek** – uživatelé mohou snadno posoudit výsledek již vyhledaných informací prostřednictvím dynamicky vytvářeních úryvků, které zobrazí zadaný dotaz v kontextu celé stránky.
- **Třídění výsledků** – orientace ve výsledcích vyhledávání je snadná díky použití inteligentního třídění dokumentů, které jsou řazeny do logicky souvisejících podadresářů.
- **Řazení a omezení dokumentů** – řazení dle data, možnost omezení výsledků prostřednictvím data, omezení počtu zobrazení jednotlivých výsledků vyhledávání na dané časové období, respektive datum.
- **Možnosti automatických oprav** – Google Search Appliance automaticky navrhuje správné tvary slov a frází a tím se snaží eliminovat pravopisné chyby.
- **Načtení stránek do vyhledávací paměti** – umožňuje vyhledávání i v rámci momentálně nedostupných stránek, které jsou od poslední návštěvy uložené ve vyrovnávací paměti.
- **Zvýraznění klíčových slov** – rychlá orientace v nalezených částech dokumentu prostřednictvím zvýraznění klíčových slov
- **HTML náhled** – umožňuje zobrazení dokumentů v HTML formátu bez nutnosti využití příslušné aplikace díky automatickému převodu z více než 220 formátů do formátu HTML.
- **Pokročilé vyhledávání** – při specifikaci dotazu umožňuje Google použití více než deseti různých logických výrazů (AND, OR, NOT a další).

4.1.2 Správa a možnosti přizpůsobení

Spojením HW a SW do jediného zařízení dochází k usnadnění správy dat Google Search Appliance a lze tak jednoduše nainstalovat a spravovat prostřednictvím jednoho správce. Je k dispozici Webové rozhraní pro správu umožňující konfiguraci uživatelských účtů a rolí pro prohledávání, obsluhu a sledování s jednoduchým uživatelským rozhraním s intuitivním ovládáním. Následuje výčet nejdůležitějších možností podporujících dobré přizpůsobení produktu.

- **SNMP dohled** – monitoring, zabezpečené prohledávání a vytváření statistik prostřednictvím standardního SNMP rozhraní.
- **Neomezená možnost vytváření různých množin výsledků** – rozdělení indexu výsledků pro různé uživatele (např. podle domény, lokace, pracovního zařazení a podobně).
- **Filtry** – jednoduchá aplikace filtrů na jazyk, typ souborů, webové stránky či meta tagů AND/OR.

- **Synonyma** – možnost definování a rozpoznávání synonym, často používaných podnikových zkratek a terminologie s možností nabídky návrhu alternativních dotazů.
- **Možnost definování vazeb** – vytváření vazeb mezi URL adresami a klíčovými slovy. Cílové adresy se zobrazí nad množinou výsledků vyhledávání.
- **Nastavení uživatelského rozhraní** – prostřednictvím XSLT šablon lze provádět uživatelské úpravy zobrazení a vzhledu výsledků vyhledávání. Různé části webu lze přizpůsobit různým grafickým stylům.
- **Statistiky** – statistiky každodenních nebo často se opakujících výsledků vyhledávání, nejpoužívanějších dotazů či použití speciálních funkcí, atd. s možností exportu.
- **Podpora RAID** – zapojení disků do RAID pole zvyšuje spolehlivost a eliminuje rizika při selhání disků.
- **Vzdálená diagnostika** – volitelně je možné využívat správu prostřednictvím vzdáleného připojení.

4.1.3 Bezpečnost vyhledávání v datech

Každá společnost má dva druhy dat: data, která chce zpřístupnit všem mimo firemní firewall, a citlivé informace, které jsou k dispozici pouze omezenému okruhu uživatelů. Google Search Appliance nabízí snadnou integraci se stávajícími technologiemi zajišťujícími ochranu dat, jako jsou LDAP, Active Directory, NTLM, X.509, systémy Single Sign-On a další. Kontrola přístupu probíhá v reálném čase, a je tak zajištěno, že informace se dostanou pouze k autorizovanému uživateli.

Řešení Google Search Appliance poskytuje správu přístupů ke konkrétním dokumentům až na úroveň konkrétního uživatele. Díky tomu je zajištěno,že mohou v rámci celého podniku uživatelé vyhledávat pouze v rámci těch dokumentů, ke kterým mají nastaven přístup. Nastavení práv je možné propojit s existujícím nastavením přístupových práv a omezit tak přístup na jednotlivé dokumenty a aplikace.

4.1.4 Google Desktop a Google Toolbar

Google Desktop pro Enterprise umožňuje uživatelům snadné vyhledávání informací uložených na pevném disku jejich počítače a prohlížení výsledků vyhledávání z intranetu, vyhledávače Google.com a jejich počítače prostřednictvím jednoho jediného rozhraní.

Verze Enterprise k těmto nástrojům umožňuje firmám centrálně kontrolovat, řídit a implementovat uživatelské funkce/preference, šifrovat uživatelská data ve vyhledávacích indexech a vynucovat strategie pro práci s dokumenty

Google Desktop a Toolbar pro aplikaci Enterprise jsou ke stažení zdarma. K Desktop for Enterprise lze ovšem též přikoupit prémiovou podporu (Premium Support Package), jež nabízí konzultace a podporu při instalaci těchto nástrojů.

4.2 Oracle Text

Oracle Text představuje proprietární technologii společnosti Oracle, která umožňuje vývoj databázových aplikací určených k vyhledávání v textech uložených v databázi. Disponuje celou řadou nástrojů určených pro indexování textů a textových souborů, hledání slov uvnitř textových dokumentů, hledání dokumentů podle jejich tematické klasifikace a v neposlední řadě poskytuje i prostředky podporující prezentaci a další zpracování nalezených dat.

Balík Oracle Text je dnes již standardní součástí moderních relačních databázových systémů dodávaných společností Oracle. Jelikož je Univerzitní informační systém vybudován právě nad zmíněným databázovým produktem, představuje tento doplňující softwarový balík jedno z možných řešení pro implementaci fulltextového vyhledávání. Podrobné informace lze získat z [21] nebo [22]. Zde se zaměříme pouze na z našeho pohledu podstatné vlastnosti.

4.2.1 Podporované typy aplikací

Při návrhu uživatelské aplikace využívající Oracle Text je potřeba nejdříve specifikovat typy dotazů, které bude umět aplikace zpracovat. Rozlišují se tři velké základní kategorie aplikací:

- dotazy nad kolekcí textových dokumentů,
- dotazy nad skupinami údajů (tzv. „katalogové“ dotazy),
- dotazy pro klasifikaci dokumentů.

S ohledem na charakter aktivit a dat Univerzitního informačního systému bude pro nás nejzajímavější první kategorie, a sice dotazy nad kolekcí textových dokumentů. Nadále se tedy budeme zabývat především možnostmi této kategorie. O zbývajících dvou kategoriích se zmíníme jen okrajově pro úplnost. Pro podrobnější informace odkazuji na dokumentaci k systému Oracle ([21], resp. [22]).

Uživatelské aplikace vyvinuté pro první kategorii dotazů umožňují nalézt relevantní textový dokument nebo více dokumentů v množině textových dokumentů různých velikostí a formátů (nejen obyčejných řetězců a textů, ale např. i souborů HTML, PDF nebo dokumenty typu MS Word či MS PowerPoint a dalších). Jejich úplný výčet je k dispozici v [22]. Jak již bylo nastíněno v úvodní kapitole, vyhledávání je samozřejmě i zde podmíněno předchozím naindexováním všech dokumentů z této kolekce. Kolekce takových dokumentů bývá uložena v nějakém definovaném datovém úložišti (zpravidla v databázové tabulce, ale není podmínkou) a má obvykle spíše statický charakter. V tomto kontextu to znamená, že po úvodním naindexování jednotlivých dokumentů se jejich obsah v čase buď už vůbec nemění nebo se mění jen málo a ne příliš často. Za to dochází k průběžnému nárůstu počtu dokumentů v dané kolekci, což způsobuje neustálé zvětšování objemu prohledávaných textových dat. Spolu s tím pak rostou i požadavky na výkon a na prostorovou náročnost

fulltextových indexů. Pro větší objemy textových dat jsou pak také speciálně koncipovány vyhledávací dotazy. Pro uvedené podmínky se proto jeví jako nejvhodnější právě oblast indexování a vyhledávání nad kolekcí textových dokumentů.

Typická aplikace s textovými dotazy umožní uživateli zadat dotaz, který zpracuje a vrátí seznam dokumentů vyhovujících zadanému dotazu (tzv. *hitlist*). Dále aplikace umožní uživateli dokumenty prohlížet a dále s nimi pracovat. Obecně tedy takové aplikace můžeme charakterizovat následující posloupností kroků:

1. uživatel zadá dotaz,
2. aplikace zpracuje dotaz použitím operátoru CONTAINS (viz. dále),
3. aplikace zobrazí hitlist,
4. uživatel si vybere dokumenty z hitlistu,
5. aplikace prezentuje dokumenty a umožňuje uživateli s nimi nějakým způsobem dále pracovat.

Jedním z nejdůležitějších aspektů těchto aplikací je vyhledání co nejvíce relevantních dokumentů a naopak nerelevantních co nejméně. Nejvíce relevantní dokumenty je pak žádoucí při prezentování uživateli zařadit na přední pozice seznamu nalezených dokumentů.

4.2.2 Typy indexů

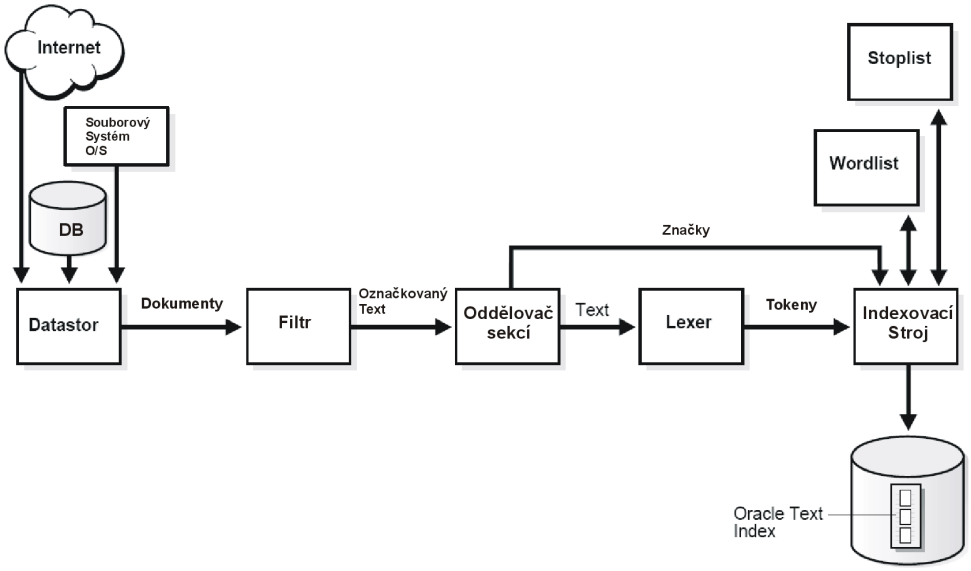
Pomocí technologií používaných v balíku Oracle Text je možné vytvořit čtyři typy indexů:

- **CONTEXT** index – slouží k vytváření vyhledávacích aplikací nad kolekcí rozsáhlých koherentních dokumentů. Dokumenty mohou být v různých formátech jako např. MS Word, HTML nebo čistý text. Tento index nabízí z ostatních jmenovaných nejvíce možností přizpůsobení pro daný účel.
- **CTXCAT** index – slouží k rychlejšímu zpracování tzv. kombinovaných dotazů. Do indexu jsou zahrnuty další sloupce jako název, cena apod. a je tak zefektivněno kombinované vyhledávání podle takových údajů. Tento index je vhodný pro indexování menších textových dokumentů nebo fragmentů textu.
- **CTXRULE** index – používá se pro vytváření klasifikace dokumentů a pro směrování aplikací. Tento index se vytváří nad tabulkou s dotazy, které definují klasifikační kritéria.
- **CTXXPATH** index – lze jej využít pro vyhledávací dotazy nad sloupci typu XMLType.

Nejzajímavější pro nás bude CONTEXT index, kterým lze indexovat i rozsáhlé soubory obsahující textová data. Tímto indexem Oracle Text indexuje texty za pomoci konverze textu do tzv. *tokenů*, což jsou jisté fragmenty textu, zpravidla slova. Základ struktury CONTEXT indexu tvoří invertovaný index, v němž se ke každému tokenu přiřazuje seznam dokumentů, v nichž se token vyskytuje a další přídavné informace.

4.2.3 Indexovací proces

Cílem indexovacího procesu je vytvořit Oracle Text index v souladu s požadavky specifikovanými při vytváření indexu. Celý proces indexování začíná DDL příkazem CREATE INDEX, který poskytuje různé možnosti nastavení preferencí. Jak ukazuje následující obrázek, indexovaný text prochází během procesu indexování několika sekcemi.



Obr. 3: Oracle Text – indexovací proces

V *datastoru* začíná indexovací proces čtením dokumentů v závislosti na způsobu jejich uložení. Dokumenty mohou být uloženy přímo v databázi, v souborovém systému operačního systému nebo na internetu. Ať už jsou dokumenty fyzicky uloženy kdekoliv, musí existovat tabulka obsahující vždy buď samotné dokumenty nebo ukazatele na soubory s dokumenty.

Poté indexovaná data putují do *filtru*. Co se s daty stane v této sekci závisí na nastavení filtrovacích preferencí. V případě čistě textových dokumentů typu HTML nebo XML se neprovádí žádná filtrace. Formátované (binární) dokumenty je ovšem již potřeba filtrovat. Výsledkem filtrování je text opatřený speciálními značkami. Další možností tohoto objektu je konverze do znakové sady databáze.

Po filtrování postupuje (označkovaný) text do *oddělovače textových sekcí*, který od sebe oddělí vlastní text od informací o textu. Typ informací jde opět ovlivnit nastavením. Tyto informace pak postupují přímo do *indexovacího stroje*, který je později použit. Vlastní text musí nejdřív podstoupit proceduru v *lexeru*. Tento objekt má za úkol rozložit text do tokenů v závislosti na jazyku textu a dalších nastaveních. Tokeny představují určité řetězce (obvykle slova) a pomocí různých specifikací definovaných při vytváření indexu je možno jejich tvary a uložení ovlivnit. Lze tak

například stanovit oddělovací znaky (tzv. bílé), určit, zda se mají řetězce ukládat pouze ve formátu velkých písmen apod.

Indexovací stroj pak vytváří vlastní invertovaný index mapováním tokenů na dokumenty, kde se vyskytují. V této fázi Oracle Text využívá dalších přídavných informací, například tzv. *stoplistu*, který vylučuje použití některých příliš obecných slov pro účely indexování. Dále využívá preferencí tzv. *wordlistu*, který říká například jakým způsobem se mají vytvářet prefixy nebo podřetězce.

4.2.4 Dotazování

Vyhledávací dotazy obvykle tvoří slova nebo skupiny slov, mezi kterými lze specifikovat vztahy. Uživateli je pak umožněno zadávat i jejich různé kombinace, a to nejenom použitím notoricky známých logických operátorů jako AND nebo OR, ale i dalších speciálních operátorů, jejichž úplný výčet je možné nalézt v dokumentaci systému Oracle [22].

Pro konstrukci fulltextových dotazů nad kolekcí textových dokumentů se používá operátor CONTAINS. Tento operátor se uvádí v části WHERE standardního databázového příkazu SELECT a lze jej použít jen tehdy, pokud je nad příslušným sloupcem z prohledávané tabulky vytvořen index typu CONTEXT. Zde je naznačena syntaxe dotazovacího operátoru CONTAINS:

```
CONTAINS(
       [schema.]sloupec,
       text_dotaz VARCHAR2
      [,label NUMBER])
RETURN NUMBER;
```

sloupec – určuje sloupec tabulky, který má být prohledáván. Nad tímto sloupcem musí být vytvořen Oracle Text index typu CONTEXT.
text_dotaz – představuje buďto řetězec nebo řetězcový výraz, vyjadřující co se má hledat, nebo to také může být speciální označkovaný dokument specifikující tzv. dotazovací šablonu. Dotazovacích šablon je více typů a pro zájemce jsou podrobně rozepsány v dokumentaci k systému Oracle.
label – tato nepovinná volba určuje, zda se má k nalezeným dokumentům generovat také tzv. skóre, tedy údaj určující relevanci dokumentu (úzce souvisí s operátorem SCORE, viz dále).

Vyhledávací operátor CONTAINS umožňuje použití celé řady speciálních fulltextových operátorů, kterými lze docílit přesnější definování a specifikaci vyhledávacích kritérií. Tyto operátory umožňují jednak vyjádřit logické vazby v případě více slov ve vyhledávacím dotazu, dále pak umožňují například také proximitivní a fuzzy vyhledávání, použití stemmingu nebo thesauru. Pro jejich podrobný popis

opět odkazuji na dokumentaci k systému Oracle. Za předpokladu patřičné konfigurace indexu je možné i vyhledávání ve vybraných částech dokumentů typu HTML resp. XML.

Dále pro každý vrácený řádek operátor CONTAINS vrací číslo mezi 0 až 100, které indikuje, do jaké míry je nalezený dokument v hitlistu relevantní vůči položenému dotazu. Číslo 0 znamená, že dokument zadaný vzorek vůbec neobsahuje. K získání tohoto čísla slouží operátoru SCORE uváděný spolu s návěštím *label*. Následující příklad ukazuje, jak může vypadat jednoduchý fulltextový SQL dotaz s použitím operátoru CONTAINS a s využitím indikátoru relevance.

```
SELECT SCORE(1), nazev from DOKUMENTY
    WHERE CONTAINS(TEXT, 'informační', 1) > 0
    ORDER BY SCORE(1) DESC;
```

Uvedený příkaz SELECT najde všechny dokumenty ve sloupci TEXT tabulky `DOKUMENTY`, které obsahují slovo „*informační*“. Pro smysluplné využití je zapotřebí, aby operátor CONTAINS byl následován výrazem typu > 0. Tím se totiž zajistí, že ve výsledku se objeví pouze záznamy ohodnocené číslem větším než 0, jinými slovy dokumenty obsahující zadaný výraz. Pokud je volán operátor SCORE (např. v části příkazu SELECT), klauzule CONTAINS na něj musí odkazovat příslušnou hodnotou návěští *label*, jak vidíme na příkladu. Použitím operátoru SCORE v kombinaci s klauzulí SQL jazyka ORDER BY DESC pak dosáhneme setřídění nalezených záznamů od nejvíce relevantních po nejméně relevantní.

Pro ukázku použití různých fulltextových operátorů přesněji specifikujících vyhledávací kritéria si můžeme představit stejnou konstrukci dotazu jako v předchozím příkladě a zaměřit pozornost pouze na hledaný řetězec v uvozovkách. Jak již bylo řečeno, může být tento řetězec vytvořen kombinací několika řetězců se stanovením vazeb mezi částmi dotazu. Zde se otevírá prostor pro využití širokého spektra nabízených operátorů. Místo prostého řetězce „*finance*“ z předchozího příkladu tak můžeme psát např. „*informační systém*“ nebo zkráceně „*informační & systém*“.

Lze zadávat i mnohem složitější kombinace a používat přitom různých operátorů, které Oracle Text nabízí. Při vyhodnocování takových výrazů se databázový systém řídí buďto explicitně zadanými závorkami, nebo implicitně podle stanovené priority jednotlivých operátorů.

4.2.5 Jazyková podpora

Oracle Text podporuje indexování pro různé jazyky, umožňujíce zvolit typ lexeru při indexovacím procesu. Použití jednotlivých typů lexeru je závislé na jazyku. Z lexerů podporujících více jazyků uveďme alespoň tyto nejvýznamnější:

- **BASIC_LEXER** podporuje angličtinu a většinu západoevropských jazyků, které používají jako oddělovače slov bílé znaky. S tímto lexerem je možné indexovat sloupec s dokumenty v jednom konkrétním jazyce.

- **MULTI_LEXER** je určen pro indexování tabulek obsahujících dokumenty v různých jazycích. V jednom sloupci tabulky tak mohou být uloženy dokumenty v angličtině, němčině, nebo třeba japonštině. Pro tento lexer je zapotřebí mít v tabulce ještě přídavný sloupec s označením jazyku příslušného dokumentu.
- **WORLD_LEXER** tak jako MULTI_LEXER umožňuje indexovat dokumenty v různých jazycích, avšak automaticky detekuje jazyk dokumentu a nepotřebuje tedy žádný speciální jazykový sloupec. Tento lexer podporuje velkou většinu všech jazyků.

Nejširší možnosti parametrizace indexování poskytuje BASIC_LEXER, nicméně tyto vyspělé funkce poskytuje pouze pro vybrané jazyky, mezi něž čeština nepatří. Pro češtinu tak není dobře podporována například konverze znaků do jejich základní podoby (tedy možnost ukládat při indexování znaky s diakritikou i bez ní), dále není podporováno indexování kořenů slov, alternování pravopisu a mnohé další užitečné funkce.

Také při dotazování jsou některé operátory dostupné pouze pro vybrané jazyky, mezi které není zahrnuta čeština. Při dotazování nad dokumenty v češtině jsme tak ochuzeni o *fuzzy* operátor, *stem* (kořenový) operátor a *about* operátor. Pro vybrané jazyky (mezi něž rovněž čeština nepatří) v sobě Oracle Text zahrnuje i defaultní stoplisty.

V dokumentech databáze UIS je samozřejmě převaha českých dokumentů, nicméně vyskytují se v něm i dokumenty v jiných jazycích a mohou se vyskytovat i dokumenty smíšené, kdy je část napsaná v jednom jazyce a část v jiném. Nasazení technologie Oracle Text za účelem umožnění fulltextového prohledávání dokumentů by umožnilo dosáhnout poměrně vysoké úrovně fulltextového vyhledávání, avšak pro češtinu – jakožto převažující jazyk v dokumentech UIS – nelze využít všech funkcí, které Oracle Text nabízí a nelze tak dosáhnout nejvyššího stupně uživatelského komfortu.

4.3 ConText CZ

Pro plnohodnotné využití balíku Oracle Text v UIS se naskýtá možnost využití komerčního produktu CONTEXT CZ firmy Sefira, který představuje lokalizaci produktu Oracle Text do češtiny. Nasazením tohoto produktu je již možno pro češtinu využít vyspělých funkcí, jenž Oracle Text nabízí a poskytnout uživatelům vysoce kvalitní fulltextové vyhledávání.

Sefira je česká SW firma, která se zaměřuje zejména na vývoj specializovaných informačních systémů a aplikací pro prostředí intranet/internet. Produkty této firmy jsou zaměřeny zejména na bezpečnost, automatizaci knihoven a informačních středisek, fulltextové systémy, práci s dokumenty atd. Podrobněji viz [25].

Pro realizaci svých produktů využívá společnost Sefira od svého vzniku výhradně technologií Oracle a je také certifikovaným partnerem společnosti Oracle. Pro naše účely nejzajímavějším produktem je již zmiňovaný ConText CZ. Pomocí

tohoto produktu je možno fulltextově vyhledávat v textech psaných v českém jazyce podle slovního základu. Jedná se o rozšíření funkčnosti balíku Oracle Text o práci s českými texty [25].

Tato jazyková lokalizace spočívá v tom, že při použití operátoru *stem* (nebo ekvivalentně znak $) se na jednoduchý i komplikovanější český výraz provede jeho lexikální analýza (skloňování podstatných jmen, časování sloves atd.), která zjistí, dle jakého slovního základu se mají pro tento výraz generovat další tvary, přes které se bude vyhledávat. Jako základ slouží speciální datový slovník a sada algoritmů, které byly sestaveny skupinou odborníků zabývajících se matematickou lingvistikou. Následující přehled, tak jak je uveden v [25], ukazuje, co CONTEXT CZ ve spojení s Oracle Text nabízí:

- možnost vyhledání českých slov v dokumentech i strukturovaných datech,
  - vyhledávání ve všech časech, resp. pádech,
  - vyhledávání podle stejného slovního základu nebo kmene,

- vyhledávání operátorem **stem**,
  - například po zadání vyhledání slova *jíst* bude výsledková sada obsahovat dokumenty obsahující toto slovo i ve tvarech *jedla*, *jez*, *jíme* atd.,
  - dále například po zadání vyhledání slova *kůň* bude výsledková sada obsahovat dokumenty obsahující toto slovo i ve tvarech *koně*, *koňmi* atd.,

- vyhledávání s různými operátory (AND, OR, NOT, NEAR),

- možnost třídit texty stejně efektivně, jako strukturovaná data,

- možnost použití ConTexT CZ i v již existujících aplikacích nad databází Oracle,

- možnost rozšířit funkčnost aplikací pracujících se strukturovanými daty – textové položky (názvy knih, atd.).

Firma Sefira poskytuje instalaci produktu ConTexT CZ standardně spolu s patřičnými školeními, dokumentací a technickou podporou. Nadstandardně pak nabízí volitelně vývoj fulltextových aplikací, integraci fulltextové technologie pro správu dokumentů, poradenství a konzultace.

Principiálně lze tedy říci, že pro lokalizaci fulltextového vyhledávání pro český jazyk se při definování, konfiguraci a vytváření fulltextových indexů postupuje shodným způsobem, jako např. pro anglický jazyk.

Okamžik, kdy přichází ke slovu ConText CZ je okamžik odeslání dotazu ke zpracování, čili předání operátoru CONTAINS. Než je takto celý hledaný výraz předán do jádra, je na něj aplikována stěžejní funkce produktu ConText CZ s názvem CZ, která zajišťuje právě lexikální rozvoj každého klíče v dotazu (resp. každého klíče, u kterého je rozvoj vyžadován uplatněním fulltextového operátoru $ na začátku klíče).

Pro podmínky vývoje UIS na MZLU v Brně je zapotřebí produkt ConTextT CZ nainstalovat a vyvinout k němu aplikační rozhraní. ConText CZ funguje jako nadstavba nad produktem Oracle Text. Hlavní funkcí ConText CZ je zajistit pro české tvary hledaných klíčů jejich lexikální rozvoj podle slovního základu. Jinými slovy pro požadovaný klíč vygeneruje všechny jeho ostatní české tvary. Po tomto rozvoji fulltextový dotaz postupuje dále do jádra Oracle Text, kde je zpracován a vyhodnocen standardním způsobem jako dotazy obsahující slova ostatních běžně podporovaných jazyků (např. angličtiny).

4.4 Vývoj vlastní fulltextové technologie

Vedle hotových řešení (ve většině případů komerčních) je jednou z možných cest při implementaci fulltextového vyhledávání do UIS i vývoj vlastní fulltextové technologie. Na první pohled by se mohlo zdát, že se jedná o méně schůdné a po všech stránkách náročné řešení. Vhledem ke specifickým požadavkům kladeným na fulltextovou technologii ze strany UIS však vývoj vlastní technologie může vést k uspokojivému řešení.

Využitím teoretických poznatků o indexování a vyhledávání spolu s uplatněním zkušeností z dosavadních používaných technik lze vyvinout technologii, která bude splňovat kladené požadavky. Základním bodem je kvalitní analýza a návrh – jen tak je možné zajistit, aby technologie disponovala potřebnými funkcemi včetně těch pokročilých a dosahovala vysoké úrovně kvality. Při implementaci pak může být zužitkováno know-how, získané při vývoji informačního systému a lze využít i zkušenosti získané z dosavadního způsobu indexování.

4.5 Volba vhodné fulltextové technologie

Při výběru vhodné fulltextové technologie pro nasazení do UIS je nutné posuzovat jednotlivé výše analyzované možnosti z několika různých hledisek. Základním předpokladem je, aby zvolené řešení poskytlo definovanou funkcionalitu, přičemž musí plně respektovat podmínky prostředí, ve kterém je UIS provozován (zejména pak propracovaný systém oprávnění). Dalším kritériem je technická i časová náročnost implementace technologie. V neposlení řadě hraje důležitou roli také ekonomická stránka věci, tedy zejména náklady na pořízení a provoz technologie.

Nasazení Google Search Appliance naráží na jisté problémy související s licenční politikou produktu (podrobněji v [20] a [24]) ve spojení s chápáním pojmu dokument. Z hlediska Google Search Appliance je za jeden unikátní indexovaný dokument považováno každé jedinečné URL. Z hlediska UIS je však za jeden dokument považován daný objekt UIS se všemi jeho informacemi. Například při vyhledávání osob by tak za samostatné dokumenty byly považovány i jednotlivé záložky, případně podzáložky v aplikaci pro zobrazování informací o člověku. Takových záložek (resp. jedinečných URL adres) může být pro jednoho člověka i desítky. Počet dokumentů tak enormně vzrůstá a pro reálný provoz by bylo nutné nasazení nejsilnější verze

produktu, která nemá omezení na maximální počet indexovaných dokumentů. Tato skutečnost ovšem posouvá produkt za hranice rentability. Řešení navíc nedisponuje vhodnými prostředky pro integraci s právním systémem UIS, což by mohlo vést k přístupu uživatelů i k dokumentům, ke kterým nemají oprávnění.

Technologie Oracle Text jako součást databázového systému, nad kterým je UIS vystavěn, je robustním nástrojem pro dosažení požadované funkcionality. Koncepce a způsob použití technologie umožňuje plnou integrovatelnost s prostředím UIS, včetně jeho právního systému. Její nasazení ovšem není triviální a vyžaduje podrobné studium velmi rozsáhlé dokumentace. Technologie totiž disponuje obrovským množstvím různých možností nastavení a konfigurace, z nichž zdaleka ne všechny jsou vhodné pro nasazení do UIS. Při implementaci technologie je tak potřeba prodělat několik fází vývoje s testováním a zpětnou vazbou. Pro testovací účely bylo vyvinuto prototypové rozhraní a technologie byla nasazena do oblasti překladových řetězců, která je dostupná pouze úzké skupině uživatelů tvořené zejména vývojovými pracovníky. Při některých specifických dotazech se databázová instance dostávala do stavů, kde vykazovala prvky nestability. Za těchto podmínek zatím nebylo možné technologii nasadit do reálného provozu UIS a zpřístupnit technologii širokému spektru uživatelů. Předopkladem při podstoupení další fáze ladění a přizpůsobování technologie je postupné odstraňování nežádoucích vlivů na běh databáze a dosažení stabilnějšího řešení.

Paralelně s laděním a přizpůsobováním technologie Oracle Text je další volbou také implementace vlastní fulltextové technologie. Výhodou tohoto způsobu řešení je vývoj technologie na míru přesně pro účely UIS se zachováním výhradní kontroly nad každou z fází indexovacího a vyhledávacího procesu. Technologie postupně obsáhne všechny požadované funkcionality a nabídne všechny užitečné fulltextové operátory. S využitím nástroje ConText CZ je možné do technologie zabudovat i vysoce pokročilý fulltextový operátor stemmingu. Implementací vlastního řešení fulltextového vyhledávání se zabývají následující kapitoly.

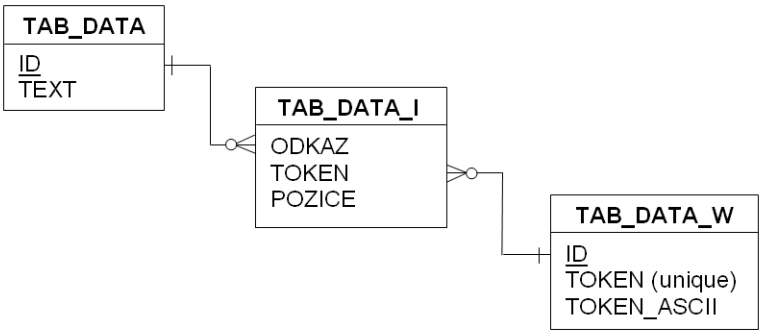
5 Implementace fulltextových technologií do UIS

5.1 Realizace fulltextového indexování

5.1.1 Indexovací struktury

Pro fulltextové indexování je charakteristické použití speciálního, tzv. inverzního indexu. Konstrukce inverzního indexu je postavena na odlišné filozofii než je tomu u klasických indexů doposud používaných v UIS. Klasické indexování je založeno na myšlence ke každému objektu definovaným způsobem vytvořit a přiřadit sadu klíčů, jejichž zadání vede k nalezení konkrétního záznamu (záznamů). Tuto myšlenku zachycuje struktura na obr. 2 v kapitole 3.2.1.

Naproti tomu u inverzního indexu je cílem ze zdrojových (indexovaných) dokumentů extrahovat jisté fragmenty textu – tokeny – a ke každému takovému fragmentu pak přiřadit seznam dokumentů, ve kterých se vyskytuje. Inverzní index může obsahovat i další informace (např. pozici tokenu v konkrétním dokumentu, apod.) potřebné pro implementaci různých fulltextových vyhledávácích operátorů. Tato myšlenka je spolu s poznatky uvedenými v [16] také výchozím bodem při návrhu a implementaci inverzních indexů v UIS. Na následujícím obrázku je zachycena základní podoba indexovací struktury implementující inverzní index v UIS.



Obr. 4: Struktura inverzního indexu

Nutnou podmínkou je, aby indexovaná tabulka se zdrojovými daty – v tomto případě fiktivní tabulka `TAB_DATA` – měla (jednosloupcový) primární klíč, aby bylo možné daný záznam jednoznačně identifikovat při mapování tokenu na výskyt v dokumentu. Samotné mapování je zachyceno v tabulce `TAB_DATA_I`, kterou tvoří odkaz na zdrojový dokument (přesněji odkaz na jeho `ID`) a odkaz na daný token (na `ID` tokenu v tabulce `TAB_DATA_W`). Sloupec `POZICE` pak slouží k zachycení pozice daného tokenu v dokumentu (resp. zde postačuje informace o pořadí tokenu). V tabulce `TAB_DATA_W` je pak uchováván seznam všech tokenů z celé sady indexovaných dokumentů.

Pokud bychom chtěli pohlížet na každý textový záznam jako na vektor tokenů, pak si tabulku `TAB_DATA_I` lze představit jako sadu vektorů. Každý z těchto vektorů

je jednoznačně přiřazen právě jednomu dokumentu, tedy každý dokument je reprezentován vektorem fyzicky uloženým v této mapovací tabulce. Dostáváme tak dobrý základ pro realizaci jednak některých fulltextových operátorů a také pro implementaci ohodnocovací funkce určené ke kalkulaci míry relevance nalezeného dokumentu, jak uvidíme později.

I u tohoto způsobu indexování lze zavést obdobu zdrojové projekce z klasického indexování. V tomto pojetí se jedná o indexování více sloupců jedné tabulky, které umožní jejich pozdější prohledávání. Při vyhledávání tak bude možné stanovit šířku hledání, čili zvolit, které sloupce mají být prohledávány. S výhodou se pro to dá využít jeden seznam tokenů, vytvářený pro každou tabulku (tabulka `TAB_DATA_W`) a k němu pro každý indexovaný sloupec z dané tabulky n mapovacích tabulek (`TAB_DATA_I1` až `TAB_DATA_I`n).

Při tomto způsobu indexování vzniká nutnost vytvoření evidence fulltextově indexovaných oblastí v podobě konfigurační tabulky, z níž bude moci indexovací i vyhledávání mechanismus zjistit potřebné údaje pro svůj korektní průběh.

Pro indexaci sloupců typu BLOB je vždy zapotřebí nejprve extrahovat textová data ze souborů a uložit je do konverzní tabulky se strukturou (ID, PLAIN_TEXT), kde sloupec ID koresponduje s hodnotami z původní tabulky. S touto tabulkou pak bude při indexování zacházeno jako ze zdrojovou indexovanou tabulkou. Údaje o konverzní tabulce jsou rovněž součástí konfigurační tabulky.

5.1.2 Příprava zdrojových dat

Při indexování dokumentů prochází zdrojová textová data procesem, který tato data přetransformuje do vhodné podoby tak, aby se dal realizovat efektivní mechanismus vyhledávání v dokumentech. Výchozím bodem tohoto procesu je úvodní vyčištění původního textu odstraněním nepotřebných znaků, případně sekvencí znaků. V případě binárních dokumentů obsahujících textová data tomuto bodu předchází ještě získání vlastního textu (plain textu) z formátovaného binárního souboru.

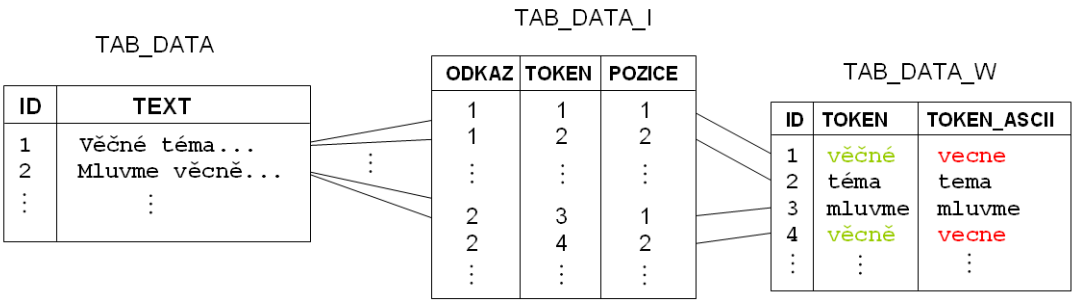
Dále je nutné stanovit způsob extrahování tokenů z textu. Tokenem je fragment textu, který v sobě nese určitý význam. Zpravidla jím bývá elementární lexikální jednotka daného jazyka. V případě jazyků, kde se jako oddělovače jednotlivých lexikálních jednotek používají bílé znaky, lze tedy jednoduše považovat za token vše mimo bílé znaky. Text je tak vlastně tvořen posloupností tokenů, jejichž hranice jsou vymezeny přítomností bílých znaků. Pokud tuto základní myšlenku rozšíříme tak, že kromě bílých znaků stanovíme jako oddělovače jednotlivých tokenů i další nevýznamové znaky (např. `.,!?;"()[]|` a další), dostaneme mechanismus, podle kterého je možné tokeny z textu extrahovat. V různých oblastech UIS se pak mohou skupiny těchto nevýznamových znaků mírně lišit, a to v závislosti na charakteru uchovávaných dat a podle požadovaných vlastností indexování a vyhledávání v dané oblasti UIS.

Extrahovaný token musí být převeden do vhodného formátu. Proto při ukládání tokenů dochází k jejich konverzi na malá písmena. Aby byla zachována možnost

vyhledávání i se zachováním diakritiky, je každý token ukládán kromě jeho základní podoby (tvaru bez diakritiky) i ve variantě s diakritikou. Indexy tak sice budou zabírat více místa, ale za to umožní rychlejší zpracování dotazovacích příkazů.

Jak vyplývá z obr. 4 v kapitole 5.1.1, tvary s diakritikou jsou ukládány do sloupce `TOKEN`, který je unikátní a token s diakritikou se tedy v seznamu tokenů může vyskytnout pouze jednou. Oproti tomu varianta tokenu s ořezanou diakritikou se ukládá do sloupce `TOKEN_ASCII`, který již unikátní není a ani být nemůže.

Pro vysvětlení si představme například slova „věčné“ a „věcně“. S diakritikou se jedná o dvě různá slova, unikátnost při ukládání tedy nebude porušena. Po jejich konverzi na tvar bez diakritiky však v obou případech dostaneme tvar „vecne“, proto jednoznačnost při ukládání tokenů bez diakritiky z principu nelze vyžadovat. Způsob ukládání dat do inverzního indexu názorně zachycuje obr. 5.



Obr. 5: Uložení dat v inverzním indexu

Při vyhledávání výrazů „věčné“, „věcně“ nebo „vecne“ v případě požadované nezávislosti na diakritice by tak vždy vyhověly oba dokumenty. V případě hledání tvarů s přesně zadanou diakritikou by při dotazu „věčně“ vyhověl jen dokument s ID=1, při dotazu „věcně“ jen dokument s ID=1 a při dotazu „vecne“ žádný z dokumentů.

5.1.3 Mechanismus indexování

Protože většina oblastí UIS, kde je žádoucí nasadit fulltextovou technologii, již běží v reálném provozu systému, je zapotřebí při indexování téměř vždy nejprve provést úvodní naindexování, a pak až zajistit průběžnou indexaci reflektující změny ve zdrojových datech.

Pro úvodní naindexování dané sady dokumentů je potřeba účasti lidského faktoru. Mezi hlavní úkony, které je třeba provést, patří vytvoření struktury inverzního indexu a spuštění funkce, která projde všechny stávající dokumenty a ke každému vloží indexovací údaje do inverzního indexu. Toto úvodní naindexování může být časově náročná operace, zejména v případě velkého objemu dat.

Aktualizace inverzních indexů už pak musí probíhat pravidelně a zcela automatizovaně. Z požadavků kladených na fulltextovou technologii vyplývá, že není

nezbytně nutné, aby indexování nového resp. modifikovaného dokumentu proběhlo bezprostředně po jeho vložení resp. modifikaci. Nicméně je žádoucí, aby k indexaci došlo co nejdříve. Pro tyto účely byl navržen mechanismus založený na průběžném sběru všech relevantních změn, podle kterých pak v vždy v určeném časovém intervalu proběhne aktualizace indexů.

Pro sběr údajů o změnách v dokumentech se nejvíce hodí databázové triggery (spouště), což jsou PL/SQL programové jednotky (podobně jako procedury), jejichž provedení je však vyvoláno určitou událostí v databázi (více viz např. [19]). Nad každou indexovanou tabulkou je tedy zapotřebí definovat standardizovaný trigger reagující na přidání nového resp. změnu stávajícího záznamu, který se má indexovat. V těle triggeru je pak vyhodnoceno, zda se má zaznamenat změna nad konkrétním záznamem, a pokud ano, zaznamená tuto změnu do speciální logovací tabulky `W_FI_ZMENY`. Struktura této tabulky je jednoduchá, tvoří ji sloupce `TABULKA`, `SLOUPEC` a `ZAZNAM`. Název tabulky, sloupce, ve kterém se nachází indexovaný záznam a jednoznačný identifikátor záznamu (hodnoty primárních klíčů ze zdrojových tabuleck, zpravidla `ID`) jsou již postačují údaje pro pozdější indexaci záznamu.

Samotnou indexaci, resp. aktualizaci indexu zajišťuje obslužná funkce, která vyčítá údaje ze změnové tabulky a pro každý záznam volá indexovací funkci. Tyto funkce jsou naprogramovány v jazyce Perl a jsou součástí modulu, který zapouzdřuje funkcionality spojené s fulltextovými technologiemi na aplikační vrstvě. Obslužná funkce je spouštěna periodicky každých 5 minut. Nově vložený nebo změněný dokument se tedy zaindexuje v nejhorším případě s pětiminutovým zpožděním. Pouze po tuto dobu není možné dokument použitím fulltextového vyhledávání nalézt. Nakonec po úspěšné indexaci záznamu se s ním související údaj ze změnové tabulky odstraní.

5.2 Realizace fulltextového vyhledávání

5.2.1 Základní fulltextové dotazy

Cílem operace vyhledávání je vždy nalezení jednoho nebo více dokumentů odpovídajících zadávanému dotazu. K tomu je zapotřebí získat sadu jednoznačných identifikátorů, v níž každému identifikátoru je přiřazen právě jeden dokument. Pokud tuto sadu získáme, splníme tak vlastně cíl vyhledávání a je dále jen otázkou zobrazovacích funkcí, jak bude výsledná sada prezentována uživateli. Soustřeďme tedy pozornost zejména na získávání vyhovující sady identifikátorů dokumentů.

Nad strukturou inverzního indexu popsanou v předchozí kapitole lze sestavit SQL příkazy, kterými jsme schopni takovou sadu jednoznačných identifikátorů snadno získat. Pro její získání navíc není vůbec potřeba pracovat s tabulkou obsahující samotná zdrojová data, která může být velmi rozsáhlá a jejíž přímé prohledávání by bylo značně pomalé, neefektivní a pro databázový stroj příliš zatěžující. Potřebnou sadu identifikátorů lze získat přirozeným spojením dvou tabulek tvořící inverzní index. Pokud bychom uvažovali strukturu indexu jako na obr. 4 v kapitole 5.1.1,

pak by se jednalo o spojení tabulek TAB_DATA_I a TAB_DATA_W. Jednoznačné identifikátory dokumentů (zpravidla hodnoty ID) lze získat ze sloupce ODKAZ bez nutnosti číst zdrojovou tabulku. Implementace jednotlivých operátorů pomocí SQL příkazů vycházejí z myšlenek uvedených v [16]. Následující příkaz SELECT názorně ukazuje, jak je možné získat sadu identifikátorů dokumentů, obsahujících slovo „informace“:

```
SELECT distinct odkaz FROM TAB_DATA_I, TAB_DATA_W
WHERE TAB_DATA_I.TOKEN = TAB_DATA_W.ID
    AND TAB_DATA_W.TOKEN = 'informace'
```

V příkladu byl použit operátor DISTINCT, který zajistí, že se ve výstupu neobjeví identifikátor stejného dokumentu víc než jednou. Pokud bychom totiž operátor DISTINCT nepoužili a některý dokument by slovo „informace“ obsahoval např. dvakrát, pak by se i ve výstupu identifikátor dokumentu objevil dvakrát, což není žádoucí.

Jinou možností, jak zajistit, aby se ve výstupní sadě dokument neobjevoval víc než jednou, je použití klauzule GROUP BY příkazu SELECT, jak ukazuje následující příklad:

```
SELECT odkaz, count(*) AS POCET
    FROM TAB_DATA_I, TAB_DATA_W
    WHERE TAB_DATA_I.TOKEN = TAB_DATA_W.ID
        AND TAB_DATA_W.TOKEN = 'informace'
    GROUP BY odkaz
    ORDER BY POCET
```

U tohoto způsobu eliminace duplicit ve výsledné sadě navíc získáváme přídavnou informaci o počtu výskytů hledaného slova v daném dokumentu. Výslednou sadu tak můžeme rovnou seřadit podle počtu výskytů (použitím klauzule ORDER BY), eventuelně může tento údaj posloužit jako součást složitějších výpočtů pro stanovení relevance dokumentu.

Citlivost vyhledávání na diakritiku

Způsob uložení dat v inverzním indexu umožňuje vyhledávání nezávislé na diakritice i vyhledávání s přesně zadanou diakritikou klíčů v dotazu. Tato problematika již byla částečně diskutována v kapitole 5.1.2 věnované způsobu ukládání dat v inverzního indexu. Nyní se podívejme jak jde popsaný způsob uložení využít ve vyhledávacích SQL dotazech.

Pokud se má hledat s přesně zadanou diakritikou, pak se z klíčů dotazu při jeho zpracování neodstraní diakritika a dojde k jejich porovnávání se sloupcem TAB_DATA_W.TOKEN. Tuto situaci znázorňují i výše uvedené příklady dotazů.

V případě vyhledávání nezávislého na diakritice se klíče z dotazu převedou na základní tvar (tedy na tvar bez diakritiky) a budou se porovnávat se sloupcem

TAB_DATA_W.TOKEN_ASCII, do kterého se tokeny ukládají rovněž ve tvarech bez diakritiky. Při hledání slova „informační“ nezávisle na diakritice by tedy SQL dotaz vypadal takto:

```
SELECT distinct odkaz FROM TAB_DATA_I, TAB_DATA_W
WHERE TAB_DATA_I.TOKEN = TAB_DATA_W.ID
    AND TAB_DATA_W.TOKEN_ASCII = 'informacni'
```

Takovému dotazu by vyhověl dokument obsahující slovo ve tvaru „informační“, „informacni“ a dokonce i dokument obsahující toto slovo s překlepem, např. „informačni“. Vyhledáváním nezávislým na diakritice lze tedy do jisté míry eliminovat i překlepy v dokumentech, případně i překlepy v dotazu. Tímto způsobem se eliminují překlepy způsobené nesprávnou diakritikou, nikoliv však překlepy způsobené chybějícím nebo naopak přebytečným znakem ve slově.

5.2.2 Logické vazby klíčů dotazu

V uživatelských fulltextových dotazech skládajících se z více klíčů je možné specifikovat mezi jednotlivými klíči určité vztahy. Základními z nich jsou logické vazby vyjádřitelné známými logickými spojkami AND a OR a dále unárním operátorem NOT. Podívejme se, jakým způsobem lze zkonstruovat databázové dotazy implementující tyto základní logické vazby mezi klíči. Jednotlivá řešení opět vychází z myšlenek uvedených v [16].

Představme si uživatelský dotaz například ve tvaru „**data OR informace**“. Databázový SELECT, který vybere dokumenty odpovídající takovému dotazu by vypadal následovně:

```
SELECT distinct odkaz FROM TAB_DATA_I, TAB_DATA_W
WHERE TAB_DATA_I.TOKEN = TAB_DATA_W.ID
    AND TAB_DATA_W.TOKEN IN ('data','informace')
```

Výše uvedeným dotazem bychom tedy získali sadu identifikátorů dokumentů, které obsahují buď slovo „data“ nebo slovo „informace“. V tomto případě je toho dosaženo využitím množinového operátoru IN jazyka SQL.

Pro realizaci logické vazby AND (nebo-li „a zároveň“) lze nasadit množinový operátor INTERSECT (průnik množin), kterým se spojí dílčí příkazy SELECT. Každý z těchto dílčích příkazů vybírá sadu (množinu) identifikátorů dokumentů vyhovujících vždy dílčímu klíči z dotazu. Aplikací operace průniku na tyto množiny pak obdržíme takové identifikátory dokumentů, které obsahují zároveň všechny klíče z dotazu. Například pro uživatelský dotaz „**informační AND systém**“, by mohl databázový dotaz mít tuto podobu:

```
SELECT distinct odkaz FROM TAB_DATA_I, TAB_DATA_W
WHERE TAB_DATA_I.TOKEN = TAB_DATA_W.ID
    AND TAB_DATA_W.TOKEN = 'informační'

INTERSECT

SELECT distinct odkaz FROM TAB_DATA_I, TAB_DATA_W
WHERE TAB_DATA_I.TOKEN = TAB_DATA_W.ID
    AND TAB_DATA_W.TOKEN = 'systém'
```

Základní dotaz realizující unární operátor NOT může pro svou konstrukci využít stejnojmenného operátoru jazyka SQL ve spojení s množinovým operátorem IN a vnořeným dotazem (subdotazem). SQL dotaz, který vybere dokumenty neobsahující slovo „data“, by tak mohl vypadat takto:

```
SELECT distinct odkaz FROM TAB_DATA_I
WHERE odkaz NOT IN
     ( SELECT odkaz FROM TAB_DATA_I, TAB_DATA_W
       WHERE TAB_DATA_I.TOKEN = TAB_DATA_W.ID
          AND TAB_DATA_W.TOKEN = 'data'  )
```

Tento přístup je založen na tom, že nejprve je nalezena množina dokumentů, které dané slovo obsahují, a poté je vybrán doplněk k této množině – tedy dokumenty neobsahující zadané slovo. Při konstrukci podmínky způsobem, kde by se přímo porovnával hledaný token operátorem nerovnosti, tedy `TAB_DATA_W.TOKEN <> 'data'` bychom narazili na problém, že ve výsledném seznamu dokumentů by se mohly objevit i takové, které slovo „data“ obsahují. Došlo by k tomu v případě, že dokument obsahuje i jiná slova než „data“ a vyhoví tak podmínce s operátorem nerovnosti. Tento problém řeší právě předchozí popisovaný přístup.

V praxi je ovšem použití jen samotného unárního operátoru NOT spíše výjimečné. Uplatňuje se až v kombinaci s některým z binárních operátorů, zejména pak v kombinaci s logickou spojkou AND. Podívejme se tedy, jak by se realizoval uživatelský dotaz „**informace AND NOT data**“:

```
SELECT distinct odkaz FROM TAB_DATA_I
WHERE TAB_DATA_I.TOKEN = TAB_DATA_W.ID
    AND TAB_DATA_W.TOKEN = 'informace'
    AND TAB_DATA_I.ODKAZ NOT IN
       ( SELECT odkaz FROM TAB_DATA_I, TAB_DATA_W
         WHERE TAB_DATA_I.TOKEN = TAB_DATA_W.ID
            AND TAB_DATA_W.TOKEN = 'data' )
```

Výchozím bodem je základní dotaz vybírající dokumenty obsahující slovo „informace“, ke kterému se přidává podmínka vybírající současně dokumenty neobsahující slovo „data“. Dochází tedy ke kombinaci dříve uvedených základních konstrukcí pro jednotlivé operátory.

Kombinováním základních konstrukcí operátorů je možné dále realizovat složitější uživatelské dotazy typu „systém AND (informace OR data)“. Toho se docílí dynamickým skládáním podmínek výsledného SQL dotazu, a to v závislosti na zadaném uživatelském dotazu. Uživatelský dotaz pro tento musí být vhodným způsobem zpracován do takové podoby, aby bylo možné vygenerovat odpovídající SQL dotaz. Zpracováním uživatelských dotazů se dále zabývá jedna z následujících kapitol.

5.2.3 Implementace dalších operátorů

Kromě základních operátorů, sloužících k vyjádření logických vazeb mezi klíči ve víceslovných dotazech, bylo dále jedním z požadavků umožnit v uživatelských dotazech použití některých speciálních fulltextových operátorů. Jedná se zejména o operátory NEAR a PHRASE a také funkcionalitu, umožňující nalézt jen záznamy tvořené pouze celým textem, který odpovídá zadanému dotazu.

Operátorem **NEAR** se vyjadřuje požadavek, že dva nebo více klíčů se má v prohledávaných textech vyskytovat *blízko* sebe. Pro algoritmizaci tohoto vágního pojmu je potřeba stanovit hraniční hodnotu, která vyjadřuje, kolik maximálně slov můžou být od sebe daná dvě slova v textu vzdálena, aby bylo možné prohlásit, že jsou blízko sebe. Nabízí se tuto hodnotu stanovit pevně v systému, případně je možné její zadání ponechat na uživateli. V UIS je implicitně tato hodnota nastavena na vzdálenost pěti slov. Následující příklad ukazuje konstrukci pro realizaci uživatelského dotazu „**informační NEAR systém**“:

```
SELECT distinct word1.odkaz
FROM (SELECT odkaz,pozice from TAB_DATA_I i, TAB_DATA_W w
      WHERE i.token=w.id
         AND w.TOKEN_ASCII = 'informacni') word1,
     (SELECT odkaz,pozice from TAB_DATA_I i, TAB_DATA_W w
      WHERE i.token=w.id
         AND w.TOKEN_ASCII = 'system') word2
WHERE  word1.odkaz = word2.odkaz
   AND (word2.pozice - word1.pozice) BETWEEN 0 and 5
```

Princip spočívá v tom, že pro každý klíč (jenž je v uživatelském dotazu operandem operátoru NEAR) se známým způsobem vytvoří seznam dokumentů jako datový zdroj do části FROM příkazu SELECT se strukturou (ODKAZ, POZICE). V příkladu jsou takto vzniklé dva zdroje pojmenovány pomocí aliasů *word1* resp. *word2*. Na datové zdroje je aplikována operace přirozeného spojení (`word1.odkaz = word2.odkaz`) a dále podmínka využívající údaj o pozici slova v dokumentu, která

zajistí výběr dokumentů vzdálených o maximálně pět slov včetně. Je přitom respektováno pořadí slov z uživatelského dotazu, tedy v tomto případě slovo „informační“ musí v textu předcházet slovu „systém“.

Operátor **PHRASE** umístěný mezi dvěma klíči říká, že uvedené klíče se mají v textu vyskytovat bezprostředně za sebou, tedy najít tvořit fázi. Tento operátor nachází uplatnění při vyhledávání dokumentů obsahujících ustálená spojení v daném jazyce, nebo při hledání přesného fragmentu textu v dokumentu. Princip implementace je podobný jako v případě operátoru NEAR. Rozdílný je pouze tvar poslední podmínky. Implementace uživatelský dotazu „**informační PHRASE systém**“ by tedy měla stejnou podobu jako předchozí uvedený příkaz SELECT, jen poslední podmínka realizující operátor PHRASE by měla tento tvar:

```
AND word1.pozice + 1 = word2.pozice
```

Touto podmínkou se vyjadřuje skutečnost, že rozdíl mezi pozicemi dvou slov v textu, která mají tvořit frázi, musí být roven právě jedné. Navíc je zde – podobně jako u operátoru NEAR – respektováno pořadí slov ve frázi, tedy druhé uváděné slovo musí mít pozici o jedničku vyšší než první a ne naopak.

Užitečnou funkcionalitou při vyhledávání v některých oblastech UIS je také možnost uživatele říct, aby položenému dotazu vyhověly jen takové textové záznamy, které kromě textu odpovídajícímu dotazu neobsahují žádný další text. Jedná se tedy o možnost „hledat jen celý přesně zadaný řetězec“. K implementaci této funkcionality je možné použít jeden ze dvou přístupů. První přístup spočívá v tom, že se příslušným způsobem vyhledají všechny dokumenty tak jako doposud a teprve na aplikační úrovni záznamy projdou filtrem. Tento způsob se ovšem jeví jako ne příliš efektivní.

Proto při implementaci do UIS byl zvolen jiný přístup, u kterého dochází k filtraci dokumentů už na databázové vrstvě. Lze toho docílit přidáním vhodné podmínky do části WHERE příkazu SELECT. Například dotaz vybírající záznamy obsahující pouze frázi „informační PHRASE systém“ by vypadal následovně:

```
SELECT distinct word1.odkaz
FROM (SELECT odkaz,pozice from TAB_DATA_I i, TAB_DATA_W w
      WHERE i.token=w.id
         AND w.TOKEN_ASCII = 'informacni') word1,
     (SELECT odkaz,pozice from TAB_DATA_I i, TAB_DATA_W w
      WHERE i.token=w.id
         AND w.TOKEN_ASCII = 'system') word2
WHERE  word1.odkaz = word2.odkaz
   AND word1.pozice + 1 = word2.pozice
   AND (SELECT max(pozice) from TAB_DATA_I
        where TAB_DATA_I.odkaz = word1.odkaz) = 2
```

Předposlední podmínka zajišťuje výběr dokumentů obsahujících požadovanou dvouslovnou frázi a poslední podmínka omezí výběr jen na takové dokumenty, které mají délku dvě slova. Tím je zajištěno, že se vyberou pouze textové záznamy ve tvaru „informační systém“ a žádné jiné. Pokud by frázi měla tvořit tři slova, pak by se vybíraly dokumenty o délce tří slov atd. Stejnou podmínku je možno přidat i k operátorům NEAR a AND. U operátoru OR (např. „informace OR data“) se vždy vybírají pouze jednoslovné textové záznamy. V případě kombinovaných dotazů se pak výpočet výsledného počtu slov pro použití v podmínce odvíjí od složitosti dotazu.

Jedním z pokročilejších fulltextových operátorů je operátor pravostranného rozšíření. Jeho prostřednictvím lze dosáhnout jedné z forem podřetězcového vyhledávání. K jeho implementaci postačí datová struktura navržená pro inverzní index v kombinaci s vhodným využitím lexikografického řazení řetězců v databázi. Databázový SELECT, implementující uživatelský dotaz „**systém**∗“, kterému vyhoví i dokumenty obsahující systém*y*, systém*em*, systém*ový* apod. by vypadal následovně:

```
SELECT distinct odkaz FROM TAB_DATA_I i, TAB_DATA_W w
WHERE i.token=w.id
    AND w.TOKEN_ASCII >= 'system'
    AND w.TOKEN_ASCII < 'systen'
```

Princip vychází z faktu [16], že v seznamu tokenů všechna slova začínající na *system* jsou v lexikografickém uspořádání mezi samotným řetězcem *system* a případným slovem začínajícím na *systen*. Dolní hranici pro vyhledávání získáme prostým odtržením hvězdičky z konce řetězce. Pro získání horní hranice dále extrahujeme ještě jeden znak a nahradíme jej znakem v abecedě následujícím. Pokud uvažujeme jednobajtové kódování řetězců, postačí nám přičíst k ASCII hodnotě posledního znaku jedničku (z písmene „m“ tímto způsobem získáme písmeno „n“). Pro korektní fungování je zapotřebí pracovat se sloupcem `TOKEN_ASCII`, kde jsou tokeny ukládání ve tvaru bez diakritiky.

Implementací dalšího pokročilého operátoru – operátoru stemmingu, se bude zabývat zvláštní kapitola pojednávající o zapojení nástroje ConText CZ. Tento operátor totiž nelze dost dobře implementovat prostými nástroji RDBMS jako v případě doposud zmiňovaných operátorů, ale pro jeho úspěšné využití je zapotřebí nasadit vhodný nástroj, kterým je právě produkt ConText CZ, jehož popisem se zabývala kapitola 4.3.

5.2.4 Zpracování uživatelského dotazu

Základní myšlenkou pro úspěšnou realizaci fulltextového dotazování je, že dotaz by měl být zpracován stejným tokenizačním algoritmem, jakým je zpracováván zdrojový text při procesu indexování [16]. Jen za splnění tohoto předpokladu je možné korektně realizovat porovnání podobnosti dotazu a dokumentů.

Při zpracování uživatelského dotazu je tedy možné (a také žádoucí) nasazení mechanismů použitých při indexování. Jedná se zejména o prvotní vyčištění textu od nepotřebných znakových sekvencí, extrakci tokenů a jejich konverzi do potřebného formátu (převod na malá písmena, případně odstranění diakritiky). Tyto mechanismy je dále potřeba vhodně rozšířit tak, aby umožnily detekci a zpracování fulltextových operátorů. Pokud dále pro indexování prohledávané oblasti byl použit stoplist, je zapotřebí, aby stejný stoplist byl aplikován na dotaz. Pokud se tedy v dotazu vyskytne slovo obsažené ve stoplistu, bude – stejně jako při indexování – ignorováno.

Podstatná odlišnost zpracování dotazu a indexovaného textu spočívá zejména v možnosti výskytu speciálních slov v dotazu, které reprezentují operátory. Ekvivalentně je možné místo slovního výpisu operátoru použít speciální znak (např. pro AND je to znak „&“). V takovém případě se jednoduše rozvine znak do slovní podoby operátoru (& → AND, | → OR, ; → NEAR apod.).

Další odlišností související s použitím unárních operátorů (např. „$systém“ nebo „systém∗“ je nutnost zachování nevýznamových znaků reprezentujících tyto operátory. Při indexování se ze slov odstraňují všechny nevýznamové znaky, kdežto při zpracování dotazu je nutno některé vyhrazené znaky u klíčů zachovat, aby bylo možné podle nich sestavit výsledný databázový SQL dotaz.

Toto všechno je zajištěno uvnitř funkce `search()` modulu `Fulltext.pm` (použití viz kapitola 5.6). Jejími stěžejními parametry jsou uživatelský dotaz a specifikace prohledávané oblasti. Funkce dále zajišťuje seskládání a provedení odpovídajícího databázového SQL dotazu a jeho výsledek vrací jako návratovou hodnotu ve formě odkazu na strukturu standardně používanou pro výsledky SQL dotazů jádrem UIS.

Jádrem pro konstrukci SQL fulltextových dotazů je funkce `base_query()`, která vygeneruje základní SQL dotaz na základě sady dodaných klíčů z uživatelského dotazu a binárního operátoru spojujícího všechny tyto klíče a dodaného rovněž z uživatelského dotazu (implicitně je to spojka AND, v případě jednoho klíče je binární operátor irelevantní).

Nejjednodušším případem pro zpracování jsou jednoslovné uživatelské dotazy, u nichž odpadá řešení binárních operátorů a jejich kombinací. Problém se tak zužuje jen na řešení unárních operátorů (stemming, pravostranné rozšíření, NOT apod.), jejichž řešením se zabývaly předchozí kapitoly.

V případě víceslovných uživatelských dotazů, lze opakovaným voláním (vždy s příslušnými klíči a jejich spojkou) dynamicky seskládat fulltextový SQL dotaz, jehož výsledkem je sada identifikátorů vyhovujících dokumentů. Unární operátory jsou řešeny podobně jako v případě jednoslovných dotazů vždy až na úrovni konstrukce příslušné podmínky implementující daný operátor.

S klíčem obsahujícím znak unárního operátoru se tedy zachází jako s celkem a k odstranění znaku operátoru dojde až v okamžiku vytváření vlastní podmínky v části WHERE příkazu SELECT. Takto vygenerovaný dotaz je následně použit jako datový zdroj výsledného SQL dotazu, který již dodává aplikaci všechny potřebné údaje o jednotlivých nalezených dokumentech, aby bylo možné nalezené výsledky

odpovídajícím způsobem uživateli prezentovat. Příklad kompletního vyhledávacího SQL dotazu je obsažen v příloze D.

5.2.5 Optimalizace fulltextových dotazů

Při práci s vyhledáváním je kladen velký důraz na rychlou odezvu. Jedná se zřejmě o nejdůležitější požadavek z pohledu běžného uživatele systému. Získávání informací musí být rychlé a přesné, jinak jejich hodnotu není možné efektivně proměnit v užitek. Doposud jsem se zabývali zejména dosažením požadované funkčnosti vyhledávání, tak aby uživatel mohl svůj vyhledávací požadavek co nejpřesněji formulovat zadáním uživatelského fulltextového dotazu. Zaměřme teď naši pozornost na rychlost prováděných dotazů a na jejich optimalizaci.

Výkonově nejnáročnější fází vyhledávání jsou bezesporu operace na databázové vrstvě. Klíčové je zejména provádění SQL dotazů, které zajišťují vlastní vyhledání a výběr dat. Pro maximalizaci rychlosti jejich provedení je zapotřebí nejprve identifikovat nejčastější typy dotazů a jejich omezovacích podmínek s ohledem na strukturu inverzního indexu. Vybrané dotazy je dále vhodné analyzovat pomocí prostředků, které nabízí databázový systém Oracle. Jedná se zejména o rozbor prováděcího plánu, jenž je určován optimalizátorem systému (tzv. cost based optimalizátor, viz kapitola 2.4.1), dále pak využití různých – zpravidla grafických – průvodců a poradců, kteří dávají doporučení pro optimalizaci dotazu. V neposlední řadě je vhodné možnosti pro zvýšení výkonu nastudovat z dokumentace k systému Oracle a dalších zdrojů, např. [23] nebo [6], a následně je vhodně uplatnit. Nutným předpokladem pro ladění SQL dotazů je také dostatečně reprezentativní vzorek indexovaných dat, ideálním případem je inverzní index naplněný na základě reálných dat konkrétní indexované oblasti UIS.

Jedním z kroků prováděných databázovým systémem při parsování SQL dotazu je jeho přepis do standardizované podoby. Pro nás nejzajímavější je přepis podmínky s operátorem IN na ekvivalentní sadu podmínek spojených pomocí logické spojky OR. Proto všude při dynamické konstrukci dotazů je vždy použit tento druhý zmiňovaný přístup, který je sice náročnější na zápis, nicméně přináší jistou výkonovou úsporu při parsování dotazu, protože není již nutný přepis. Úspora se projeví zejména v případech, kdy se aplikuje lexikální analýza slova (viz následující kapitola), která může vést až na desítky různých tvarů slova a tato slova je zapotřebí zahrnout do databázového dotazu.

Při zefektivňování SQL dotazů hrají patrně nejvýznamnější roli databázové indexy. Vhodně vytvořený index může provedení vyhledávacího dotazu urychlit až o několik řádů. Při vytváření indexů je zapotřebí dodržovat některá pravidla zmiňovaná v kapitole 2.4 věnované indexům a jejich vlastnostem. Základní podmínkou, která je používána ve všech fulltextových dotazech je porovnání klíče z dotazu s hodnotou ve sloupci `TOKEN` resp. `TOKEN_ASCII` z inverzního indexu. Nad těmito sloupci je žádoucí definovat indexy pro rychlé vyhledání záznamů s tokeny. V případě sloupce `TOKEN` je index automaticky vytvářen už při definici jeho unikátnosti,

protože toto omezení je v databázovém systému Oracle realizováno právě pomocí unikátního indexu nad sloupcem. Pro sloupec `TOKEN_ASCII` je index (neunikátní) potřeba explicitně vytvořit.

U všech fulltextových konstrukcí dotazů se dále vyskytuje operace spojení tabulek inverzního indexu přes spojovací podmínku `TAB_DATA_I.TOKEN = TAB_DATA_W.ID`. Sloupec `ID` je primárním klíčem a index se pro něj opět zakládá systémem Oracle automaticky. Pro sloupec `TAB_DATA_I.TOKEN` je potřeba index dodatečně založit. Vzhledem k faktu, že je vždy potřeba získat identifikátory dokumentů, které jsou ukládány ve sloupci `TAB_DATA_I.ODKAZ`, je vhodné vytvořit místo samostatných indexů jeden kompozitní index nad oběma sloupci v pořadí (TOKEN, ODKAZ). Hodnotu identifikátoru dokumentu pro vyhovující token je tak možné zjišťovat přímo z indexu bez nutnosti čtení tabulky.

Kompozitní index lze s výhodou využít i pro případné dotazy pracující pouze se sloupem `TAB_DATA_I.TOKEN`, ovšem pro sloupec `TAB_DATA_I.ODKAZ` již použití kompozitního indexu tolik efektivní není. Výběr dat podle tohoto sloupce přitom může nastávat relativně často (viz například přídavná podmínka pro vyhledání přesného řetězce). Proto je dobré nad sloupcem `TAB_DATA_I.ODKAZ` také založit jednoduchý index. Výše zmiňované indexy najdou své uplatnění zejména při nárůstu dat v tabulkách inverzního indexu, tedy typicky po úvodním naindexování dané oblasti. V takovém případě je velmi pravděpodobné, že optimalizátor při určování prováděcího plánu příslušný index využije a provedení databázového dotazu se tak výrazně urychlí.

Další optimalizace souvisí spíše s procesem indexování než dotazování a týká se databázových omezení (tzv. *constraintů*) pro referenční integritu v tabulkách vytvářejících inverzní index. Při manipulaci s daty v inverzním indexu se musí vždy nejprve zkontrolovat, zda by daná operace nezpůsobila porušení některého constraintu, což může znamenat určité zpomalení operace[5]. Proto constrainty na databázové úrovni byly odstraněny a referenční integritu si hlídá samotný indexovací proces, jehož funkce jsou zapouzdřeny do modulu a manipulace s daty je prováděna výhradně přes tyto „obslužné“ funkce.

5.3 Zapojení nástroje Context CZ

Jednou z vysoce sofistikovaných funkcí, kterou lze běžnými prostředky relačního databázového systému odpovídajícím způsobem implementovat jen velmi těžko, je využití lexikální analýzy slov při vyhledávání. Vyplývá to zejména z faktu, že čeština je – narozdíl třeba od angličtiny – jazykem velmi bohatým a ohebným. Pro aplikaci této funkcionality při hledání v českých textech je tak zapotřebí zapojit hotový

[5]V databázovém systému Oracle jsou při zpracovávání příkazu generovány tzv. rekurzivní dotazy, které slouží pro ověření správnosti příkazu a integrity dat proti datovému slovníku. Pokud by příkaz měl způsobit porušení definované integrity v databázi, zadavatel příkazu je na toto upozorněn – je vyvolána výjimka a příkaz se neprovede. Jeden uživatelský dotaz může vyvolat obecně n rekurzivních dotazů.

softwarový produkt s vestavěnými komplexními algoritmy a daty, který je schopen lexikální analýzu daného výrazu zajistit.

Nástrojem, který poskytuje funkcionalitu lexikální analýzy a lze jej relativně snadno zapojit, je produkt ConText CZ, jehož vlastnosti byly podrobně popsány v kapitole 4.3. Nyní se podívejme, jakým způsobem je nástroj využit při zavádění fulltextových technologií do UIS.

5.3.1 Lexikální analýza

Hlavní funkcí nástroje ConText CZ je zajistit pro české tvary hledaných klíčů jejich lexikální analýzu, tedy vygenerovat všechny ostatní české tvary daného klíče. Tak například pro slovo „dům“ by jeho další (vyskloňované) tvary byly: *domů, domům, domu, domy, dům, domě, domama, dome, domech* a *domem*. Při vyhledávání záznamů uživatelským dotazem ve tvaru „dům“ se pak musí zajistit, aby dotazu vyhověly nejen dokumenty obsahující samotné slovo „dům“, ale také dokumenty obsahující některý z jeho vyskloňovaných tvarů.

Pro nasazení lexikální analýzy a vygenerování všech českých tvarů daného slova lze použít následující SQL dotaz:

```
SELECT cz('$(dům)','A_FULLTEXT_FINDX',0,0,1,0) from dual
```

Stěžejní je zde databázová fukce `cz()`, která je právě součástí balíku ConText CZ. Prvním parametrem je uživatelský dotaz (v tomto případě jednoduchý dotaz „dům“), který je obalen operátorem STEM (`$`). Pro svoji uspokojivou činnost potřebuje mít funkce vytvořen index typu CONTEXT (viz kapitola 4.2.2). Název tohoto indexu je pak druhým parametrem funkce. Zbývajícími čtyřmi parametry je možné do jisté míry ovlivňovat výsledné chování funkce. Prvním z nich je příznak, zda u neznámých slov zachovat operátor STEM (lze využít v případě, že dotaz je zpracováván balíkem Oracle Text), druhým je příznak, zda provést filtrovaní vygenerovaných slov podle slov vyskytujících se v příslušném fulltextovém indexu (opět lze využít ve spojení s Oracle Text), třetím parametrem říkáme, zda dotaz obsahuje nebo neobsahuje diakritiku (0 - dotaz musí obsahovat správnou diakritiku, 1 - dotaz může, ale nemusí obsahovat diakritiku, 2 - dotaz může, ale nemusí obsahovat diakritiku, pokud ji obsahuje, bude před zpracováním dotazu odstraněna), čtvrtý z této série parametrů říká, zda systém generuje tvary hledaných slov s diakritikou nebo bez (0 - generují se pouze tvary s diakritikou, 1 - generují se pouze tvary bez diakritiky, 2 - hledají se tvary s diakritikou i bez diakritiky).

Výsledkem tohoto dotazu je pak řetězec obsahující všechny požadované tvary slov ve formátu, který je vhodný pro jeho další automatizované softwarové zpracování:

```
({domů}={domům}={domu}={domy}={dům}={domě}=
 {domama}={dome}={domech}={domem})
```

Z takto získaného řetězce pak lze snadno extrahovat jednotlivé tvary slova. Všechny tyto tvary se následně využijí pro seskládání vlastního databázového dotazu tak, aby se vyhledaly i dokumenty obsahující všechny vyskloňované resp. vyčasované tvary daného slova. Konstrukce dotazu implementujícího hledaný výraz „**$dům**“ tak vypadá následovně:

```
SELECT distinct odkaz FROM TAB_DATA_I, TAB_DATA_W
WHERE TAB_DATA_I.TOKEN = TAB_DATA_W.ID
    AND ( TAB_DATA_W.TOKEN = 'dům' OR
          TAB_DATA_W.TOKEN = 'domu' OR
          TAB_DATA_W.TOKEN = 'domů' OR
          TAB_DATA_W.TOKEN = 'domům' OR
          TAB_DATA_W.TOKEN = 'domy' OR
          TAB_DATA_W.TOKEN = 'domě' OR
          TAB_DATA_W.TOKEN = 'domama' OR
          TAB_DATA_W.TOKEN = 'dome' OR
          TAB_DATA_W.TOKEN = 'domech' OR
          TAB_DATA_W.TOKEN = 'domem' )
```

Při konstrukci dotazu se využívá jednoduchého spojení podmínek tvořených porovnáním sl. `TOKEN` inverzního indexu s jednotlivými tvary slova spojkou OR. V tomto případě se jedná o vyhledávání závislé na diakritice. V případě vyhledávání nezávislém na diakritice by se pro tvorbu jednotlivých podmínek použil nám již známý sloupec `TOKEN_ASCII`, přičemž už při lexikální analýze slova by se příslušným parametrem funkce `cz()` nastavilo jen generování tvarů bez diakritiky (viz výše).

5.3.2 Stoplist

V každé sadě prohledávaných textových dokumentů existují slova, která se vyskytují ve velké většině všech dokumentů, v extrémním případě pak ve všech dokumentech. Taková slova mají při vyhledávání velmi malou rozlišovací schopnost, protože pokud bychom chtěli na jejich základě vyhledávat, obdrželi bychom jako výsledkovou sadu většinu prohledávaných dokumentů nebo dokonce všechny. Žádoucí přitom je, aby výsledková sada obsahovala pouze malé procento všech dokumentů.

Obecně se za taková slova považují ta, která jsou v daném jazyce velmi často používána při stavbě vět a souvětí. Je proto u nich vysoce pravděpodobné, že se vyskytnou ve velké většině daných textových záznamů. V češtině mezi tato slova obvykle řadíme slovní druhy jako předložky nebo spojky a dále pak i některá slovesa, jako například „být“.

Protože taková slova nemají ve vztahu k dokumentu příliš velkou vypovídací schopnost, je také zbytečné taková slova indexovat. Při procesu indexování se tak pracuje se speciálním seznamem slov, který říká, jaká slova mají být při indexování ignorována. Tento seznam se nazývá *stoplist*. Stejný stoplist jako při indexování

dané oblasti je pak uplatněn na uživatelský dotaz – tedy slova z dotazu, která jsou obsažena ve stoplistu jsou ignorována a nevyhledává se podle nich.

Obecný stoplist českých a anglických slov je i součástí produktu ConText CZ. Úplný seznam slov, které stoplist implicitně obsahuje, je uveden v příloze A. Je také umožněno přidávat do něj další slova podle potřeby.

Pro účely implementace fulltextového vyhledávání a indexování do UIS byl tento seznam přenesen z databázové vrstvy na vrstvu aplikační. Je to zejména kvůli urychlení práce s ním díky faktu, že indexovací a vyhledávací algoritmy jsou rovněž realizovány na úrovni aplikační vrstvy.

U některých specifických oblastí UIS (např. při vyhledávání překladových řetězců) je však použití stoplistu nežádoucí. V těchto případech je zapotřebí indexovat všechna slova, aby bylo možné řetězce vyhledat v jejich přesném znění, a to včetně předložek, spojek apod. V definici fulltextového indexu bylo proto implementována také volba, zda se pro danou indexovanou oblast má stoplist uplatnit či nikoliv.

5.4 Ohodnocovací funkce

Pro stanovení relevance nalezeného dokumentu byla vyvinuta ohodnocovací funkce `SCORE`, která zjišťuje míru podobnosti obsahu dokumentu se zadaným vyhledávacím dotazem. Čím vyšší je vrácené číslo, tím větší je podobnost dokumentu a dotazu. Funkce je realizována na databázové vrstvě v jazyce PL/SQL, aby ji bylo možné využít ve vyhledávacím SQL dotazu, např. pro řazení. Vstupními parametry jsou ID nalezeného dokumentu a sada identifikátorů tokenů z dotazu, která se stanovuje již na aplikační vrstvě při parsování dotazu a uplatní se při konstrukci vyhledávacího SQL dotazu. V těle funkce se pak s využitím poznatků a vztahů uvedených v kapitole 2.7 provede porovnání dvou vektorů, z nichž jeden je tvořen sadou identifikátorů tokenů dokumentu a druhý sadou identifikátorů tokenů dotazu.

Tokeny dokumentu můžou mít stanoveny různou váhu, která je při výpočtu podobnosti také zohledněna. Pro určení vah tokenů je nutno rozšířit struktury inverzního indexu o údaje vyplývající ze vztahů v kapitole 2.7.2, tedy o inverzní frekvenci tokenu (IDF) a normalizovanou v daném dokumentu (NTF). Výsledná váha tokenu je pak součinem těchto hodnot.

Strukturu tabulky `TAB_DATA_W` je tak nutné rozšířit o sloupce `DF` a `IDF`. Ve sloupci `DF` je zapotřebí udržovat aktuální hodnotu, říkající, v kolika dokumentech se daný term vyskytuje. Z této hodnoty lze dopočítat výslednou inverzní frekvenci a pro rychlost tyto hodnoty předpočítané ukládat do sloupce `IDF`. Do indexovacího mechanismu dále musí být doplněna údržba hodnot ve sloupci `DF` při indexování každého dokumentu. Protože výpočet hodnot IDF je závislý na celkovém počtu indexovaných dokumentů, nelze z výkonových důvodů tuto hodnotu udržovat stoprocentně aktuální indexovacím mechanismem při indexování každého nového či modifikovaného dokumentu. Přepočítávání těchto hodnot je výhodnější realizovat speciální procedurou, spouštěnou pravidelně v určitém časovém intervalu podle předpokládané četnosti změn v indexovaných datech.

Tabulku TAB_DATA_I inverzního indexu je potřeba rozšířit o sloupce TF a NTF. Ve sloupci TF je udržována frekvence výskytu daného tokenu v daném dokumentu. Do sloupce NTF je pak ukládána její normalizovaná hodnota, jejíž použití je výhodnější, protože eliminuje bezdůvodné zvýhodňování dlouhých dokumentů před krátkými. Aktuálnost těchto hodnot musí rovněž zajistit indexovací mechanismus.

5.5 Podpora prezentace nalezených výsledků

Nedílnou součástí procesu vyhledávání textových dokumentů je i jejich prezentace, tedy vhodné zobrazení nalezené sady výsledků. Vzhledem k časté rozsáhlosti daných textových záznamů je zapotřebí nabídnout prostředky, které umožní rychlou a snadnou orientaci v těchto výstupech. Ve fulltextových technologiích je pro tyto účely často využívána funkce zvýrazňování hledaných výrazů v nalezeném textu a dále funkce pro vytváření zkráceného náhledu na dlouhé texty. Při zavádění do UIS je rovněž žádoucí, aby výstupy po grafické i funkční stránce korespondovaly se stylem používaným v UIS.

5.5.1 Zvýrazňování klíčů v textech

Zvýrazňování slov v textu bývá často označován termínem *highlighting*. Zpravidla se jedná o vytučnění hledaných výrazů z dotazu v textu, který tomuto dotazu vyhoví. Na aplikační úrovni byla realizována funkce, jejímiž parametry jsou klíč, který se má zvýraznit (nebo i více klíčů), a text, ve kterém má být zvýraznění uplatněno. Návratovou hodnotou funkce je pak tento text vhodně označkovaný, tak aby v něm při zobrazení na webu byly tučně zvýrazněny příslušné klíče.

Základem této funkce je naplnění offsetu v podobě asociativního pole, kde klíčem je absolutní pozice začátku zvýrazňovaného slova a hodnotou je jeho délka. Podle takto zjištěného offsetu je možné text jednoduše označkovat a docílit tak zvýraznění potřebných slov v textu.

Pro naplnění offsetu je využívána standardní funkce jazyka perl `index()`, jejíž parametry jsou *řetězec*, *podřetězec* a nepovinný parametr *pozice*. Funkce vyhledá v řetězci zadaný podřetězec a vrátí pozici, kde se nachází V případě neúspěšného nalezení je vrácena hodnota o jedna menší než je index prvního prvku (implicitně je to hodnota -1) [2]. Vhodným využitím této standardní perlovské funkce v cyklu je možné pro každé zvýrazňované slovo text projít a naplnit offset potřebnými hodnotami.

Funkce pro highlighting musí dále brát v potaz i případný lexikální rozvoj jednotlivých zvýrazňovaných slov. Pokud bychom chtěli v daném textu zvýraznit např. slovo „systém“ a chtěli bychom přitom uplatnit lexikální analýzu (tedy zvýraznit všechny tvary tohoto slova), funkce zajistí, že pokud se v textu vyskytne např. tvar „systémech“, tak se nezvýrazní jen **systém**ech, ale celý tvar slova **systémech**.

Tento problém je řešen již na úrovni zjišťování offsetu. Ukládá se do něj vždy pozice a délka nejdelšího možného zvýraznitelného slova. Přitom se dbá na to aby

jednotlivé záznamy v offsetu nevedly při zvýraznění ke kolizi, tedy aby se offsetem vymezená slova nepřekrývala.

5.5.2 Náhled na dlouhé texty

Pro rychlou orientaci v delších textových záznamech je žádoucí odpovídajícím způsobem záznamy zkrátit a přispět tak ke zpřehlednění zobrazovaného výstupu. Také implementovaná technologie podporuje tvorbu náhledů na dlouhé texty. Jedná se o funkci, která má základní parametry stejné jako výše zmiňovaná zvýrazňovací funkce, ale na výstupu vrací vhodně zkrácený text, opět se zvýrazněnými klíči. Kromě základních parametrů je dále možné dalšími doplňujícími parametry ovlivnit výsledný textový náhled.

Funkce vychází ze stejného jádra jako zvýrazňovací funkce. Základem je opět naplnění offsetu definujícího pozice a délky jednotlivých zvýrazňovaných výrazů. Na základě těchto informací je pak snahou vybrat takové úseky textu, které nejlépe charakterizují text vzhledem k výrazům v dotazu. Kritériem je zejména hustota a vzdálenost zvýrazňovaných slov. Čím větší hustota slov a čím více jsou slova blízko sebe, tím je větší pravděpodobnost, že takový úsek textu dobře vystihne jeho obsah (resp. tematickou souvislost mezi dotazem a textem). Je také žádoucí do výsledného náhledu zahrnout pokud možno všechny klíče použité v dotazu. Toto pravidlo nemusí být ovšem vždy bezezbytku dodrženo, zejména s ohledem na požadavek co nejkratšího náhledu. Uživatel si tak velmi rychle může udělat obrázek o obsahu textu a nemusí přitom celé texty procházet.

Předáním dalších nepovinných parametrů je pak možné částečně zasáhnout do chování funkce. Takto je možné stanovit maximální délku výsledného náhledu nebo délku úseku textu mezi zvýrazňovanými slovy, který lze zachovat celý (při překročení této délky dochází ke zkrácení úseku pomocí řetězce „ . . . “).

5.6 Integrace do aplikací UIS

Při implementaci podpory prezentace nalezených výsledků je nutné dbát na dodržení stylu a prvků používaných v UIS a zachovat tak jednotnost prostředí, ve kterém se uživatel při práci se systémem pohybuje. Zároveň však je snahou zanést prvky, na které jsou uživatelé zvyklí z jiných vyhledávacích prostředí, zejména pak webových. Tím lze výrazně přispět k tomu, že uživatelé můžou bez zdlouhavého učení funkcionalitu vyhledávání ihned začít používat, a to včetně všech jejich pokročilých funkcí.

Aby tento komfort bylo možné nabídnout koncovým uživatelům, je potřeba umožnit aplikačním programátorům jednoduše funkcionalitu vyhledávání zabudovat do příslušné aplikace. Přitom je důležité nechat při vývoji aplikace programátorovi zcela volný prostor tak, aby nebyla narušena funkčnost nebo konzistence aplikace. Všechny potřebné metody pro práci s fulltextovým vyhledávání jsou proto zapouzdřeny do perlovského modulu `Fulltext.pm`. Všechny jeho funkce lze tak vy-

užít opakovaně v různých aplikacích UIS. Následuje ukázka zdrojového kódu, který demonstruje jednoduché použití modulu při nasazení fulltextového vyhledávání do aplikace UIS:

```
use UIS;
use Fulltext;

my $q = new UIS;
my $f = new Fulltext -UIS => $q;
 ...
my $set = $f->search( -oblast => 'preklady',
                      -dotaz => $dotaz,
                      -radit_vzestupne => 1,
                      -rozvoj => 1
                     );

foreach (@$set) {
    $q->add( $f->hl(-text => $_->{PREKLAD}) );
}
```

Jedná se konkrétně o ukázku vyhledávání v překladových řetězcích, které jsou uloženy ve sloupci `PREKLAD` v tabulce s překlady. První čtyři řádky kódu zajistí nezbytné zavedení a inicializaci modulů, a to modulu jádra UIS (objekt `$q`) a modulu zapouzdřujícího funkce pro fulltextové technologie (objekt `$f`). Následoval by kód implementující kompletní aplikaci pro práci s překladovými řetězci, který pro nás v této chvíli není příliš zajímavý.

Zajímavá je až ta část kódu, která implementuje fulltextové vyhledávání. Slouží pro to funkce `search()`, dostupná jako metoda objektu `$f`. Jejími základními parametry jsou specifikování oblasti, která se má prohledávat (v tomto případě identifikátor oblasti 'preklady') a uživatelský dotaz, reprezentovaný proměnnou `$dotaz`. Naplnění této proměnné musí zajistit programátor aplikace. Kromě těchto dvou povinných parametrů je možné funkci předat další volitelné parametry upřesňující vyhledávání. Tyto parametry se můžou lišit podle toho, jaká oblast je prohledávána. V našem případě je použit parametr, určující, že výsledky se mají řadit od nejkratšího překladu po nejdelší, což je funkcionalita implementovaná speciálně pro vyhledávání v překladech. Další nepovinný parametr říká, že na všechna slova v dotazu se má aplikovat lexikální rozvoj. Je to tedy ekvivalent situace, kdy před každým klíčem v dotazu je uveden operátor STEM (`$`) zajišťující lexikální rozvoj příslušného klíče. V případě více klíčů v dotazu však může být ruční vkládání tohoto operátoru před každý klíč pro uživatele zdlouhavé. Proto je k dispozici parametr, zajišťující automaticky rozvoj všech slov z dotazu a tento parametr lze využít – narozdíl od předchozího – při hledání v kterékoliv z oblastí.

Návratovou hodnotou funkce `search()` je výsledková sada s nalezenými překlady v podobě odkazu na pole, který se uloží do proměnné `$set`. Počet prvků

v poli odpovídá počtu vyhovujících řádků z tabulky s překlady. Prvky pole jsou reprezentovány opět odkazem, tentokrát však na asociativní pole, kde klíče tvoří názvy sloupců z prohledávané databázové tabulky a hodnoty jsou obsahy těchto sloupců. Každý prvek pole, na něž ukazuje proměnná `$set`, tak vlastně představuje řádek tabulky vyhovující uživatelskému dotazu. Toto pole lze pak jednoduše procházet v cyklu a obsahy sloupců v odpovídající podobě zobrazit uživateli (v našem zjednodušeném příkladě ze zobrazuje pouze obsah nalezeného překladu bez jakéhokoliv formátování).

Při zobrazování výsledků v cyklu je ještě ukázána možnost využití funkce pro zvýraznění hledaných slov v každém vyhovujícím překladu. Toto zajistí funkce `hl()`, opět dostupná přes objekt `$f`. Funkci stačí předat jediný parametr, a to aktuálně zobrazovaný překlad. Na úrovni modulu je zajištěno, že funkce si seznam klíčů, které je potřeba v textu zvýraznit, zjistí sama, a to díky naplnění speciální struktury již při zpracování dotazu v metodě `search()`. Výsledek zvýrazňovací funkce se pak může přímo předat některé ze zobrazovacích funkcí jádra UIS, v tomto případě funkci `add()`. Analogicky lze místo zvýrazňovací funkce `hl()` použít také funkci `nahled()` pro vytváření zkrácených náhledů na text. Ukázky vyhledávacích aplikacích jsou k nahlédnutí v přílohách B a C.

5.6.1 Nasazení do Tematického vyhledávání

Do portálové vyhledávací aplikace Tematické vyhledávání se rovněž počítá s použitím fulltextové technologie. Mezi volitelně prohledávané oblasti přibude zejména prohledávání blogů, diskuzí a také možnost prohledávání obsahů dokumentů v dokumentovém serveru. Pro fulltextové vyhledávání je dále ve vývoje rozšířené rozhraní umožňující uživatelům zadávat fulltextový dotaz strukturovaným způsobem.

U takového způsobu zadávání není ze strany uživatelů nutná znalost syntaxe fulltextových výrazů a operátorů. Pomocí vhodných prvků aplikace lze i složitější požadavky vyjádřit rychle a jednoduše.

Jisté potíže zde působí fakt, že fulltextové dotazy využívající speciální operátory nelze použít pro vyhledávání v oblastech indexovaných původním způsobem. Řešením je buďto dovolit zadávat pokročilé fulltextové dotazy jenom při vyhledávání v oblastech indexovaných novou technologií, např. v oddělené záložce, nebo vybrané oblasti dodatečně indexovat s využitím nově implementované fulltextové technologie. Původní indexování však musí být zachováno kvůli kompatibilitě s ostatními aplikacemi využívající dohledávání subjektů systému s pomocí klasického indexování.

6 Diskuze

Následující kapitola se bude zabývat rozborem dosažených vlastností implementované technologie vzhledem k získávání informací z databáze UIS, praktickým přínosem pro běžné uživatele systému, srovnáním možností vyhledávání před nasazením nové technologie a po jeho zavedení, dále pak porovnáním vyvinutého řešení s alternativními nástroji z různých úhlů pohledu. Nastiňuje také možnosti dalšího rozšiřování a vývoje technologie.

6.1 Vlastnosti vyvinuté technologie

Technologie vyvinutá vlastními silami a nasazená do provozu UIS nabízí i přes svůj jednoduchý myšlenkový základ poměrně široké možnosti pro vyhledávání v textových datech. Nespornou výhodou této technologie je zachování plné kontroly nad celým komplexním procesem, který zahrnuje přípravu a indexování dat, vyhledávání a prezentaci informací. Do všech fází procesu je možné odpovídajícím způsobem zasáhnout a technologii je tak možné bezezbytku a relativně jednoduše přizpůsobit potřebám UIS. Protože informační systém stále prochází velmi dynamickým vývojem, je velmi výhodná pružnost technologie, kterou lze čistě vlastním vývojem modifikovat tak, aby reflektovala aktuální požadavky.

Důležitým aspektem je také možnost plné infiltrace technologie do prostředí UIS, včetně respektování propracovaného právního systému. Při vývoji nových nebo úpravách stávajících aplikací lze z pohledu aplikačního programátora snadno funkcionalitu fulltextového vyhledávání zapojit a dovolit tak uživateli aplikace využívat všechny možnosti implementované technologie.

6.2 Uživatelský přínos

Uživatelské použití technologie, spočívající ve vyjádření vyhledávacího požadavku pomocí fulltextového dotazu, je intuitivní a je koncipováno tak, že nevyžaduje zdlouhavé studium ani hluboké speciální znalosti. Pro zdatnější uživatele – disponující alespoň základními informatickými znalostmi – je však zachována i možnost zadávání „expertních“ dotazů využívajících speciální fulltextové operátory pro stanovení vztahů mezi klíči ve víceslovném dotazu. Uživatelům systému se tak do rukou dostává silný nástroj pro získávání informací z rozsáhlé databázové základny systému, a to s poměrně jednoduchým rozhraním. Implementovaná technologie disponuje paletou operátorů a funkcí, které umožňují plnohodnotné fulltextové vyhledávání.

6.2.1 Možnosti vyhledávání před a po zavedení technologie

Před zavedením fulltextové technologie bylo možné v systému vyhledávat objekty jednoduchým zadáním klíče (tedy charakteristického údaje pro požadovaný objekt), který je ve stejné podobě ukládán v indexovací struktuře pro objekt. Výsledkem je vždy sada vyhovujících objektů, které lze použít při k další práci s informačním

systémem. Jelikož charakteristické údaje lze indexovat i podřetězcově a lze k sobě zkombinovat i více charakteristických údajů (např. zřetězení jména a příjmení člověka dovoluje nalézt osobu zadáním jakéhokoliv podřetězce ze jména či příjmení) je tato možnost vyhledávání velmi užitečná. Výhodná je zejména při vyhledávání objektů charakterizovaných krátkými řetězcovými případně číselnými údaji na základě zadání velmi jednoduchého klíče.

Tato původní funkcionalita je s úspěchem využívána jednak v aplikacích, jejichž primárním účelem není vyhledávání, ale v určitém okamžiku je v nich zapotřebí dohledat určitý objekt a dále s ním v aplikaci pracovat, ale je také základem čistě vyhledávacích aplikací (vyhledávání lidí, předmětů, pracovišť apod.) včetně portálového vyhledávacího řešení v podobě aplikace Tematické vyhledávání.

Dosavadní používaný způsob však nedovoluje prohledávání rozsáhlejších textových záznamů (dovoluje jejich vyhledání pouze na základě jejich metadat). Také možnosti zadávání vyhledávacího dotazu jsou poměrně strohé, což souvisí s jednoduchostí a malým rozsahem indexovaných dat.

Nově zavedená fulltextová technologie vhodným způsobem doplňuje stávající vyhledávácí mechanismy a rozšiřuje tak možnosti vyhledávání a získávání informací potenciálně do všech oblastí systému. Pomocí nové technologie lze prohledávat i rozsáhlejší sady textových záznamů, u kterých se prohledává i jejich obsah, což doposud v systému citelně chybělo. Tomu jsou také přizpůsobeny možnosti zadávání dotazů, tak aby uživatel mohl přesněji vyjádřit, co hledá.

Uživateli se dále dostává možnost využít při vyhledávání i velmi vyspělých funkcí, kterými technologie disponuje (např. skloňování a časování slov nebo řazení výsledků dle relevance) Použití těchto funkcí může významně zefektivnit proces získávání informací. Efektivnímu získávání informací napomáhá i způsob prezentace nalezených výsledků využívající speciální funkce pro zvýrazňování hledaných slov v dokumentu a vytváření inteligentních náhledů na delší texty. Tím se významně zlepšuje orientace ve výsledkové sadě nalezených dokumentů.

V mnoha případech také dochází ke kombinaci obou způsobů, kdy výsledek původní dohledávací funkcionality je použit pro omezení výsledkové sady získané fulltextovým dotazem. Lze tak například omezit výpis nalezených diskuzních příspěvků jen na příspěvky od konkrétního uživatele, který byl dohledán zadáním klíče (např. jména a příjmením). Tuto možnost demonstruje i aplikační ukázka fulltextového vyhledávání v příloze B.

6.3 Srovnání nástrojů

Jednotlivé nástroje pro fulltextové vyhledávání lze porovnávat z několika různých hledisek, mezi něž patří možnosti a rozsah funkcionalit pro přístup k požadovaným informacím, náročnost zavádění, provozu a upgrade technologie a v neposlední řadě také ekonomické hledisko srovnání.

Hotová komerční řešení disponují zpravidla širokou množinou fulltextových operátorů a funkcí, z nichž ovšem bývá při běžném vyhledávání využita jen část. Dů-

vodem je buďto nemožnost daný operátor pro prostředí českých textů použít vůbec (např. operátor *soundex* pro vyhledávání stejně znějících slov v Oracle Text, který je použitelný pouze pro angličtinu), nebo se jedná o operátor, který pro své pochopení vyžaduje hlubší studium a jím dosažený užitek není úměrný vynaloženému úsilí pro jeho použití. Takové operátory jsou pak uživateli využívány velmi málo. Dochází tak k určité nerentabilitě, jelikož v nákladech na zavedení technologie jsou zahrnuty i náklady na nepotřebné nebo nepoužívané funkcionality.

Technologie zavedená do UIS má k dispozici užší paletu operátorů (základních i pokročilých), nicméně všechny jsou uživatelsky jednoduše použitelné a také využívané. Implementované operátory společně s dalšími funkcemi pro podporu vyhledávání a prezentaci výsledků tak tvoří aparát nabízející plnohodnotné fulltextové vyhledávání s pokročilými vlastnostmi.

Velkou výhodou technologie vyvinuté vlastními silami je absolutní kontrola nad všemi fázemi indexovacího i vyhledávacího procesu. Každou z fází je tak možné přizpůsobit přesně pro účely UIS a na základě vzniku nových uživatelských požadavků modifikovat a rozšiřovat. V případě hotových řešení vzniká silná závislost na dodavateli technologie. V případě problémů při zavádění nebo provozu technologie je tak jedinou možností využít dokumentaci produktu (velmi rozsáhlá) nebo vzdálený support (málo efektivní). Podobná je situace i při rozšiřování a upgradu technologie, což je opět závislé na dodavateli a jeho vůli technologii dále rozšiřovat. Z výše uvedeného také vyplývá, že vlastní technologie disponuje největší mírou integrovatelnosti do prostředí UIS. Velmi dobrou integrovatelnost však prokazuje i technologie Oracle Text vzhledem k její přítomnosti přímo na databázové vrstvě UIS. Z porovnávaných nástrojů vykazuje nejmenší míru integrovatelnosti technologie Google Search Appliance.

Při porovnání časové náročnosti nasazení jednotlivých nástrojů je zřejmé, že vývoj vlastní technologie je časově nejnáročnější. Zkušenosti ovšem ukazují, že vzhledem k rozsáhlosti dokumentace a velmi širokým možnostem nejrůznějších nastavení při nasazování hotové technologie, vývoji rozhraní pro aplikační použití a testování, není časový rozdíl tak výrazný.

Jak při zavádění vlastní technologie, tak i při testování technologie Oracle Text bylo výchozím bodem její zpřístupnění do produkční instalace UIS pro úzkou skupinu uživatelů. Tato skupina byla tvořena zejména vývojáři a externími pracovníky pro vývoj UIS, kteří pro svou práci potřebují vyhledávání v překladových řetězcích (databáze krátkých i delších textů v různých jazycích, používaných pro multilingualitu systému). Původně bylo vyhledávání realizováno jednoduchým způsobem pomocí operátoru LIKE v podmínce databázového dotazu. S nárůstem dat však tento způsob přestal být vyhovující, stal se pomalým a neúměrně konzumujícím systémové prostředky.

Nasazením technologie Oracle Text se dosáhlo velmi výrazného zefektivnění práce s překladovými řetězci. Nicméně bylo možné pozorovat, že některé dotazy způsobují jisté potíže při běhu databázové instance. Technologii tak nebylo možné nasadit do dalších oblastí a zpřístupnit všem uživatelům. Místo toho musí zavádění

technologie projít další fází ladění a testování. Nasazením vlastní technologie bylo dosaženo srovnatelného zefektivnění práce (srovnatelné doby odezvy) při vyhledávání překladových řetězců. Řešení se ukázalo jako stabilní a tedy rozšiřitelné do dalších oblastí produkčního UIS.

Z ekonomického hlediska se jedná zejména o porovnání nákladů na pořízení a provoz technologie a o porovnání rentability, tedy zrychlení přístupu k požadovaným informacím ve vztahu k vynaloženým nákladům. V případě Google Search Appliance jsou díky licenční politice (zmiňované již v kapitole 4.5 a podrobně v [20]) pořizovací náklady neúměrně vysoké vzhledem k dosažitelným výsledkům.

Balík Oracle Text je standardní součástí instalace databázového systému Oracle, jeho pořizovací náklady jsou zahrnuty již v nákladech na pořízení celého databázového systému. Pro zavedení technologie Oracle Text do UIS jsou tak náklady tvořeny mzdovými náklady na člověkohodiny pro vývoj rozhraní a testování technologie. Podobná situace nastává i v případě vývoje vlastní technologie. V obou případech vznikají ještě dodatečné náklady na pořízení produktu ConText CZ, který je primárně určen pro Oracle Text, ale lze jej využít i pro vývoj vlastní technologie. Přínos tohoto produktu při získávání informací z UIS je obrovský, protože poskytuje vysoce vyspělou funkcionalitu ohýbání českých slov, která významně usnadňuje vyhledávání.

6.4 Možnosti rozšíření

Implementací technologie se otvírají nové prostory pro následné další rozšiřování možností vyhledávání informací v UIS. Pro další vývoj technologie jsou mimo jiné důležité ohlasy uživatelů a jejich postřehy a požadavky. Bez této zpětné vazby lze jen těžko technologii dále přizpůsobovat uživatelským požadavkům a kvalitativně posunout.

Především se jedná o rozšíření palety speciálních fulltextových operátorů. Pro doplnění možností podřetězcového vyhledávání může operátoru pravostranného rozšíření klíčů v dotazu (např. „systém∗“) tvořit jeho komplement operátor levostranného rozšíření (např. „∗jímat“), případně operátor oboustranného rozšíření (např. „ne∗jímat“). Dalším zajímavým operátorem, je operátor ekvivalence (EQUIV), který lze v dotazu použít pro specifikaci přijatelné náhrady výrazu. Například dokumenty, které obsahují výrazy *vklad hotovosti na účet* nebo *vklad peněz na účet*, by se vyhledaly dotazem „vklad hotovosti EQUIV peněz na účet“. Pokročilým operátorem je dále operátor WEIGHT, který danou hodnotou násobí výslednou relevanci výrazu. Jeho implementace úzce souvisí s dalším zinteligentňováním ohodnocovací funkce, která je implementována ve své základní podobě.

Oblastí o kterou by mohla být dále technologie rozšířena, je podpora vyhledávání v HTML, resp. XML dokumentech. To vyžaduje, aby indexovací proces ukládal další přídavné informace o sekcích a pozicích tokenů v nich do inverzního indexu. Vyhledávácí dotaz by pak s dodatečnou informací o tagu, ve kterém se má zadaný

výraz vyhledávat (např. v názvu), zajistil, že se budou prohledávat jen příslušné sekce dokumentů. K realizaci je možné zapojit poznatky uvedené v [9].

Pro nasazení fulltextového vyhledávání do všech relevantních oblastí UIS je zde možnost koexistence technologie Oracle Text a technologie vyvinuté vlastními silami. Každá technologie má své přednosti a vhodnou kombinací obou přístupů je možné tyto přednosti maximalizovat. Zejména v oblasti, kde se vyskytují rozsáhlé dokumenty nejrůznějších formátů, včetně binárních souborů, obrázků apod. (např. oblast Dokumentového serveru nebo závěrečných prací), se velmi dobře uplatní technologie Oracle Text (po odladění a důkladném otestování), jejíž indexační proces je pro zpracování takových souborů optimalizován. Pro indexaci těchto dokumentů je stěžejní korektní extrakce textových dat z dokumentu, což nepatří mezi triviální úkony. Oracle Text odporuje indexaci velmi širokého spektra různých souborových formátů, jejichž úplný výčet je dostupný v dokumentaci systému Oracle.

Pro zvýšení účinku využití stoplistu je dále možné implementovat mechanismus, který by v dané indexované oblasti automaticky detekoval slova vhodná pro zařazení do stoplistu a zajistil by modifikaci příslušného použitého stoplistu.

Pro každou oblast je také dobré použít vlastní stoplist, protože některá slova se můžou v jedné oblasti vyskytovat velmi často, v jiné zase jen zřídka. Například zadáním dotazu „univerzita“ nad kolekcí závěrečných prací lze s jistotou tvrdit, že tomuto dotazu vyhoví téměř 100 % dokumentů a slovo zde ztrácí rozlišovací schopnost při snaze vystihnout obsah dokumentu. Naopak v diskuzích lze takovým dotazem získat malé procento příspěvků, ve kterých je univerzita zmíněna.

Užitečnou funkcionalitou by byla i kontrola pravopisu slov ve vyhledávacím dotazu s případnou nabídkou substituce za správný nebo podobný výraz. Implementace této funkcionality zřejmě vyžaduje použití externího nástroje disponujícího potřebnou databází a algoritmy. Další rozsáhlou problematikou, která se otvírá v souvislosti s vyhledáváním, je zapojení sémantických principů [18] do vyhledávacích procesů.

7 Závěr

Cílem této práce bylo analyzovat možnosti pro zavedení fulltextových technologií do rozsáhlého informačního systému a jejich následné nasazení do provozu UIS. V rámci práce byla vyvinuta vlastní fulltextová technologie, která byla nasazena do různých oblastí UIS. Uživatelům systému tak přibyla možnost prohledávání diskuzí, blogů, překladových řetězců, helpdesku, evidence interních dokumentů a dalších oblastí.

Vyvinutá technologie využívá běžných prostředků RDBMS Oracle a pro dosažení některých pokročilých vlastností vyžadujících lexikální analýzu českých slov byl dále použit produkt ConText CZ. Technologie implementuje základní i pokročilejší fulltextové operátory umožňující přesnější vyjádření vyhledávacího požadavku a specifikaci vazeb mezi klíči v dotazu. Nejvíce sofistikovaným operátorem je operátor stemmingu, využívající zmiňovanou lexikální analýzu pro rozvoj slov z dotazu do všech jeho dalších tvarů, přes které se vyhledává.

Technologie dále disponuje přidruženými funkcionalitami pro podporu prezentace nalezených výsledků (zvýrazňování klíčů v textu a tvorba náhledů na text) zvyšujícími efektivnost použití vyhledávacích aplikací. Implementována byla také jednoduchá funkce pro stanovení relevance nalezeného dokumentu.

Nasazením technologie se uživatelům systému dostává do rukou poměrně komplexní nástroj umožňující plnohodnotné fulltextové vyhledávání. Nabízí prostředky nejen pro prosté získávání dat, ale také pro získávání informací obsažených v datech. Přínos lze spatřovat zejména ve zrychlení přístupu k informacím a v rozšíření vyhledávácích možností UIS o prohledávání samotných obsahů dokumentů, což doposud v systému chybělo. Významným způsobem se tak podpořila jedna ze základních funkcí informačního systému, kterou je rychlé poskytnutí informace v potřebný čas, v požadovaném rozsahu a ve vhodné formě uživateli.

Jisté rezervy lze najít ve způsobu zpracování uživatelských fulltextových dotazů. Vývoj v této oblasti by měl směřovat k uplatnění poznatků z teorie programovacích jazyků tak, aby vyhledávací dotaz mohl být zadáván jako komplexní výraz s možností definice priorit operátorů a libovolného zanořování subvýrazů.

Otevřená zůstává problematika indexování sad rozsáhlých dokumentů s různými typy souborů, včetně binárních (v UIS se jedná zejména o Dokumentový server a archiv závěrečných prací). Zde se jako výhodné řešení jeví použití nástroje Oracle Text, který je pro indexování takových dat optimalizován a podporuje indexaci velmi širokého spektra typů souborů. Za předpokladu existence nástrojů pro extrakci textových dat z daného typu souboru je možné nasadit i vlastní technologii. Tato oblast zůstává ve fázi vývoje a testování a výsledkem může být i koexistence obou technologií v provozu UIS.

8 Literatura

[1] Ciml, J. *Personifikovaný metavyhledávač*. Diplomová práce. Praha: UK, 2006, 76 s.

[2] Dařena, F. *Myslíme v jazyku PERL*. 1. vyd. Praha: Grada Publishing, 2005. 700 s. Knihovna programátora. ISBN 80-247-1147-8.

[3] Húsek, D. a kol. Metody vyhledávání v rozsáhlých kolekcích dat. In *Datakon 2003* Brno: MU, 2003, s. 23–52. ISBN 80-210-3215-4.

[4] Kopecký, M. *Dokumentografické Informační Systémy* [online]. Slidy k přednášce [cit. 3. dubna 2008]. Dostupné na internetu: `http://www.ms.mff.cuni.cz/~kopecky/vyuka/dis/disslide.ppt`.

[5] Kutín, A. Indexování a dohledávání. In *Univerzitní informační systém I.* Brno: Konvoj, 2003, s. 27–30. ISBN 80-7302-058-0.

[6] Kutín, A. *Využití obecné kvalifikace v jádře IS*. Diplomová práce. Brno: MZLU, 2004, 85 s.

[7] Lacko, L. *Oracle – správa a programování databázového systému*. Brno: Computer Press, 2003. 464 s. ISBN 80-7226-699-3.

[8] Lemay, L. *Naučte se Perl za 21 dní*. Praha: Computer Press, 2003. 546 s. ISBN 80-7226-616-0.

[9] Mužátková, M. *Nástroje pro vyhledávání v XML dokumentech*. Bakalářská práce. Brno: MZLU, 2004, 35 s.

[10] Pánek, K. *Architektury a modely webových strojů* [online]. Lupa [cit. 3. dubna 2008]. Dostupné na internetu: `http://www.lupa.cz/clanky/architektury-a-modely-webovych-stroju/`.

[11] Pánek, K. *Jehla v kupce sena: rozšířený boolský model* [online]. Lupa [cit. 4. dubna 2008]. Dostupné na internetu: `http://www.lupa.cz/clanek.php3?show=2094`.

[12] Prachař, M. *Rozšíření možností vyhledávání v UIS*. Bakalářská práce. Brno: MZLU, 2006, 43 s.

[13] Rábová, I. *Informační systémy/Informační technologie (IS/IT)*. Dokument ve formátu Ppt. Dokumentový server. 29. října 2005.

[14] Rybička, J. *LaTeXpro začátečníky*. 3. vydání. Brno: Konvoj, 2003.

[15] Šlapák, O. Data, informace, znalosti. In *E-logos, Electronic journal for philosophy*. VŠE v Praze, 2003. ISSN 1211-0442. Dostupné na internetu: `http://nb.vse.cz/kfil/elogos/miscellany/slapa103.pdf`.

[16] Šrámek, D. *Princip jednoduchého fulltextu s příklady v SQL a PHP* [online]. Pro ROOT.cz [cit. 24. března 2008]. Dostupné na internetu: `http://www.insula.cz/dali/material/fulltext101.php`.

[17] Šorm, M. *Univerzitního informační systém*. Diplomová práce. Brno: MZLU, 2002.

[18] Trenz, O. Integrace sémantických principů vyhledávání do informačních systémů. In *MendelNet 2005 (sborník abstraktů z evropské vědecké konference doktorandů)*. 1. vyd. Brno: Konvoj, 2005, s. 122–123. ISBN 80-7302-107-2.

[19] Urman, S. – Hardman, R. – McLaughlin, M. *Oracle database 10g PL/SQL programming*. Emeryville: Oracle Press, 2004. 862 s. ISBN 0-07-223066-5.

[20] *Google Search Appliance*. Firemní dokumentace produktu. [cit. 7. dubna 2008] `http://www.google.co.uk/enterprise/`.

[21] *Oracle Text Application Developer's Guide* [online]. Oracle10g dokumentace [cit. 10. března 2008]. `http://tahiti.oracle.com`.

[22] *Oracle Text Reference* [online]. Oracle10g dokumentace [cit. 10. března 2008]. `http://tahiti.oracle.com`.

[23] *Performance Tuning Guide* [online]. Oracle10g dokumentace [cit. 10. března 2008]. `http://tahiti.oracle.com`.

[24] *Produktové řešení Google*. Webové stránky dodavatele řešení Google. [cit. 12. dubna 2008] `http://www.inspiro.cz/sluzby-smb-produktove-reseni-google.cml`.

[25] *Sefira*. Webové stránky firmy. [cit. 7. dubna 2008] `http://www.sefira.cz`.

Přílohy

A Stoplist českých a anglických slov v ConText CZ

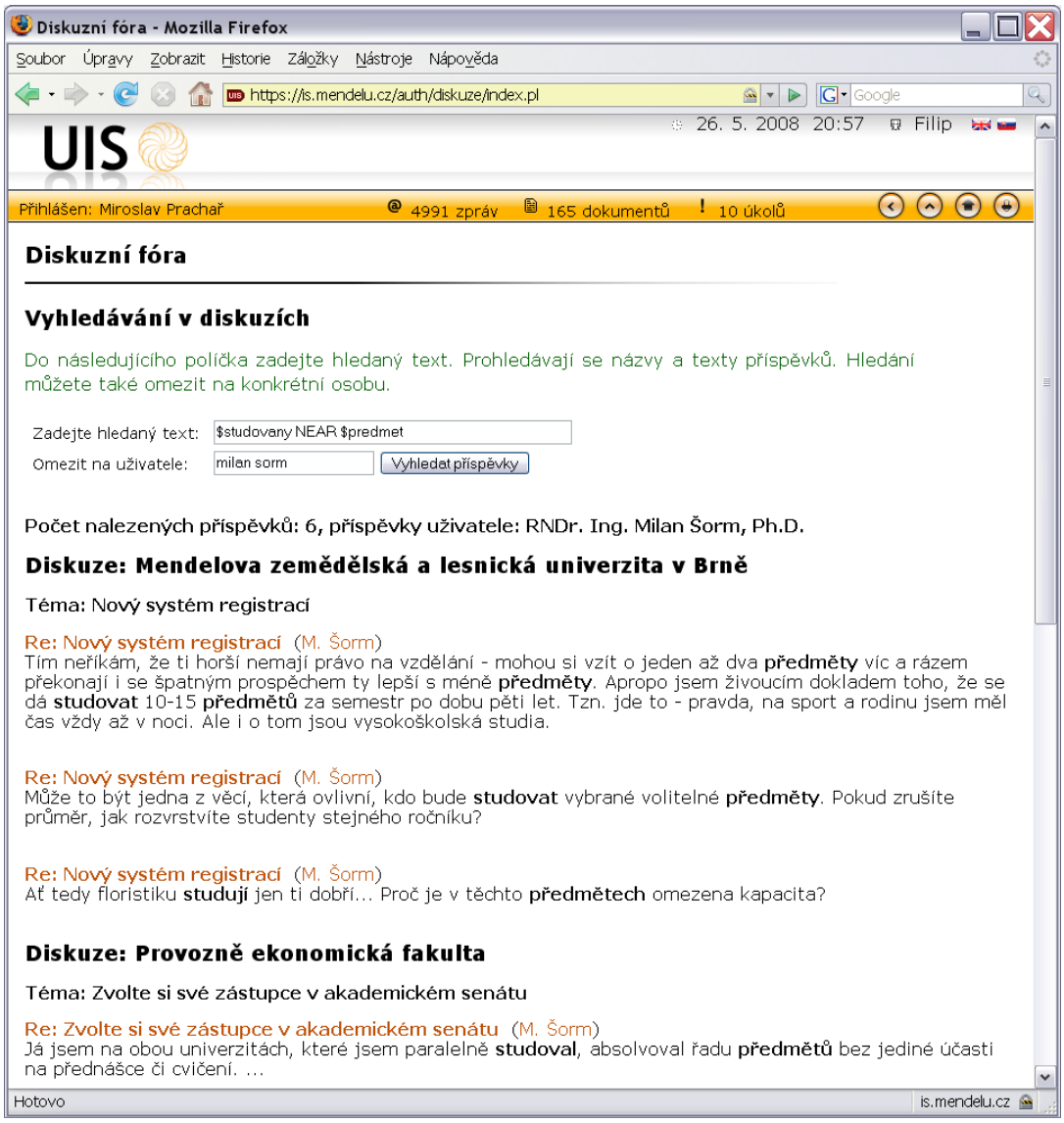
aby	což	jenom	ke	náš	pokud	svůj	váš
abychom	čemu	jenž	koho	ne	pro	tady	včera
ale	či	jestliže	komu	nebo	proti	tak	ve
ani	čí	ještě	které	něco	proto	také	velmi
asi	dát	jež	kterého	nesvůj	protože	taková	více
ať	do	jim	který	než	prý	tam	vlastně
až	chtít	jimi	kteří	nic	před	teď	však
bez	i	jiné	ku	nikdo	přes	tedy	vůbec
bude	já	již	lze	o	při	ten	vy
budou	jak	jsem	mezi	od	přičemž	tento	vým
byl	jaké	jsi	mít	ona	přitom	tím	z
byla	jako	jsou	moci	oni	říci	totiž	za
bylo	jakoby	jste	možná	ono	sám	třeba	zda
byly	jaký	k	můj	ony	se	tvůj	ze
být	je	každé	muset	ovšem	si	ty	že
býval	jeho	kde	my	pak	sice	u	
bývali	jejich	kdo	na	po	snad	už	
bývalo	jelikož	kdy	nad	pod	spíše	v	
bývaly	jen	když	namísto	podle	stát	vás	

Tab. 1: Implicitní stoplist českých slov v produktu ConText CZ

a	at	for	is	not	s	then	who
about	be	from	it	of	says	there	will
after	because	have	its	on	she	they	with
all	been	he	last	one	so	this	would
also	but	his	more	only	some	to	
an	by	if	mr	or	such	up	
any	can	in	mrs	other	that	was	
are	corp	inc	ms	out	the	were	
as	could	into	mz	over	their	which	

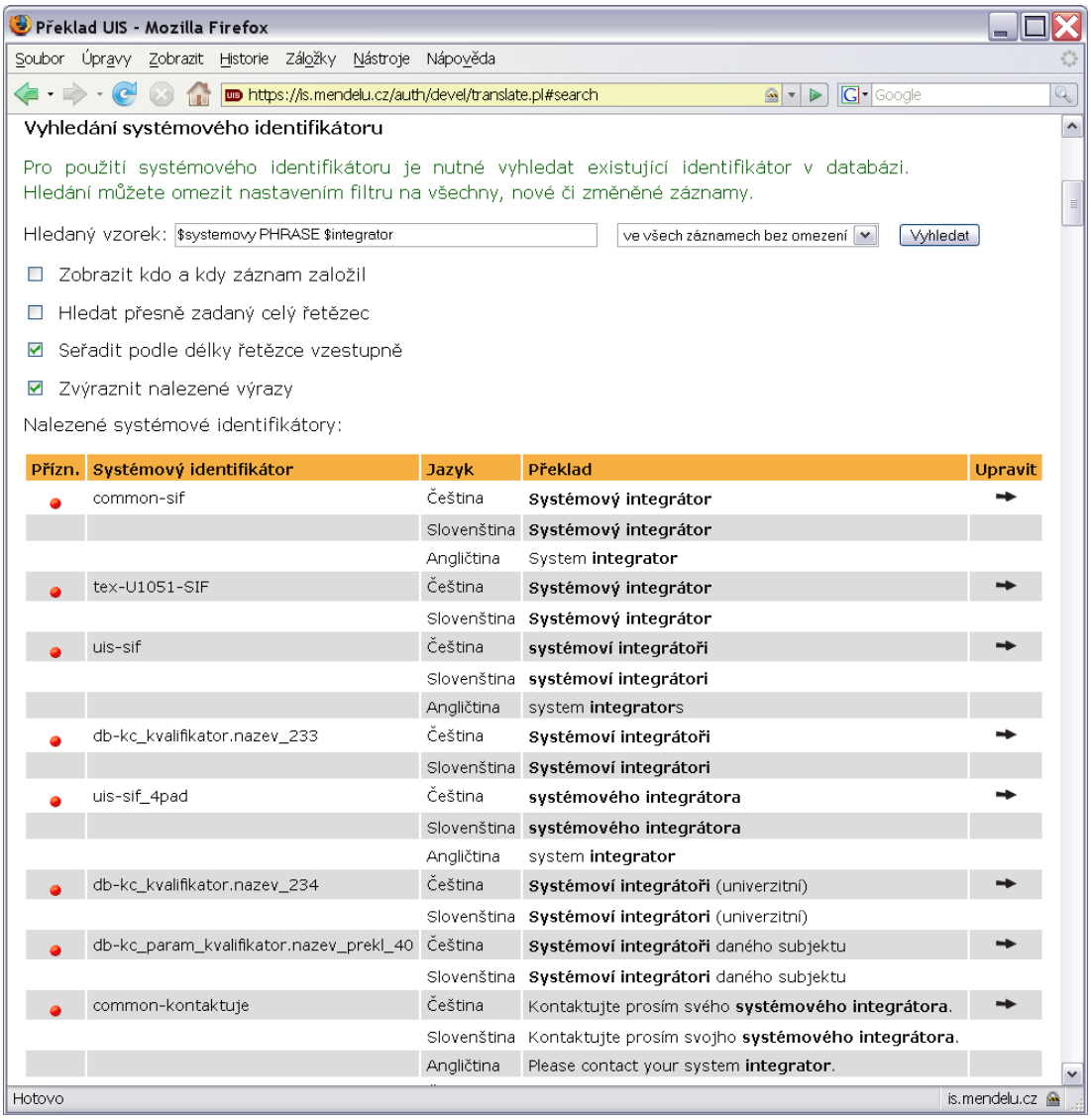
Tab. 2: Implicitní stoplist anglických slov

B Ukázka vyhledávání v diskuzních fórech



Obr. 6: Ukázka aplikace pro vyhledávání v diskuzních fórech

C Ukázka vyhledávání v překladových řetězcích



Obr. 7: Ukázka aplikace pro vyhledávání v překladových řetězcích

D Příklad komplexního vyhledávacího SQL dotazu

```
SELECT full.ID TEMA, full.CBRI ID, pr.NAZEV_PRISPEVKU, pr.PRISPEVEK,
        pr.DISKUZE, pr.VLOZIL VLOZIL_ID, ZKRACENE_JMENO(pr.VLOZIL) VLOZIL,
        pr.REAKCE_NA, full.nazev_tematu NAZEV_TEMATU,
        disk.nazev NAZEV_DISKUZE
from L_DISKUZE_PRISPEVKY pr, LC_DISKUZE disk,
     (SELECT distinct id, cbri, nazev_tematu
      from (SELECT id, cbri, diskuze, nazev_tematu
            from (SELECT id, connect_by_root id cbri, reakce_na,
                         diskuze, nazev nazev_tematu
                  FROM L_DISKUZE_PRISPEVKY
                  start with id in (
                      -- vlastní fulltextový dotaz
                      SELECT distinct odkaz
                      FROM L_DISKUZE_PRISPEVKY_FIDX i,
                           L_DISKUZE_PRISPEVKY_FWIDX w
                      WHERE i.TOKEN = w.ID and w.TOKEN = 'informační'
                      INTERSECT
                      SELECT distinct odkaz
                      FROM L_DISKUZE_PRISPEVKY_FIDX i,
                           L_DISKUZE_PRISPEVKY_FWIDX w
                      WHERE i.TOKEN = w.ID and w.TOKEN = 'systém'
                  )
                  connect by prior reakce_na = id)
                  where reakce_na is null) fullset,
                  LA_DISKUZE_CTENARI prav
            where prav.diskuze = fullset.diskuze
                  and prav.ctenar = 6891 ) full
where full.cbri = pr.id
      and pr.diskuze = disk.id
      and disk.zruseno = 0
      and vlozil = 4021
order by disk.nazev, full.nazev_tematu
```