Univerzita Pardubice

Fakulta ekonomicko-správní

Současný stav fulltextového vyhledávání v MySQL

Ivana Broklová

Bakalářská práce

2011

Univerzita Pardubice
Fakulta ekonomicko-správní
Akademický rok: **2010/2011**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Ivana BROKLOVÁ, DiS.**
Osobní číslo:	**E080017**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Regionální a informační management**
Název tématu:	**Současný stav fulltextového vyhledávání v MySQL**
Zadávající katedra:	**Ústav systémového inženýrství a informatiky**

Zásady pro vypracování:

Vyhledávací metody a jejich porovnání
Charakteristika fulltextového vyhledávání
Varianty fulltextového vyhledávání, jejich omezení a porovnání
Návrh přípravy tabulek v databázi a jejich optimalizace
Možnosti prezentace výsledků, zajištění relevance
Další možnosti doplnění fulltextu
Otestování vlastního návrhu fulltextového vyhledávání internetového obchodu, zhodnocení výsledků

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. SHELDON, Robert. SQL začínáme programovat. Vydání první. Praha: Grada Publishing, a.s., 2005. 500 s. ISBN 80-247-0999-6.
2. D. SCHNEIDER, Robert . MySQL: Oficiální průvodce tvorbou, správou a laděním databází. Vydání první. Praha: Grada Publishing, a.s., 2006. 372 s. ISBN 80-247-1516-3.
3. KOFLER, Michael. Mistrovství s MySQL: Kompletní průvodce webového vývojáře.Vydání první. Brno: Computer Press, a.s., 2007. 805 s. ISBN 978-80-251-1502-2.
4. DUBOIS, Paul. MySQL profesionálně: Komplexní průvodce použitím, programováním a správou MySQL. Vydání druhé. Praha: Mobil Media a.s., 2003. 1071 s. ISBN 80-86593-41-X.
5. GUTMANS, Andi; SAETHER BAKKEN, Stig; RETHANS , Derick. Mistrovství v PHP 5. Vydání druhé. Brno: Computer Press, a.s., 2007. 655 s. ISBN 9788025115190
6. MySQL [online]. 2010 [cit. 2010-05-25]. Dostupné z WWW: <http://dev.mysql.com/>.

Vedoucí bakalářské práce: **Ing. Renáta Máchová, Ph.D.**
Ústav systémového inženýrství a informatiky

Datum zadání bakalářské práce: **5. října 2010**

Termín odevzdání bakalářské práce: **6. května 2011**

L.S.

doc. Ing. Renáta Myšková, Ph.D.
děkanka

doc. Ing. Jiří Krupka, Ph.D.
vedoucí ústavu

V Pardubicích dne 5. října 2010

Prohlašuji:

Tuto práci jsem vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila, jsou uvedeny v seznamu použité literatury.

Byla jsem seznámena s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 25. 4. 2011

Ivana Broklová

PODĚKOVÁNÍ

Tímto děkuji vedoucí bakalářské práce Ing. Renátě Máchové, Ph.D. za odborné vedení, trpělivost a cenné rady a připomínky při zpracování této práce.

Děkuji Martinovi Rozhoňovi, řediteli společnosti VIVANTIS, a.s., za jeho ochotu a čas strávený společnými konzultacemi. Dále děkuji Petrovi Želichovskému, z vývojářského týmu společnosti VIVANTIS a.s., za odborné rady, ochotu, trpělivost a čas při zodpovídání mých dotazů.

Děkuji svému manželovi a vůbec celé rodině za podporu při studiu.

ANOTACE

Tato bakalářská práce popisuje současný stav fulltextového vyhledávání webových aplikací realizovaných nad databázovým serverem MySQL. Práce popisuje varianty fulltextového vyhledávání v MySQL včetně jejich porovnání. V práci je také popis návrhu a optimalizace databázové struktury pro fulltextové vyhledávání. Závěrečná část práce zahrnuje otestování fulltextového vyhledávání internetového obchodu a také popisuje možnosti doplnění fulltextového vyhledávače.

KLÍČOVÁ SLOVA

Fulltextové vyhledávání, MySQL, databáze, optimalizace

TITLE

The current status of full-text search in MySQL

SUMMARY

This bachelor’s project describes the current status of full-text search of web applications implemented over the MySQL database server. The work deals with variations of the full-text search in MySQL, including their comparison. The work explains design and optimizing the database structure for full-text search. The final part involves testing the full-text search in an e-shop and also describes the possibilities to complement full-text search engine.

KEYWORDS

Full-text search, MySQL, database, optimization

OBSAH

ÚVOD

V průběhu života získává člověk velké množství informací. Mezi často používané informační zdroje patří počítačová síť internet.

V posledních letech prudce vzrůstá počet uživatelů internetu. Dle Českého statistického úřadu [4] se mezi lety 2005 a 2009 v České republice téměř zdvojnásobil počet uživatelů internetu ve věku 16 – 74 let (32 % v roce 2005 a 60 % v roce 2009). Obdobným tempem rostl i počet jednotlivců, kteří internet používají ke komunikaci – z 27 % v roce 2005 na 55 % v roce 2009.

Nejznámější službou poskytovanou v rámci internetu je World Wide Web neboli webové stránky, na kterých jsou data v různých podobách ukládány – a to ve formě tiskových zpráv, odborných článků, příspěvků diskusních fór, firemních prezentacích atd.

K efektivnímu využívání internetu je nezbytné znát a používat různé druhy vyhledávání. Někteří uživatelé nevědí, kde a jak požadované „odpovědi na otázky“ vyhledat nebo jak formulovat své zadání, aby výsledek vyhledávání byl relevantní našemu očekávání, tedy správná odpověď. Vyhledávání by mělo být jednoduché, rychlé a zobrazovat co možná nejpřesnější (relevantní) nalezené reference podle shody se zadaným hledaným výrazem.

Cílem této bakalářské práce je charakterizovat vyhledávací metody používané na internetu a dále charakterizovat současný stav fulltextového vyhledávání webových aplikací vytvořených nad databázovým serverem MySQL a uvést varianty fulltextového vyhledávání v MySQL včetně jejich porovnání.

Dalším cílem této práce je otestování vlastního návrhu fulltextového vyhledávání internetového obchodu včetně zajištění relevance zobrazených výsledků. Pro zobrazení relevantních výsledků vyhledávání je nutný správný návrh databáze a proto je dalším cílem provést optimalizaci databáze pro fulltextové vyhledávání a vystihnout možnosti doplnění fulltextového vyhledávání.

Přála bych si, aby tato práce pomohla zájemcům o internetové technologie přiblížit problematiku fulltextového vyhledávání a tím pomohla i k vytvoření fulltextového vyhledávače, který usnadní potenciálním uživatelům vyhledávání na internetových stránkách.

1 VYHLEDÁVACÍ METODY POUŽÍVANÉ NA INTERNETU A JEJICH POROVNÁNÍ

Důležitým článkem potřebným k získávání informací v prostředí internetu je jejich vyhledávání. Tato kapitola je věnována vyhledávacím nástrojům, metodám vyhledávání, jejich popisu a porovnání.

Nejdříve je nutné definovat základní pojmy obsažené v této kapitole. Dle [14] jsou data „surová (nezpracovaná) fakta, která mají určitou důležitost pro jednotlivce nebo organizaci. Informace jsou data, která prošla zpracováním nebo dostala strukturu, která jim dává pro jednotlivce nebo organizaci význam."

Vyhledávací nástroje lze dělit na dva základní typy podle metody vyhledávání – fulltextový vyhledávač a katalog. Mezi méně časté způsoby používané běžnými uživateli patří vyhledávání pomocí metavyhledávače.

Uživatelé mohou na internetu vyhledávat požadované výrazy za pomoci fulltextových vyhledávačů, které jim dle shody v názvu nebo popisku zobrazí relevantní odkazy odpovídající zadanému výrazu. Příkladem fulltextového vyhledávače je Google (http://www.google.cz).

Technologie fulltextových vyhledávačů se skládá ze tří částí. První částí je program nazývaný spider (pavouk) nebo crawler (slídil), který prochází webové stránky a ukládá jejich obsah a reference do databáze vyhledávací centrály, kde jsou postupně zpracovávány a uspořádány druhou částí - dalším programem nazývaným indexér. Výsledkem je uspořádaný rejstřík neboli index. Třetí část vyhledávací centrály je vlastní vyhledávač (search engine). To je program, který přebírá od uživatele dotaz na vyhledávání, v indexu nalezne webové stránky s odpovídajícím obsahem, seřadí je dle relevantnosti a zobrazí reference uživateli. V případě pokročilého vyhledávání je nejčastěji používán booleovský model založený na vyhodnocování booleovských výrazů (`AND, OR, NOT`). [7]

Další možností pro vyhledávání daného výrazu je použití katalogu, ve kterém jsou odkazy roztříděny do základních kategorií, které se dále dělí na podkategorie. Uživatelé mohou sekvenčně procházet jednotlivé kategorie a podkategorie, o které mají zájem, nebo použít jednoduchého vyhledávání. Mezi nejznámější katalogy patří české portály Seznam (http://www.seznam.cz), Centrum (http://www.centrum.cz) a Atlas (http://www.atlas.cz).

Hlavní charakteristikou katalogu je, že je vytvářen lidmi. Odkazy jsou do databáze zadávány ručně podle jejich obsahu do hierarchické struktury kategorií a podkategorií, a to buď autory stránek, nebo správci katalogu.

Google využívá jako katalog data z projektu „DMOZ – Open Directory Project (ODP)", jehož jádrem je soubor internetových stránek vybraných lidmi, kteří dobrovolně pracují jako

redaktoři Open Directory. Podle názoru společnosti Google na kvalitu stránek jsou jednotlivé internetové stránky řazeny dle jejich důležitosti a tak jsou nejrelevantnější stránky v každém seznamu uváděny na prvních místech. [13]

Ve většině vyhledávacích centrál se používají obě varianty vyhledávání. Jedna je však preferovaná. Například Seznam upřednostňuje katalog, Google upřednostňuje fulltextový vyhledávač.

Ze základních rozdílů mezi fulltextovým vyhledavačem a katalogem vyplývají přednosti i nevýhody použití obou způsobů, které jsou dle kriterií shrnuty v následujícím textu [2,7]:

- **Kriterium: Rozsah vyhledávání**

  a) Katalog

  Při ručním zatřiďování se zařazuje do odpovídající kategorie pouze reference na úvodní webovou stránku a nikoliv jednotlivé webové stránky. Tím je snížen počet celkových zatříděných položek a snížen rozsah vyhledávání.

  b) Fulltextový vyhledávač

  Fulltextový vyhledávač vytvářený automatickým programem má široký záběr stránek.

  Zhodnocení dle kriteria

  Z hlediska rozsahu vyhledávání je použití fulltextového vyhledávače výhodnější.

- **Kriterium: Kvalita odkazů**

  a) Katalog

  U katalogu je možnost kontroly kvality odkazů a tak je jeho použití z toho hlediska výhodnější.

  a) Fulltextový vyhledávač

  Kvalita odkazů u fulltextového vyhledávače kolísá.

  Zhodnocení dle kriteria

  Z hlediska kvality odkazů je výhodnější používat katalog.

- **Kriterium: Vyhledání specifických výrazů**

  a) Katalog

  Katalogy obsahují spíše obecná témata.

  b) Fulltextový vyhledávač

  Fulltextový vyhledávač může najít i specifické termíny.

  Zhodnocení dle kriteria

  Z hlediska vyhledávání specifických výrazů je použití fulltextového vyhledavače výhodnější.

- **Kriterium: Zobrazení relevantních referencí**

  a) Katalog

  Při ručním zatřiďování do katalogu člověk daleko lépe zařadí stránky do odpovídající kategorie. Znamená to, že uživatelé naleznou stránky, které do nich tematicky patří a tím je z tohoto hlediska použití katalogu výhodnější.

  Pokud jsou internetové stránky v rámci jednoho webu různorodé, může nastat problém se správným zařazením do jednotlivých kategorií.

  b) Fulltextový vyhledávač

  Při zatřiďování položek se může stát, že automatický program nezatřídí stránky správně a tak uživatelům nezobrazuje relevantní reference.

  Zhodnocení dle kriteria

  Z hlediska zobrazování relevantních referencí je použití výhodnější použití katalogu.

U obou způsobů vyhledávání záleží na kvalitě zpracování. V případě katalogu záleží na precizním zatřídění, u fulltextu rozhoduje, jak často svá data aktualizuje.

Dalším typem vyhledávání jsou metavyhledávače shromažďující výsledky několika jiných vyhledávačů. Lze je rozdělit na metavyhledávací servery na internetu a metavyhledávací programy nainstalované v počítači uživatele. Po zadání dotazu do vyhledávače jej metavyhledávač mnohonásobně „zkopíruje“ a rozešle na jiné vyhledávače. Položí jim dotaz, na který vyhledávače odpoví zobrazením stránek s výsledky. Metavyhledávač tyto stránky s výsledky utřídí, seřadí, odstraní duplicity a zobrazí uživateli. [7]

Nevýhodou metavyhledávačů je prodloužení doby vyhledávání a možnost snadno a spolehlivě je blokovat vlastníky vyhledávacího serveru, kteří si snadno zjistí jejich adresu a následně všechny dotazy z této adresy blokují. Tento proces blokace je značně omezen v případě nainstalování a spuštění metavyhledávače v počítači uživatele. [7]

Příkladem metavyhledávače je Search (http://www.search.com) nebo Alenka (http://www.alenka.cz).

V prostředí internetu se nachází množství webových aplikací (diskusní fóra, internetové obchody, online aukce) obsahujících až několik set stránek. Pro uživatele je vyčerpávající pátrat po informacích postupným procházením jednotlivých stránek, a proto většina těchto prezentací obsahuje interní vyhledávání. Jedná se o programové rozšíření.

Existuje několik variant interního vyhledávání informací. Jednou z možností je data třídit dle různých kriterií do kategorií, které lze postupně zobrazovat a tematicky prohledávat.

Další možností je vyhledávání pomocí zadaných kriterií neboli tzv. průvodce, kterému

uživatel vyplní atributy k filtrování (např. obor zaměstnání nebo značku hledaného produktu) a následně se zobrazí relevantní výsledky vyhledávání odpovídající zadaným požadavkům.

Velmi používané je interní fulltextové vyhledávání, které zobrazuje relevantní výsledky na základě shody se zadaným výrazem. Tato bakalářská práce se dále zaměří na interní fulltextové vyhledávání webových aplikací, které používají databázový server MySQL.

Dříve, než budou popsány základní charakteristiky současného stavu fulltextového vyhledávání v MySQL, považuji za nutné přiblížit základní komponenty používané pro webové aplikace.

2 KOMPONENTY WEBOVÝCH APLIKACÍ

Cílem této kapitoly je přiblížit základní komponenty používané pro vývoj webových aplikací, popsat jejich základní komunikaci a prezentovat hlavní představitele jednotlivých komponent.

Webové aplikace (typu online aukce, internetového obchodu nebo diskusního fóra) používají kód programovacího jazyka, který je spouštěný na serveru, a tak pohotově reagují na požadavky klienta a dynamicky tvoří stránky HTML, jež se zobrazí v okně webového prohlížeče spuštěného na klientském počítači. Webové aplikace ukládají údaje do souborů a databází. [10]

Webová aplikace je na straně serveru tvořena [10]:

- webovým serverem,
- programovacím jazykem,
- databázovým serverem,
- operačním systémem.

Velmi oblíbená a často používaná konfigurace se skládá z webového serveru Apache HTTP server, skriptovacího jazyka PHP a databázového serveru MySQL, jež jsou implementovány na platformě operačního systému Linux. [8]

2.1 Webový server

Webový server může být počítač, který je připojen k počítačové síti a vyřizuje požadavky HTTP od klientů - často webových prohlížečů. Vyřízením požadavku se rozumí odeslání HTTP odpovědi. [8]

Komunikace protokolu HTTP je založena na principu požadavek/odpověď. Klient sestaví požadavek (dotaz) a po otevření přenosového kanálu jej odešle serveru. Tím serveru sdělí, co od něj vyžaduje. Nejčastěji se jedná o zaslání konkrétního dokumentu. Webový server přijme požadavek HTTP a odešle klientovi odpověď – konkrétní požadovaný dokument. V případě zajištění dynamiky na straně serveru je požadováno, aby server načetl určitá data (např. z databáze), tato data určitým způsobem zpracoval a následně automaticky vygeneroval příslušný HTML kód, který se bude nacházet v těle HTTP odpovědi. Nakonec server odešle vytvořenou odpověď zpět prohlížeči pro interpretaci a zobrazení. [8]

Webový server lze instalovat službou IIS (internetová informační služba) společnosti Microsoft nebo instalací Apache HTTP serveru. IIS funguje pouze na operačním systému Windows, Apache HTTP server je možné instalovat na operační systém Windows i Linux.

Apache HTTP Server (http://www.apache.org) je open-source produkt s dostupným zdrojovým kódem implementace HTTP webového serveru. Tržní podíl serverů dle výrobce ukazuje průzkum společnosti Netcraft Ltd. Od roku 1996 patří mezi nejčastěji používané servery Apache HTTP server. Podle průzkumu společnosti Netcraft Ltd byl implementován v květnu 1999 na 57 % všech serverů, v listopadu 2005 jeho používání dosáhlo 69 % a v únoru 2011 pokleslo na 60 %. [12]

2.2 Programovací jazyk

Pro vývoj webových aplikací se často používá skriptovací jazyk PHP, dále lze použít jazyk Perl nebo Python. Nejvážnějšími konkurenty pro PHP je platforma JAVA se svými Java Server Pages (JSP) a Java servlety a platforma .NET se svým ASP.NET. [8]

PHP (http://www.php.net) je skriptovací jazyk s podporou objektově orientovaného programování a jazyka XML. Jedná se o multiplatformní Open Source Software s přímou podporou databáze (např. MySQL, Oracle).

K zajištění dynamiky na straně uživatele lze použít jazyk JavaScript. Existuje i rozšíření jazyka JavaScript – objektový model dokumentu (DOM). V posledních letech vznikla pro JavaScript řada open source knihoven, např. jQuery (http://jquery.com).

2.3 Databázový server

Databáze je souhrn uspořádaných a navzájem propojených dat uložených v jednom nebo několika datových souborech, bez zbytečné redundance, aby mohla sloužit více databázovým aplikacím. Data jsou uložena v paměti tak, že jsou nezávislá na programech, které je používají. Data jsou strukturována v tabulkách, s možnými referencemi mezi tabulkami. Existence relací mezi tabulkami vedla k zavedení pojmu relační databáze. [1,9]

Databázovou aplikací je program napsaný v některém programovacím jazyce a slouží pro výběr, prezentaci, zpracování a tisk dat, uložených v databázi. [1]

Systém řízení báze dat (SŘBD) je softwarový systém sloužící k vytváření, udržování a aktualizaci databáze. Tento systém zajišťuje bezporuchový průběh současného využívání databáze více uživateli a všestranné zabezpečení proti nesprávné a nedovolené manipulaci nebo chybám počítače a jeho programového vybavení. Databáze a SŘBD tvoří dohromady databázový systém. [1,3]

K úkolům relačních databázových systémů nepatří jen bezpečné skladování dat, ale také vykonávání příkazů pro vyhledávání, analyzování a třídění existujících dat a ukládání nových dat.

Tyto úkoly neprobíhají pouze na jednom počítači, ale také na celé síti počítačů. Proto je často použit termín databázový server. [9]

Databázový server je dle [8] „počítač vybavený příslušným hardwarem a softwarem, nejčastěji SQL serverem, což je aplikace, která pro klienty (pracovní stanice) zpracovává veškeré manipulace s daty. Jedním z často používaných SQL serverů je MySQL.“

Pro vývoj webových aplikací lze použít i objektově-relační PostgreSQL nebo multiplatformní databázový systém Oracle.

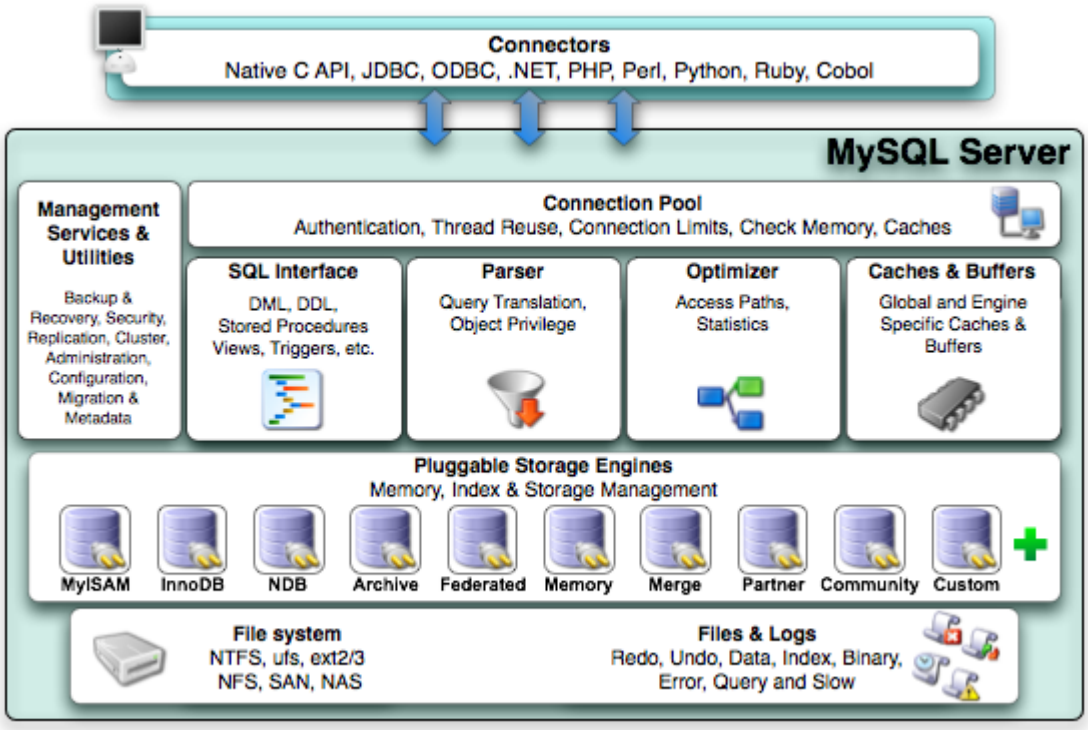
MySQL (http://www.mysql.com) je databázový systém vytvořený švédskou firmou MySQL AB, nyní vlastněný společností Sun Microsystems, dceřinnou společností Oracle Corporation. MySQL je k dispozici jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí.

Díky tomu, že se jedná o snadno implementovatelný volně šiřitelný databázový server, dále je multiplatformní (např. pro operační systémy Linux, Solaris, Windows), výkonný a nejčastěji používaný v kombinaci se skripty napsanými v jazyce PHP získal MySQL vysoký podíl mezi současně používanými databázemi**.** [8]

Obrázek 1 znázorňuje architekturu MySQL serveru. Horní vrstva obsahuje služby obsluhující většinu potřebných nástrojů klient/server, jako je např. zpracování připojení, autentizace, bezpečnost atd. [18]

Ve druhé vrstvě se nachází parser pro rozbor dotazu a optimalizátor dotazu. MySQL provádí rozbor dotazů a vytváří tak interní stromovou struktur (parse tree) a následně aplikuje všechny optimalizace. Může např. dotaz přepsat, určit pořadí čtení tabulek anebo zvolit, který index použije. V této vrstvě se nachází také další zabudované funkce (například pro datum a čas, matematické výpočty, pro optimalizaci) a také veškerá funkcionalita, která se poskytuje prostřednictvím úložných enginů – např. uložené procedury nebo pohledy. Na této úrovni se nachází i cache dotazů, kam se ukládají dotazy s jejich výslednými sadami. Pokud uživatel zadá identický dotaz s dotazem uloženým v cache, předá server zpět uloženou sadu a nemusí dotaz zpracovávat. Třetí vrstva obsahuje úložné enginy, které jsou zodpovědné za ukládání a načítání dat uložených v MySQL. [18]

Server komunikuje s úložnými enginy prostřednictvím API rozhraní, které umožňuje provádět takové operace, jako je např. získání určitých řádků dle primárního klíče nebo zahájení transakce. Úložných enginů neboli databázových úložišť nabízí MySQL několik druhů – MyISAM, InnoDB, NDB, Merge, Archive, Memory, a Federated. [17,18]



Obrázek 1: Architektura MySQL serveru [16]

Výchozí úložný engine MySQL je MyISAMkterý je rychlý, podporuje fulltextové vyhledávání, ale nepodporuje transakce. Engine InnoDB je robusní úložiště navržené pro zpracování transakcí se silnou referenční integritou, které je často používáno pro složité aplikace s velkým objemem dat. V současné době InnoDB nepodporuje fulltextové vyhledávání, ale vývojáři na jeho integraci pracují. Pokud je žádoucí používat InnoDB lze replikovat tabulky do nějaké repliky, jejíž tabulky používají úložný engine MyISAM, a na této replice obsluhovat fulltextové dotazy. Další možností je tabulku vertikálně rozdělit na dvě a udržovat textové sloupce odděleně od zbytku dat. Další alternativou je některé sloupce zduplikovat do tabulky, která je fulltextově indexovaná a následně udržovaná pomocí triggerů[1]. Zajímavá je i varianta s použitím nějakého externího fulltextového enginu, jako je Mnogosearch nebo Sphinx, který je podrobněji představen v kapitole 6.5. [17,18]

Samotný výběr úložného enginu, případně jejich kombinace, může být do značné míry ovlivněn konkrétními požadavky zadání. Lze tak předejít komplikacím, kdy je zjištěno, že úložný engine neposkytuje nějakou konkrétní funkcionalitu.

[1] Trigger je automatické volání uložených procedur nebo příkazů SQL před nebo po provedení příkazů INSERT, UPDATE nebo DELETE. [9]

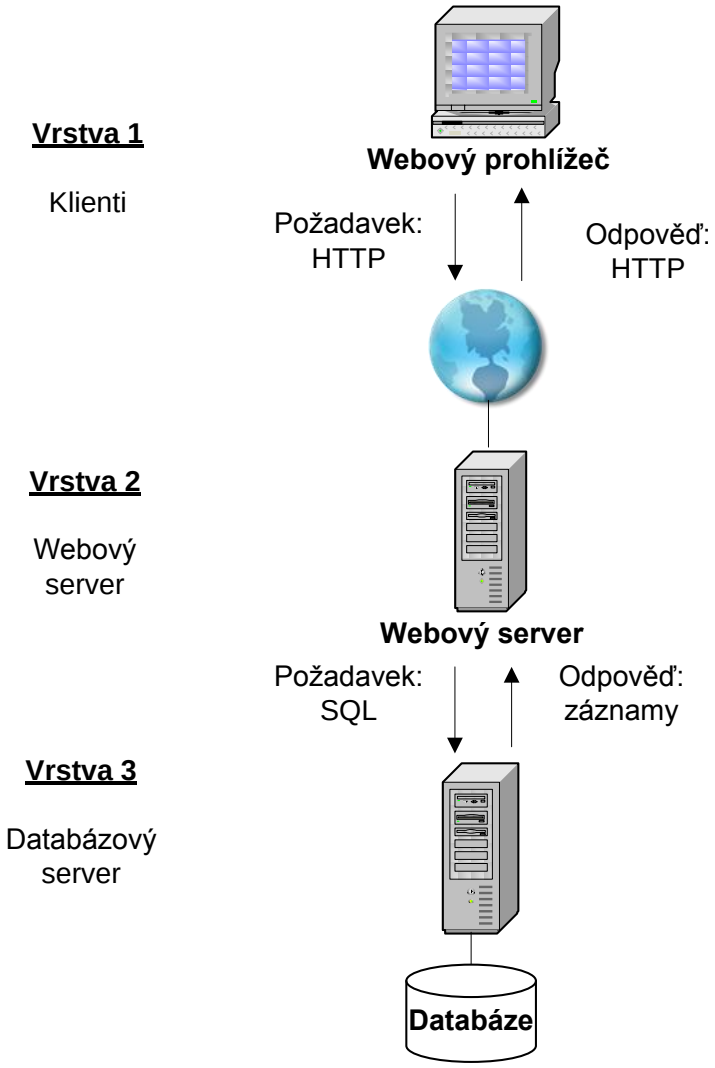
MySQL je databázový server, se kterým probíhá komunikace pomocí jazyka SQL. **Jazyk SQL** (*Structured Query Language)* je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. Jazyk SQL je možné také označit jako univerzální jazyk relačních databází, protože jej podporují prakticky všechny SŘBD, které se v současné době používají. [14]

SQL je neprocedurální neboli deklarativní jazyk. Nedefinuje, jak výsledky získat, ale pouze je počítači sděleno, jaké výsledky jsou požadovány. Toto sdělení se provádí pomocí dotazu. Jazyk SQL slouží ke správě a údržbě relačních databází. Obvykle se používá v kombinaci s procedurálními a objektově orientovanými jazyky (např. PHP), které zajišťují ukládání a načítání dat, prezentaci dat na webové stránce, nebo přímo reagují na uživatelský vstup z klávesnice a vykonávají určité příkazy. Pokud vzniká požadavek aplikace na interakci s databází, tak program vytvoří pomocí příkazů procedurální jazyka příkaz jazyka SQL a ten pak odešle ke zpracování RSŘBD, ze kterého následně přijme výsledky a zpracuje je dle požadavku. [14]

SQL je nejrozšířenější jazyk, který umožňuje tvořit databázové dotazy. Dotaz (query) je požadavek, který se odesílá databázi a na základě tohoto požadavku poskytne zpětně databáze určitou odpověď žadateli. Pomocí SQL lze získat odpověď i na komplikované dotazy. [14]

Obrázek 2 znázorňuje jednotlivé komponenty webových aplikací, které tvoří třívrstvou architekturu a také zobrazuje jejich základní komunikace. Skript[2] na webovém serveru přečte požadavek HTTP od klienta (webového prohlížeče) a provede jeho zpracování sestavením odpovídajících příkazů v SQL. Následně se skript na webovém serveru připojí k SŘBD na databázovém serveru a předá mu SQL příkaz. SŘBD vykoná SQL příkaz, sestaví výsledné záznamy a vrátí je na webový server. Skript na webovém serveru přijme tyto výsledky, zformátuje je pomocí HTML značek a odešle zpět prohlížeči pro interpretaci a zobrazení. [3]

[2] Skript je kratší program, jehož úkolem je provést jeden nebo více úkolů v rámci určité aplikace, jejíž instrukce a pravidla používá. [11]



Obrázek 2: Komponenty webové aplikace a jejich komunikace, upraveno dle [3]

3 FULLTEXTOVÉ VYHLEDÁVÁNÍ V MYSQL

Tato kapitola popisuje základní charakteristiku fulltextového vyhledávání v MySQL, varianty fulltextového vyhledávání, jejich omezení a porovnání.

„Fulltextové vyhledávání je způsob vyhledávání nebo také organizace databáze textů umožňující porovnání každého slova dokumentu se zadaným vzorem.“ [11]

3.1 Charakteristika fulltextového vyhledávání v MySQL

Současná verze MySQL 5.5 (od verze 3.23) obsahuje schopnosti fulltextového indexování a vyhledávání textů, které umožňuje hledat slova nebo slovní spojení. Schopnost fulltextového vyhledávání se musí pro danou tabulku zapnout tím, že pro ni vytvoříme speciální indexy. Fulltextové vyhledávání v MySQL má tyto hlavní charakteristiky [6]:

- Vyhledávání jsou založena na indexech typu `FULLTEXT`. Tyto indexy lze vytvářet jen u tabulek MyISAM a jen pro sloupce typu `TEXT` a nebinární sloupce `CHAR` a `VARCHAR`.
- Index `FULLTEXT` lze vytvořit pro jeden nebo několik sloupců. Pokud se rozprostírá přes několik sloupců, prohledávání podle tohoto indexu probíhá simultánně ve všech sloupcích.
- Při fulltextovém vyhledávání se nerozlišuje velikost písmen.
- Slova jsou definovaná jako posloupnosti znaků, které se skládají z písmen, číslic, apostrofů a podtržení. To znamená, že např. slovo „full-blooded“ se považuje za dvě slova „full“ a „blooded“. Za normálních okolností se hledají celá slova, ne části slov. Fulltextový stroj považuje záznam za odpovídající, pokud v něm nalezne jakékoli ze slov uvedených v hledaném řetězci. Booleovská varianta hledání, která bude popsána později, umožňuje přidat dodatečné omezení, že se musí vyskytovat všechna slova (například v jakémkoli pořadí, nebo při hledání slovních spojení, v přesně stejném pořadí, v jakém jsou slova napsaná v hledaném řetězci). U booleovského vyhledávání lze také hledat záznamy, které neobsahují určitá slova, nebo lze přidat modifikátor zástupného symbolu a tak hledat všechna slova, která začínají na zadanou předponu.

3.2 Popis realizace fulltextového vyhledávání

Při realizaci fulltextového vyhledávání je potřebné vytvořit indexy typu `FULLTEXT` a následně se spouští dotazy. Index typu `FULLTEXT` lze vytvořit různými způsoby. První možností

je definovat index už při vytváření tabulky příkazem `CREATE TABLE`, druhou možností je index následně přidat příkazem `ALTER TABLE` nebo `CREATE INDEX`. [6]

Při vyhledávání údajů z tabulky ze dvou sloupců dohromady i z každého sloupce zvlášť, je potřeba vytvořit separátní indexy pro oba sloupce a další index pro oba sloupce dohromady. Příkazem `ALTER TABLE` se následně přidají indexy:

```
ADD FULLTEXT (sloupec1),
ADD FULLTEXT (sloupec2),
ADD FULLTEXT (sloupec1, sloupec2);
```

Jakmile je tabulka nakonfigurována, lze spouštět dotazy, v nichž se využívá operátor `MATCH`. Pomocí tohoto operátoru lze stanovit, který sloupec nebo které sloupce se mají prohledávat a hledaný řetězec se zadá pomocí `AGAINST()`.

Při vyhledávání záznamu obsahujícího slova v různých sloupcích probíhá hledání ve všech sloupcích současně. Při vyjmenování všech sloupců u operátoru `MATCH` nezáleží na jejich pořadí, ale u jednotlivých sloupců musí existovat index `FULLTEXT`. [6]

```
SELECT * FROM produkt
WHERE MATCH (nazev, popisek)
AGAINST ('hledany_vyraz')
LIMIT 0 , 30
```

Pomocí klauzule `LIMIT` lze vybrat oblast řádků. Klauzule přebírá jeden nebo dva argumenty, což musí být celočíselné konstanty. `LIMIT` *m,n* přeskočí prvních *m* řádků a vrátí dalších *n* řádků. [6]

Počet záznamů, ve kterých se hledaný výraz vyskytuje, zjistíme pomocí `COUNT(*)`. [6]

Pokud se v klauzuli `WHERE` použije operátor `MATCH`, lze výstupní řádky seřadit sestupně podle jejich relevance. Hodnoty relevance jsou nezáporná čísla, přičemž nula vyjadřuje „irelevantní“. Pokud je žádoucí hodnoty relevance vidět na výstupu, je nutné uvést výraz `MATCH` v klauzuli `SELECT`. [6]

```
SELECT nazev_firmy, MATCH (nazev_firmy)
AGAINST ('hledany_vyraz')
AS relevance FROM firmy;
```

Výraz složený ze dvou slov vrátí záznamy obsahující kterékoli z uvedených slov. Počínaje verzí MySQL 4.0.1 lze zlepšit kontrolu nad vyhledáváním více slov pomocí booleovského fulltextového vyhledávání, které se děje tehdy, přidáme-li do `AGAINST()` klauzuli `IN BOOLEAN MODE`. [6]

Před verzí MySQL 4.0 bylo možné parametry fulltextového vyhledávání modifikovat pouze tak, že se provedla změna ve zdrojovém kódu a překompiloval server. MySQL od verze 4.0 poskytuje několik konfiguračních parametrů, jimiž lze proměnné serveru modifikovat. Nejpoužívanější jsou ft_min_word_len, která určuje nejkratší slovo a ft_max_word_len, které určuje nejdelší slovo, které se bude indexovat. Výchozí hodnoty jsou 4 a 254 a slova kratší než 4 znaky nebo delší než 254 znaků se budou při budování fulltextových indexů ignorovat. [6]

3.3 Varianty fulltextového vyhledávání, jejich omezení a porovnání

Existují tři typy fulltextového vyhledávání: Booleovské vyhledávání, vyhledávání přirozeným jazykem a vyhledávání pomocí rozšířeného dotazu.

3.3.1 Booleovské vyhledávání

V současné verzi MySQL 5.5 (od verze 4.0.1) lze realizovat Booleovské vyhledávání, které interpretuje vyhledávání slov podle pravidel dotazovacího jazyka. Může obsahovat operátory (`AND, OR, NOT`), které specifikují požadavky pro vyhledávání. MySQL vykonává booleovské fulltextové vyhledávání pomocí `BOOLEAN MODE` modifikátoru. [15]

Booleovské fulltextové vyhledávání má následující charakteristiky [6,15]:

- Vyhledávají se i slova, která se vyskytují více než v polovině záznamů.
- Výsledky se neřadí podle relevance, ale parser MySQL přiřazuje každému slovu nějakou „váhu“. Slova vyskytující se velmi často mají tuto váhu nižší než slova vyskytující se vzácně. Použitím znaku > lze v dotazu váhu slova snížit a použitím znaku < naopak váhu zvýšíme.
  ```
  SELECT * FROM tabulka WHERE MATCH(sloupec1,sloupec2)
  AGAINST('+slovo1 +(>slovo2 <slovo3)' IN BOOLEAN MODE)
  ```
  Uvedený dotaz vybere řádky obsahující slovo1 a slovo2 a řádky obsahující slovo1 a slovo3 budou mít nižší skóre.
- Výrazy začínající znakem ~ budou mít nejnižší „váhu“, ale nebudou zcela vyloučeny.
- U vyhledávaného výrazu se mohou používat modifikátory. Znaménkem plus nebo mínus je označeno, zda slovo má nebo nemá být přítomné ve shodujících se záznamech. Při booleovském vyhledávání odpovídajících záznamů k výrazu `'+hledane -slovo'`, vyhovují pouze záznamy, které obsahují `'hledane'`, ale neobsahují `'slovo'`.
- Při vyhledávání na přesnou shodu je nutné hledaný výraz uzavřít do uvozovek a tím se zobrazí pouze výsledky přesně odpovídající zadanému výrazu.
- Zástupný znak hvězdička způsobí, že bude vyhovovat každý záznam obsahující slova začínající

na hledané slovo. Vyhledávání pomocí zástupného symbolu nelze aplikovat u slov, která jsou kratší než minimální délka slova indexu.

```
SELECT * FROM Tabulka WHERE MATCH (sloupec1)
AGAINST ('slov*' IN BOOLEAN MODE);
```

- Například 'slov*', odpovídají 'slovo', 'slovosled'.
- Ignorují se pomocná slova (stop words), jakými jsou například frekventovaná anglická slova „the“, „after“ apod.
- Hledání slovních spojení je podporováno od verze 4.0.2 a může vyžadovat, aby byla slova v konkrétním pořadí. Slovní spojení musí být uzavřeno uvozovkami a zadáno přesně, včetně interpunkce a mezer. Verze MySQL 5.0.3 umožňuje vyhledávání stejných slov a ve stejném pořadí, ale mezi jednotlivými slovy může být vložen i jiný znak.
- Je možné prohledávat sloupce, které nejsou částí indexu `FULLTEXT`. Toto vyhledávání bude ale podstatně pomalejší, než když pracujeme se sloupci indexu.

3.3.2 Vyhledávání přirozeným jazykem

Vyhledávání přirozeným jazykem interpretuje hledaný řetězec jako frázi v přirozeném lidském jazyce a má následující charakteristiky [15]:

- Nepoužívá žádné speciální operátory.
- Standardně ignoruje slova kratší než 4 znaky, dále běžně vyskytující slova (stop words) a slova vyskytující se ve více než polovině záznamů, což je žádoucí, neboť při vyhledávání ve velkém množství záznamů nezobrazí uživateli každý druhý záznam.
- Sloupce určené pro vyhledávání musí být založeny na indexech typu `FULLTEXT`.
- Hledaný řetězec zadáme pomocí `AGAINST()` a operátoru `MATCH()`.
- Pokud se použije operátor `MATCH` v klauzuli `WHERE`, výstupní řádky se automaticky řadí sestupně podle jejich relevance. Operátor `MATCH` vrací desetinné číslo, jehož velikost odpovídá tomu, jak je výsledek relevantní. Pokud vyhovuje příliš mnoho záznamů, `MATCH` vrátí číslo 0. Následující dotaz nezobrazí relevantní výsledky:

  ```
  SELECT id, MATCH (nazev_firmy)
  AGAINST ('hledany_vyraz') FROM firmy;
  ```

  Pro zobrazení sestupné relevance výsledů je v tomto případě nutné specifikovat operátor `MATCH ()` dvakrát, jednou v klauzuli `SELECT` a jednou v klauzuli `WHERE`. Optimalizátor MySQL ale zjistí shodu vyhledávacích výrazů a zobrazí výsledek vyhledávání pouze jednou.

```
SELECT id, MATCH (nazev_firmy)AGAINST ('hledany_vyraz')
FROM firmy WHERE MATCH (nazev_firmy)
AGAINST ('hledany_vyraz');
```

- Relevance se vypočítává na základě počtu slov v řádku, počtu unikátních slov v daném řádku, celkovému počet slov v kolekci a počtu dokumentů (řádků), které obsahují určité slovo. Slovo, které se vyskytuje v mnoha dokumentech má nižší váhu (může být i nulová) než slovo, které se vyskytuje vzácně. Sečtené hodnoty vah jednotlivých slov ukazuje význam řádku.

3.3.3 Vyhledávání pomocí rozšířeného dotazu

Současná verze MySQL 5.5 (od verze 4.1.1) podporuje rozšířené fulltextové vyhledávání, které rozšiřuje přirozené fulltextové vyhledávání. Nejprve je pro hledaný řetězec použito přirozené vyhledávání a pak jsou výsledky z nejdůležitějších řádků vráceny zpět vyhledávání a hledání se provádí znovu. Dotaz zobrazí záznamy z druhého hledání. Rozšířením `QUERY` modifikátoru se určuje rozšíření dotazu vyhledávání. [15]

Tento způsob hledání je přínosný při vyhledávání implicitních výrazů. Například při vyhledávání slova „databáze“ by měly odpovídat záznamy obsahující „MySQL“, „Oracle“ nebo „DB“. Tato „automatická zpětná relevance“ je povolena přidáním `WITH QUERY EXPANSION` a způsobí, že se vyhledávání provede dvakrát. Jestliže jeden dokument obsahuje slovo „databáze“ a slovo „MySQL“ jsou při druhém hledání nalezeny dokumenty, které obsahují slovo „MySQL“, i když neobsahují slovo „databáze“. [15] Následující příklad ukazuje rozšířené fulltextové vyhledávání:

```
SELECT * FROM firmy
WHERE MATCH (nazev_firmy)
AGAINST ('hledany_vyraz' WITH QUERY EXPANSION);
```

Vzhledem k tomu, že vznikne „slepé rozšíření“ můžou se ve výsledných záznamech vracet nerelevantní dokumenty. Toto vyhledávání je smysluplné používat pouze při vyhledávání příliš krátkého výrazu. [15]

3.4 Porovnání typů fulltextového vyhledávání

Z výše uvedené charakteristiky typů fulltextového vyhledávání v MySQL vyplývají základní rozdíly, které ukazuje Tabulka 1.

Tabulka 1: Porovnání typů fulltextového vyhledávání [6,15]

Kriterium	Booleovské fulltextové vyhledávání	Vyhledávání přirozeným jazykem
Vyhledávání slov, která se vyskytují ve více než polovině záznamů	Vyhledává i slova vyskytující se ve více než polovině záznamů.	Ignoruje slova vyskytující se ve více než polovině záznamů.
Řazení výsledků dle relevance	Výsledky neřadí dle relevance.	Výsledky řadí dle relevance.
Možnost přidávat modifikátory k upřesnění vyhledávání	Lze použít modifikátory (+, -,*).	Nepoužívá žádné modifikátory.
Vyhledávání stop words	Stop words se ignorují.	Stop words se ignorují.
Požadavky pro vyhledávání	Booleovské vyhledávání může fungovat i bez indexu typu FULLTEXT, ale vyhledávání je pomalé.	Požaduje vytvořit indexy typu FULLTEXT.
Vyhledávání přesnou shodou	Lze vyhledávat přesnou shodou.	Lze vyhledávat přesnou shodou.

Kdy je výhodnější použít Booleovské vyhledávání

V případě, kdy je žádoucí vyhledávání složených výrazů, lze s výhodou použít Booleovské vyhledávání.

Přirozené fulltextové vyhledávání vyhledává každé slovo ze složeného výrazu. Např. při vyhledávání fráze „minerální odličovač“ zobrazí reference obsahující slovo minerální (např. minerální krém) a další reference obsahující odličovač (např. odličovač očí).

Booleovské vyhledávání umožňuje vyhledat reference obsahující všechna zadaná slova a zobrazí pouze tyto přesné výsledky. Slovní spojení musí být uzavřeno uvozovkami a doplněno modifikátorem +, jak je uvedeno v následujícím dotazu:

```
SELECT * FROM kosmetika_produkt
WHERE MATCH kosmetika_nazev
AGAINST ("+minerální +odličovač" IN BOOLEAN MODE)
LIMIT 0 , 30
```

Další výhodou používání Booleovského vyhledávání je používání zástupných znaků, které lze využít například při skloňování. Následující dotaz zobrazí reference obsahující slova krém, krémový, krémová atp.:

```
SELECT * FROM kosmetika_produkt
WHERE MATCH kosmetika_nazev
AGAINST ("krém*" IN BOOLEAN MODE)
LIMIT 0 , 30
```

Kdy je výhodnější použít Vyhledávání přirozeným jazykem

V případě, kdy je požadováno zobrazit výsledky dle relevance, je výhodné použít vyhledávání přirozeným jazykem. Reference následující dotazu budou seřazeny sestupně. Dotaz vrátí 30 nejlepších výsledků.

```
SELECT id_kosmetika_produkt,kosmetika_nazev,
MATCH(kosmetika_nazev)AGAINST ("krém")
FROM kosmetika_produkt
WHERE MATCH (kosmetika_nazev)
AGAINST ("krém")
LIMIT 0,30;
```

4 NÁVRH DATABÁZE

Pro správné zobrazení relevantních výsledků vyhledávání je důležité, aby data databázi byla správná a neredundantní. Tato kapitola si klade za cíl popsat základní fáze návrhu databáze.

Návrh databáze tvoří tři hlavní fáze nazývané konceptuální, logický a implementační návrh. Cílem konceptuálního návrhu je identifikovat důležité objekty, které je nutné reprezentovat v databázi a relace mezi těmito objekty a to bez jakýchkoli úvah o jejich fyzické implementaci. Technologický návrh představuje reprezentaci těchto objektů a jejich relací množinou tabulek, ale nezávisle na konkrétním RSŘBD a fyzické implementaci. Implementační návrh představuje vlastní implementaci v cílovém RSŘBD. [3,19]

4.1 Konceptuální návrh databáze

Datovému modelu na konceptuální úrovni – tzn. vstupní datové analýze - říkáme konceptuální schéma. Pro zobrazení konceptuálního schématu používáme ER diagram, neboli Entity Relationships Diagram. Jedná se modelování entit, jejich vztahů a jejich atributů.

Prvním krokem při vytvoření ER modelu je definování entit neboli rozlišitelných a identifikovatelných objektů reality a dále definování jejich atributů neboli vlastností entity. Specifikovaná množina hodnot, kterých může atribut nabývat je označována jako doména. Každá entita musí být jednoznačně identifikovatelná primárním klíčem. [19]

Entity vstupují do konkrétních vzájemných vztahů. Tyto vztahy mohou být binární (tj. mezi dvěma entitami) nebo vícenásobné (tj. mezi více entitami). Každý vztah má jméno zastoupené jmennou frází vyjadřující podstatu vztahu z hlediska obou partnerských entit. Mezi dvěma entitami může existovat jeden vztah nebo více vztahů různých typů a každý z těchto vztahů vyjadřuje jinou informaci. Vztah může mít své atributy, které daný vztah blíže charakterizují. [19]

Integrita databáze označuje správnost a konzistenci uložených dat. Integritní omezení jsou pravidla, která definují nebo omezují některé vlastnosti dat a jsou nezbytnou podmínkou pravdivosti a bezrozpornosti celku. [3]

Integritní omezení atributů

Integritní omezení atributů zachycuje vlastnosti atributů a požadavky na atributy v souladu s modelovanou databází.

Pro každý entitní typ se vypracuje tabulka, ve které se zachytí následující integritní omezení [20]:

- typ atributu – jde o zadání domény a množiny operací, které lze na doméně provádět a určení velikosti prostoru ve znacích,
- určení, zda je atribut klíčový,
- označení, zda atribut může mít prázdnou hodnotu (NULL),
- příznak, zda atribut musí mít unikátní hodnotu (UNIQUE),
- definice oboru přípustných hodnot domény,
- výchozí (default) hodnota atributu,
- omezující hodnota určující podmínky platnosti.

Integritní omezení vztahů

Integritním omezením vztahů lze specifikovat kardinalitu, parcialitu, exkluzivitu a externí identifikaci vztahu. Kardinalita vztahu vyjadřuje minimální a maximální počet výskytů entity v určitém vztahu. Parcialita (volitelnost) vztahu zachycuje členství ve vztahu. Pokud vztah musí vzniknout, je toto členství označováno jako povinné, pokud členství ve vztahu může vzniknout a nemusí, je označováno jako nepovinné (parciální). Exkluzivita (výlučnost) vztahu popisuje, zda pro jeden výskyt entity může být realizován právě jeden ze vztahů vzájemně výlučných. Externí identifikace vyjadřuje úplnou závislost určité entity v určitém vztahu na existenci jiné entity. [3,19]

Pro grafické vyjádření integritních omezení vztahů v rámci ER diagramu lze doplnit multiplicitu. Multiplicita se skládá ze dvou samostatných integritních omezení [3,19]:

- kardinality, která nabývá hodnot 0 (vyjadřuje nepovinné členství ve vztahu) a 1 (vyjadřuje povinné členství ve vztahu, tedy nutnost vzniku vztahu),
- parciality, která vyjadřuje maximální počet výskytů entity a nabývá hodnot 1 (výskyt je nejvýše jednou) a N případně M (výskytů je více).

4.2 Logický návrh databáze

V logickém návrhu databáze se mapuje (převádí) ER model do relačního modelu dat – RMD. RMD definuje způsob, jakým je možné data reprezentovat (strukturu dat), dále způsob jejich ochrany (integritu dat) a operace, které lze nad daty provádět (manipulace s daty). Charakteristickou vlastností relačního modelu je realizace vazeb pomocí relací v níž každou entitu vstupující do vazby zastupuje její primární klíč. [19]

Dle [19] „V relačním modelu dat vychází relace z matematického pojmu relace (anglicky Relation). Relace je množina prvků, které mají tvar (a1, a 2, …a n), kde *n* je řád relace. V relační terminologii se prvkům říká *n-tice*.“

4.2.1 Pravidla transformace

Pro převod ER modelu do RMD lze použít definovaná pravidla transformace. Transformací vzniknou první návrhy relací. Relace vznikají za určitých podmínek z entit a také ze vztahů mezi entitami. Entita se transformuje do jedné relace pouhým přepisem. Vztah je modelován principiálně jednou vztahovou relací a primární klíč je složen z primárních klíčů entit zúčastněných ve vztahu. Atribut nebo množina atributů, které jsou v jiné relaci primárním klíčem nebo jeho částí je označován jako cizí klíč. [20]

Při transformaci ER modelu do relací platí, že žádná hodnota primárního klíče nesmí být prázdná a každá n-tice z dané relace odkazující se na jinou relaci, se musí odkazovat na existující relaci. Dále se při transformaci ER modelu uplatňují integritní omezení atributů vztahů, jejichž důsledkem je počet vzniklých relací a určení primárního a cizího klíče relace. [20]

Pro vztah 1:1 v ER modelu platí následující pravidla transformace do RMD [20]:

- Při povinné participaci na obou stranách vznikne jediné schéma relace a jeden z primárních klíčů původních entit se stane primárním klíčem relace.
- Má-li jeden člen nepovinnou participaci a druhý povinnou (a je existenčně závislý na prvním členu), jsou definována dvě schémata relací. Primárním klíčem se stane libovolný ze dvou primárních klíčů původních entit.
- Při nepovinné participaci na obou stranách vzniknou tři schémata relací. Pro každý entitní typ vznikne jedno schéma a třetí schéma vznikne pro jejich vztah. Libovolný z primárních klíčů původních entit se stane primárním klíčem a třetí schéma bude obsahovat klíče obou předchozích jako cizí klíče.

Pro vztah 1:N v ER modelu platí následujíc pravidla transformace do RMD [20]:

- Při povinné participaci determinantu vztahu vzniknou dvě schémata, pro každý entitní typ jedno. Primárním klíčem se stane klíč determinantu.
- Má-li determinant vztahu nepovinnou účast ve vztahu, vzniknou tři schémata. Pro každý entitní typ vznikne jedno schéma a třetí schéma vznikne vztahové. Primárním klíčem zůstane klíč determinantu.

Pro vztah M:N v ER modelu platí následujíc pravidla transformace do RMD [20]:

- Pro každou binární relaci M:N definujeme bez ohledu na členství ve vztahu tři schémata

pro každý entitní typ jedno a třetí schéma vztahové. Primárním klíčem schématu se stane dvojice příslušných primárních klíčů.

4.2.2 Normalizace dat

Při základním návrhu mohou být v primární tabulce zachycena data, která se opakují a tím zabírají místo a komplikují a zpomalují práci s daty. Řešením je rozložení (normalizace) tabulky do dalších jednodušších tabulek – relací.

Normalizace dat je proces sloužící k odstranění anomálií a duplicit v datovém modelu. Důsledkem tohoto procesu je postupná dekompozice datového modelu, kterou se rozdělí atributy do většího počtu relací nevykazujících dané nedostatky. Postupně se množina všech relací převádí do tzv. vyšších normálních forem. [19]

Základních normálních forem je pět [14,19]:

- První normální formu (1NF) splňuje relace, která neobsahuje žádné atributy s násobnými hodnotami. Každý průsečík řádku a sloupce v relaci tak musí obsahovat nejvýše jednu datovou hodnotu.
- Ve druhé normální formě (2NF) se nachází relace, je-li v první normální formě a jestliže pro každý neklíčový atribut platí, že je funkčně závislý na celém primárním klíči relace a ne pouze na jeho části.
- Relace je v třetí normální formě (3NF), je-li ve druhé normální formě a platí-li, že v relaci neexistuje žádná tranzitivní závislost - tedy všechny neklíčové atributy relace jsou závislé pouze na jejím primárním klíči, žádný atribut nezávisí na jiném atributu závisícím na primárním klíči.
- Boyce-Coddovu normální formu (BCNF) splňuje relace, je-li ve třetí normální formě a v relaci (tabulce) nesmí existovat žádný určující atribut, jenž by byl primárním nebo kandidátním klíčem. Žádný neklíčový atribut nesmí jedinečně identifikovat hodnotu žádného jiného atributu, a to ani atributů, které se účastní definovaného primárního klíče.
- Ve čtvrté normální formě se nachází relace, je-li v BCNF a všechny vícehodnotové závislosti obsažené v relaci jsou zároveň funkčními závislostmi, jedná se o odstranění vícehodnotových atributů.
- Relace je v páté normální formě (5NF), pokud je v 4NF a nemůže být dále bezztrátově rozložena.

4.3 Implementační návrh

Implementační návrh představuje popis implementace databáze v konkrétním implementačním prostředí.

V průběhu implementace je nutné zajistit datovou integritu databáze – tj. správnost a konzistenci uložených dat. Lze rozlišit tři základní typy datové integrity. Doménová integrita definuje, jaká data mohou být ukládána do jednotlivých sloupců tabulek. Tabulková integrita specifikuje, že každý řádek tabulky má svůj primární klíč. Referenční integrita zajišťuje zachování vztahu mezi primárním klíčem a cizím klíčem v odkazující se tabulce. [19]

Datovou integritu lze vynutit dvěma způsoby – deklarativně (omezení při vkládání dat) nebo procedurálně (naprogramovat).

4.4 Správa databáze

Pro údržbu tabulek MyISAM v databázovém serveru MySQL lze použít následující příkazy SQL [9]:

- `ANALYZE TABLE` - poskytuje informace o interní správě indexů,
- `CHECK TABLE` - testuje tabulku a hledá chyby v její konzistenci,
- `OPTIMALIZE TABLE` - optimalizuje využití prostoru v tabulkách,
- `REPAIR TABLE` - pokusí se opravit vadné tabulky.

5 OPTIMALIZACE PRO FULLTEXTOVÉ VYHLEDÁVÁNÍ

Nejdůležitějším požadavkem pro fulltextové vyhledávání je rychlost zobrazování relevantních referencí – tj správných, odpovídajících výsledků vyhledávání, které se nejvíce shodují s požadavkem uživatele.

Důležitou podmínkou správně fungujícího fulltextového vyhledávání je kvalitní návrh databáze se správnou strukturou dat, sestavování efektivních SQL dotazů a zjišťování a následné odstraňování úzkých míst. Optimalizovaná databáze nemusí splňovat všechna kritéria normálních forem databáze, neboť díky redundanci dat lze často získat mnohem rychlejší reakci na naše dotazy. [20]

Optimalizaci lze provádět indexováním, optimalizací SQL dotazů a optimalizací serverových nastavení. Tato kapitola popisuje základní pravidla optimalizace výkonnosti databázového serveru pro vyhledávání a možné modifikace operačních parametrů serveru, aby mohl server pracovat efektivněji.

Mnoho webových aplikací se zpomaluje kvůli postupnému nárůstu velikosti databáze. Skutečné chování systému lze poznat v případě, kdy je databáze naplněna větším množstvím dat, tj. tisíce záznamů v databázi.

5.1 Indexování

Nejvýznamnějším nástrojem pro zrychlování dotazů je indexování. Indexy jsou datové struktury, které pomáhají MySQL získávat efektivně data a jsou nezbytné pro dobrý výkon. Indexy nabývají důležitosti s rostoucím množstvím dat. Špatně indexované schéma může mít značné negativní dopady na výkon. Největší rozdíl ve výkonnosti lze docílit správným používáním indexů, a proto se při optimalizaci nejprve použijí indexy a teprve potom se zjišťuje, zda lze využít i další prostředky. Pokud je požadován vysoký výkon, je nutné schéma i indexy navrhnout s ohledem na specifické spouštěné dotazy. Optimalizace často znamená kompromis, neboť změny v jednom dotazu mohou mít důsledky jinde. Pokud jsou například přidány indexy k urychlení získávání dat, mohou se zpomalit například aktualizace. [6,18]

Vytvořením indexu tabulky lze vytvořit jakousi mapu, která se využije při vyhledávání záznamů. Další výrazné úspory zdrojů lze dosáhnout při spojování tabulek. Místo spojení celkového objemu dat dojde pouze ke spojení indexů. Následně jsou aplikovány všechny omezující podmínky (např. v části klauzule `WHERE`) a teprve potom je jako výsledek vygenerována odpověď se všemi požadovanými sloupci. [20]

U tabulek MyISAM se řádky dat tabulky udržují v souboru dat a hodnoty indexu se udržují v indexovaném souboru. Pro tabulku může být vytvořeno více indexů. Všechny indexy jsou uloženy v tomtéž indexovaném souboru a skládají se ze seřazeného pole klíčů záznamů, které se používají pro rychlý přístup do souboru dat. Indexy lze použít pro urychlení vyhledávání řádků, které jsou v souladu s požadavky klauzule `WHERE` nebo z odpovídajících řádků z jiných tabulek (v případě spojení tabulek), dále pro řazení nebo seskupování, ale ne pro sloupce, které se zobrazují na výstupu. Indexy za účelem zvýšení výkonnosti je efektivní použít v také dotazech s výskytem funkce `MIN()` nebo `MAX()`, dále k urychlení operací řazení a seskupování v klauzulích `ORDER BY` a `GROUP BY`. Je doporučeno používat jedinečné indexy, indexovat krátké hodnoty a využívat hodnot levých prefixů. Indexy lze také s výhodou použít při porovnávací operace a operace s klauzulí `BETWEEN` a `LIKE`. [6]

Při použití běžných indexů je možné, aby několik záznamů mělo v indexovaném sloupci stejné hodnoty. Index, který vylučuje výskyt jedné hodnoty v daném sloupci vícekrát lze definovat klíčovým slovem `UNIQUE`. Tím lze zajistit, že do databáze nevložíme jeden záznam se stejným údajem vícekrát. [9]

Ačkoliv je indexování velmi výhodné, může mít i nevýhody. Nadměrné množství indexů zabírá více místa na disku a lze se tak dostat na horní mez velikosti tabulky. U tabulek MyISAM může nadměrné množství indexů způsobit dosažení maximální velikosti indexovaného souboru dříve než soubor dat. Indexy zrychlují získávání dat, ale zpomalují operace vkládání a odstraňování a také aktualizace hodnot indexovaných sloupců, což lze uvést jako další nevýhodu indexování. Obecně indexy zpomalují většinu operací, které obsahují zapisování. Je to způsobeno zápisem řádku do souboru dat a tím i změnou všech indexů. Čím více indexů tabulka má, tím více změn musí vykonat. [6]

5.2 Pravidla pro optimalizaci dotazů

Mezi obecná pravidla pro psaní SQL dotazů patří [20]:

- Vyjmenovat všechny sloupce místo zástupného znaku hvězdičky. Při uvedení hvězdičky musí databáze nejprve zjistit, jaké sloupce tabulka obsahuje, ale při jejich vyjmenování se tím nemusí zabývat.
- Používat co nejméně klauzuli `LIKE`, neboť vyhledávání substringů[3] je velmi náročná činnost na výkon a rychlost reakce serveru.

[3] Substring je textový podřetězec, který odpovídá zadanému vzorku v klauzuli LIKE. [6]

- Na začátku formulovat obecné podmínky, po kterých vypadne ze seznamu nejvíce záznamů. Nejprve jsou vyhledány záznamy odpovídající první podmínce a pak z těchto záznamů jsou vybrány ty, které odpovídají druhé podmínce.
- Používat klauzuli `LIMIT` a tím omezit výběr záznamů.
- Při spojování tabulek používat spojení pouze indexovaných sloupců. V případě složených indexů vybrat takový sloupec, který načte z tabulky co nejméně záznamů.
- Omezit používání operátorů `OR` a `IN` při kterých je v případě nesplnění podmínky vyhledávání nutné testovat ještě další podmínky. Při použití `AND` stačí, aby jedna podmínka nebyla splněna, a celý záznam bude z výsledné odpovědi vyřazen.
- Využít vnořených dotazů a vytvářet tak dotazy, které zpracovávají větší množství položek v jednom kroku. Při zpracování méně položek ve více krocích lze výrazně zatížit server.

Mezi pravidla pro optimalizaci rychlosti vyhledávání patří [9]:

- Vyhnout se dotazům, které vracejí 1000 záznamů a více. Zatížení nezpůsobuje jejich skutečné spouštění, ale přenos výsledných dat na interpretr PHP a následné vyhodnocování vrácených výsledků v kódu PHP. Je důležité třídit data již na serveru MySQL pomocí vhodných podmínek výběru do dotazů `SELECT`.
- Nepsat v PHP kód pro vyhledávání nebo třídění dat a potřebná kriteria výběru a pravidla třídění zabudovat přímo do příkazu `SELECT`.

Mezi další doporučení patří [17]:

- Po změně nastavení fulltextového vyhledávání nebo vytvoření fulltextového indexu je důležité vyčistit cache dotazů.
- Indexované sloupce používat na pozicích co nejvíce vlevo – levý prefix
- Používat nejnovější verzi MySQL, neboť nové verze mají obecně výkonnější fulltextové vyhledávání.

Pro zhodnocení vytvořeného dotazu lze použít příkaz `EXPLAIN`. Tento příkaz lze použít jako předponu běžného dotazu SQL (např. `EXPLAIN SELECT`). Místo výsledků dotazu se zobrazí tabulka s informacemi o tom, jak je příkaz prováděn a jaké indexy se zapojily. V tabulce jsou uvedeny názvy tabulek, typ propojení s ostatními tabulkami, klíče, které mohou sloužit pro vyhledávání záznamů, dále použitý index a jeho délka, odkazy na další tabulky a počet záznamů, které MySQL přečetl, aby provedl dotaz. Součin všech čísel ve slopci *rows* ukazuje odhad celkového počtu kombinací řádků, které se musejí ze všech tabulek prozkoumat, tj. ukazuje

složitost dotazu. Poslední sloupec obsahuje další informace o provedeném dotazu (např. vytvoření pomocné tabulky). [9,6]

MySQL disponuje nástrojem pro monitoring dotazů a optimalizaci indexů. Tento nástroj se nazývá Optimalizátor dotazů a jeho primárním cílem je využít indexy vždy, když je to možné a co možná nejvíce použít restriktivní index pro eliminaci řádků a tím rychleji najít ty řádky, které splňují zadaná kriteria. [6]

5.3 Optimalizace serverových nastavení

Optimalizaci lze provádět také pomocí modifikace operačních parametrů databázového serveru. Nejdůležitější parametry, které lze změnit, jsou velikost vyrovnávací paměti tabulky a vyrovnávacích pamětí, které používají zpracovatelé tabulek k operacím s indexy. Dostupnou paměť lze alokovat na vyrovnávací paměť serveru, aby bylo možné informace v paměti udržovat déle a tak redukovat činnost s diskem. Rychlejší je přístup k informacím v paměti než jejich načítání z disku. [6]

Konfiguračních nastavení existuje několik oborů – na úrovni serveru (globální), nebo pro různá připojení (relace, session) nebo pro každý objekt. Proměnné se nenastavují pouze v konfiguračním souboru, ale mnohé z nich lze nastavovat i při běhu serveru, jako dynamické konfigurační proměnné. [18]

Existuje několik možností nastavení konfiguračních proměnných. Například server určený pro fulltextové hledání potřebuje dostatečně velké buffery klíčů, aby se do nich vešly fulltextové indexy. Pokud jsou indexy (nikoliv data) umístěny v paměti, pracují mnohem lépe. Je tedy nutné pro cache alokovat velké množství paměti pomocí proměnné `key_buffer_size`. Pro potřeby setřiďování je výhodné alokovat paměť pomocí proměnné `sort_buffer_size`. Nastavení proměnné `query_cache_size` alokuje a inicializuje paměť pro cache dotazů v okamžiku, když server startuje. [18]

Při nastavování proměnných je nutné si uvědomit, že vyšší hodnota neznamená vždy výkonnější. Někdy zvýšení proměnné může vyvolat i nesoulad mezi MySQL, operačním systémem a hardwarem. Nastavení proměnných je třeba testovat postupně, tak aby byly v souladu a pomocí monitorovacího systému sledovat výkonnost. [18]

6 NÁVRH FULLTEXTOVÉHO VYHLEDÁVÁNÍ INTERNETOVÉHO OBCHODU

S rozvojem internetu došlo i k rozvoji internetového obchodování, jehož historie sice u nás sahá jen do několika posledních let, ale získává si stále větší oblibu.

Internetový obchod je webová aplikace sloužící k prodeji zboží. V internetovém obchodě lze získat informace o prodávaném zboží a o jeho výrobci. Současně existuje možnost si vybrané zboží prostřednictvím internetu zakoupit. Internetový obchod umožňuje snadný a rychlý nákup téměř odkudkoliv, kde je připojení k internetu.

Mezi výhody internetového obchodování oproti klasickým kamenným obchodům patří možnost prohlédnout si nabídku několika různých e-shopů a posoudit jejich možnosti, ceny a kvalitu nabízeného zboží v pohodlí svého domova. Zřejmě největším důvodem rozvoje internetového obchodování je možnost nákupu v kteroukoliv denní dobu, větší výběr zboží s možností porovnání cen a tím možnost finanční úspory a dodávka zásilky objednaného zboží přímo do rukou zákazníka.

Vybrané zboží si může potenciální zákazník i osobně zakoupit v kamenné provozovně internetového obchodu. Někteří zákazníci totiž mají obavy o bezpečnost finančních transakcí, ztráty soukromí vyplněním registračních údajů, doručení nepoškozeného balíčku nebo mají potřebu zboží před zakoupením vidět nebo vyzkoušet.

K tomu, aby mohl zákazník požadované zboží v některém e-shopu zakoupit, předchází vyhledávání zboží. Pokud zboží nebude nalezeno, nemůže být uskutečněn jeho nákup. Je důležité, aby vyhledávání bylo jednoduché, rychlé a zobrazovalo co možná nejpřesnější (relevantní) nalezené reference dle zadaného výrazu.

Tato bakalářská práce dále popisuje fulltextové vyhledávání internetového obchodu společnosti VIVANTIS a.s. (http://www.vivantis.cz), která je přední internetový prodejce v České republice. Na trhu působí od roku 2001 a provozuje specializované internetové obchody zaměřené především na prodej hodinek, parfémů, šperků a produktů pro zdraví i krásu. Jedním z těchto internetových obchodů je i Krasa.cz (http://www.krasa.cz) s nabídkou 6200 produktů od 130 značek.

6.1 Nároky na fulltextové vyhledávání

Nároky na fulltextové vyhledávání internetového obchodu Krasa.cz vycházejí ze základního poslání každého vyhledávače a to je zobrazovat relevantní výsledky vyhledávání dle zadaného výrazu v co možná nejkratším čase.

Současný stav

Při zadání výrazu k vyhledávání může dojít k překlepu uživatele, které by mohlo způsobit, že žádné výsledky nebudou nalezeny. V tomto případě je uživateli oznámeno, že se zřejmě jedná o překlep a zobrazí se mu nabídka s opraveným nebo podobným výrazem, který je dále určen k vyhledávání.

Při zobrazení výsledků vyhledávání zadaného výrazu je uživateli nabídnuto roztřídění dle kategorií, které konkrétní produkt obsahují. Uživatel si může zvolit kategorii, ze které si přeje zvolený produkt vyhledat. Pod nabídkou kategorií se zobrazuje tzv. průvodce, který slouží k filtrování výsledků. Uživatel má možnost např. zaškrtnout určení pohlaví, způsob řazení výsledků nebo značku produktu. Pod průvodcem jsou zobrazeny všechny výsledky vyhledávání produktů ze všech kategorií.

Vyhledávání probíhá ve dvou úrovních přesnosti. Nejprve probíhá vyhledávání dle přesné shody se zadaným výrazem a poté následuje vyhledávání volnou shodou.

Vyhledávání by mělo zobrazovat relevantní výsledky dle určených vah pro vyhledávání. Nejvyšší váhy jsou přiděleny značce produktu, následně názvu produktu, dále názvu řady, popisku a popisu. Výstupem je seřazení výsledků podle celkového dosaženého hodnocení relevance, nejvýše jsou zobrazeny výsledky z prohledávání ve značce produktu, pokud již žádné další výsledky nejsou nalezeny, tak se zobrazí výsledky na základě shody s názvem produktu, dále následují výsledky obsahující výraz v názvu řady, v popisku a v popisu produktu.

Relevance je číslo, které uvádí velikost shody výsledku vyhledávání s výrazem, který je požadováno vyhledat. Čím je toto číslo větší, tím je větší relevance. Podle tohoto čísla je prováděno řazení výsledků. K zobrazení výsledků vyhledávání si uživatel může zvolit počet výstupů na stránce.

V některých případech lze při zobrazení výsledků vyhledávání zjistit, že reference neodpovídají zcela přesně požadovanému výsledku. Z tohoto důvodu bude součástí této práce otestování SQL dotazu se snahou nalézt jiné řešení, které bude zobrazovat relevantní reference.

Návrh řešení

Kromě popsaných problémů u stávajícího stavu fulltextového vyhledávání internetového obchodu Krasa.cz by mělo nově vytvořené řešení také postihovat specifické případy, které jsou popsány dále v této kapitole.

Někteří uživatelé chtějí vyhledat produkt dle jeho názvu, ale již si přesně nepamatují celý název. Těmto uživatelům může být vyhledávání usnadněno pomocí tzv. našeptávače, který dle zadání prvních písmen do políčka fulltextového vyhledávače nabízí možné názvy produktů, které jsou k dispozici.

Ve zvláštním případě může být uživatelem zadán zvláštní výraz (např. česky napsaný anglický výraz), na který by nebyl nalezen žádný relevantní výsledek, ale v databázi odpovídající (např. anglicky napsaný) výraz existuje. Pro tento případ lze vytvořit tzv. slovník, ve kterém se zadaný výraz přiřadí odpovídajícímu výrazu z databáze a následně pokračuje vyhledávání identické s případem, kdy by uživatel zadal výraz správně.

Pro pracovníky obchodního oddělení je důležité zjistit, jaké produkty uživatelé nejčastěji vyhledávají. Mohou tak zjistit, o které produkty mají uživatelé zájem a eventuelně chybějící produkty zařadit do nabídky. K tomuto účelu lze v databázi vytvořit tabulku, do které jsou ukládány výrazy, které uživatelé hledali a také počet jejich hledání.

6.2 Struktura databáze

K zobrazení relevantních výsledků vyhledávání je nutný správný návrh databáze. Dále je nutné databázi optimalizovat pro fulltextové vyhledávání, aby vyhledávání probíhalo co nejrychleji.

Pro správu databázového serveru MySQL 5.1.36 byla použita MySQL Workbench CE for Windows 5.2.31, která nabízí tři hlavní oblasti funkčnosti:

1. SQL editor - umožňuje vytvářet a spravovat připojení k databázovému serveru
2. Datové modelování - umožňuje vytvářet modely databáze
3. Správa serveru - umožňuje vytvářet a spravovat serverové instance

Návrh řešení

Do databázové serveru MySQL byla importována databáze internetového obchodu Krasa.cz se stávající databázovou strukturou. Tato struktura byla podrobena kontrole z hlediska integritního omezení, normalizace dat a referenční integrity. V případě, kdy budou nalezeny závažné nedostatky, bude při jednotlivých bodech databázová struktura změněna takovým způsobem, aby byly tyto nedostatky odstraněny.

Pro realizaci fulltextového vyhledávání v MySQL je nutné použít úložný engine MyISAM, který podporuje fulltextové vyhledávání, jak již bylo zmíněno v kapitole 2.3. Tento úložný engin je používán také na serveru e-shopu Krasa.cz.

Pro potřeby zápisu výrazů, které uživatelé vyhledávají, byla v testovací databázi vytvořena nová tabulka kosmetika_fulltext. V této tabulce je také uveden počet vyhledávání daného výrazu.

Pro účely tzv. slovníku - neboli vyhledávání speciálních výrazů byla vytvořena tabulka kosmetika_slovnik_vyraz obsahující seznam několika zvláštních výrazů. V případě, kdy uživatel zadá k vyhledávání některý z těchto výrazů, přiřadí se k tomuto výrazu odpovídající výraz z tabulky kosmetika_slovnik_slova. Tato, také nově vytvořená tabulka, obsahuje ekvivalentní výrazy, které se nacházejí v databázi. Odpovídající výraz ekvivalentní zadanému výrazu je následně předán k vyhledávání, v databázi jsou nalezeny odpovídající záznamy a uživateli se zobrazí požadované výsledky.

V dalším kroku byl proveden popis a grafické znázornění databázové struktury včetně nově vytvořených tabulek.

Integritní omezení

Příloha 1 popisuje entity databáze, která je využívána při fulltextovém vyhledávání a je použita pro otestování vlastního návrhu fulltextového vyhledávání. Jsou zde uvedeny potřebné entity včetně jejich primárních klíčů a také vztahy, do kterých jednotlivé entity vstupují. Tyto vztahy jsou specifikovány parcialitou vyjadřující povinné (1) nebo nepovinné (0) členství ve vztahu a také kardinalitou vyjadřující jeden výskyt entity (1) nebo více výskytů (N případně M) entity ve vztahu.

Příloha 2 ukazuje ER diagram části struktury databáze, ve kterém jsou zobrazeny vztahy entit a jejich atributy.

Relační model databáze (formát IDEF1X), který prezentuje Příloha 3, zobrazuje fyzické rozložení tabulek s výslednými vztahy relací a kardinalitou. Záznamy, které musí mít vyplněny hodnotu, tj. mají atribut `NOT NULL`, jsou označeny příznakem <M>. Primární klíč je identifikován příznakem <pi> a je podtržen.

Pro datové modelování byl zvolen CASE nástroj PowerDesigner od společnosti Sybase.

Každý záznam v tabulce je určen primárním klíčem, který je jedinečným identifikátorem záznamu tabulky a neobsahuje hodnoty `NULL` (tj. prázdné). Primární klíč obsahuje atribut `AUTO_INCREMENT`, který má za následek automatické zvětšování hodnoty při vložení každého nového záznamu. Toto neplatí pro tabulky f_url a kosmetika_zarazeni, které obsahují složený primární klíč. Tabulka kosmetika_zarazeni má primární klíč složený z produtkID a kategorieID.

Tabulka f_url obsahuje identifikátor složený z contentID, pageID a languageID a tyto tři identifikátory dohromady určují primární klíč každého záznamu tabulky.

Tabulky neobsahují duplicitní nebo vícehodnotová pole a také neobsahují pole, jejichž hodnota by byla zjistitelná jako kombinace jiných polí (např. vypočítaná hodnota).

Normalizace dat

Všechny tabulky splňují první normální formu, neboť obsahují pouze atomické atributy. Všechny atributy jsou závislé na primárním klíči a žádný atribut nezávisí na jiném atributu závisícím na primárním klíči a tím je splněna druhá a také třetí normální forma procesu normalizace dat. Tabulky splňují Boyce-Coddovu normální formu, neboť žádný neklíčový atribut jedinečně neidentifikuje hodnotu jiného atributu.

Referenční integrita

V relačních databázích je referenční integrita zajištěna pomocí cizího klíče, který je navázaný na primární klíč. Tím lze zajistit, že při smazání rodičovského záznamu s primárním klíčem dojde i k vymazání všech propojených záznamů v tabulkách obsahující cizí klíč.

V MySQL je možné cizí klíče definovat obdobně jako v ostatních relačních databázích pomocí klauzule `FOREIGN KEY`. Jak již ale bylo uvedeno v kapitole 2.3, tabulky by ale musely mít engine InnoDB. Jelikož fulltextové vyhledávání vynucuje úložný engine MyISAM, nelze cizí klíče definovat. Klauzuli `FOREIGN KEY` je možné definovat, formálně se přijme, ale je ignorována. [6]

Odstranit záznam z této databáze tedy není možné pouze jedním příkazem pomocí integritního omezení daného cizím klíčem, který zajistí odstranění i všech navazujících záznamů, ale je nutné odstranit pomocí několika dotazů i všechny napojené záznamy.

Konzistence tabulek je zajištěna na aplikační vrstvě pomocí php skriptů. Další možností je použití triggerů v databázi.

6.3 Základní optimalizace pro fulltextové vyhledávání

Nejdůležitějším požadavkem fulltextového vyhledávání je rychlé zobrazení nalezených relevantních výsledků. Cílem optimalizace je urychlit prohledávání a získávání
dat. Optimalizaci je možné provádět indexováním, optimalizací dotazů a optimalizací serverových nastavení, jak bylo popsáno v kapitole 5.3.

Indexování

V databázi jsou definované indexy pro sloupce, které se používají pro specifikaci omezujících podmínek (v klauzuli `WHERE`) a dále pro sloupce zajišťující řazení nebo seskupování (v klauzulích `GROUP BY`, `ORDER BY`). Indexy tyto operace potřebné pro fulltextové vyhledávání zrychlují, na druhou stranu ale zpomalují operace vkládání, odstraňování a aktualizace.

Indexy pro primární klíče jsou vytvářeny v případě MySQL implicitně (platí pro úložné enginy InnoDB i MyISAM).

Pro realizaci fulltextového vyhledávání je nutné vytvořit indexy typu `FULLTEXT` u sloupců, ze kterých probíhá fulltextové vyhledávání. Fulltextové vyhledávání probíhá z následujících tabulek: kosmetika_produkt, kosmetika_firmy, kosmetika_rada, kosmetika_odstiny, kosmetika_kategorie.

Rozložení fulltextových indexů ukazuje následující Tabulka 2.

Tabulka 2: Přehled indexů typu FULLTEXT, zdroj: [vlastní]

Název tabulky	**Název pole s indexem typu FULLTEXT**
kosmetika_produkt	kosmetika_nazev, kosmetika_popisek, kosmetika_popis, artikl_kod
kosmetika_firmy	nazev_firmy
kosmetika_rada	rada_nazev
kosmetika_odstiny	odstin_nazev
f_url	url, name
kosmetika_kategorie	fulltext_nazev

Pro ukázku bylo provedeno provedení rychlosti vykonání jednoduchého SQL dotazu, ve kterém jsou použity indexy s rychlostí vykonání SQL dotazu, ve kterém indexy nejsou použity. Tento test byl proveden ve Workbench CE for Windows 5.2.31.

Jak již bylo uvedeno, vytvoření fulltextových indexů je nezbytnou podmínkou k umožnění korektní funkčnosti fulltextově orientovaných dotazů (klauzule `MATCH (sloupec) AGAINST('hledany_vyraz')`. Testování se tak zabývá výhradně vlivem existence/neexistence standardního typu indexu na rychlost provádění některých dotazů.

Výsledky testování uvádí průměrný čas z 10 provedených pokusů, které byly vybrány vždy z celkových 12 pokusů a to tak, že výsledky s minimálním a maximálním časem byly ze sady odstraněny s cílem dále zmírnit statistickou chybu, která může vzniknout v důsledku zkreslení

vlivem vnějších faktorů. Mezi tyto faktory ovlivňující měření lze zařadit např. nerovnoměrné vytížení procesoru a jiných prostředků počítače jinými procesy v systému v průběhu testování.

Nejprve bylo provedeno otestování dotazů bez indexů, následně byly přidány indexy příkazem

```
ALTER TABLE Tabulka ADD INDEX (nazev_sloupce);
```

Indexy lze opět odstranit příkazem

```
ALTER TABLE Tabulka DROP INDEX nazev_sloupce;
```

První z následně uvedených dotazů provádí seskupování výsledků dle čísla kategorie produktu, druhý dotaz řazení výsledků sestupně dle čísla kategorie produktu. Indexy typu `FULLTEXT` jsou definovány pro sloupce, ze kterých probíhá vyhledávání – tj. sloupce id_kosmetika_produkt, kosmetika_kod, cena_kosmetika a kosmetika_nazev. Index byl definován a následně odebrán u sloupce id_firmy_c, který není primárním klíčem a definuje kategorii produktu.

SQL dotaz 1:

```
SELECT id_kosmetika_produkt, kosmetika_kod, cena_kosmetika
FROM kosmetika_produkt WHERE MATCH kosmetika_nazev
AGAINST ("hydratační krém" IN BOOLEAN MODE)
GROUP BY id_firmy_c
```

Průměr z 10 pokusů měření vykonání dotazu s definovaným indexem je 0,031 sec, bez definovaného indexu bylo naměřeno 0,047 sec. Procentuální rozdíl tedy činí přibližně 51,6 % ve prospěch varianty s existujícím indexem.

SQL dotaz 2:

```
SELECT id_kosmetika_produkt, kosmetika_kod, cena_kosmetika
FROM kosmetika_produkt WHERE MATCH kosmetika_nazev
AGAINST ("hydratační krém" IN BOOLEAN MODE)
ORDER BY id_firmy_c
```

Průměr z 10 pokusů měření vykonání dotazu s definovaným indexem je 0,062 sec, bez definovaného indexu bylo naměřeno 0,078 sec. Procentuální rozdíl tedy činí přibližně 25,8 % ve prospěch varianty s existujícím indexem.

Zřejmě největší rozdíl lze zjistit u následujícího dotazu (není fulltextové vyhledávání):

```
SELECT * FROM kosmetika_produkt WHERE artikl_kod = "5051"
```

Průměr z 10 pokusů měření vykonání dotazu s definovaným indexem je 0,0013 sec, bez definovaného indexu bylo naměřeno 0,078 sec. Procentuální rozdíl tedy činí přibližně 6000 % ve prospěch varianty s existujícím indexem.

Z testu je zřejmé, že došlo pomocí definovaných indexů k navýšení rychlosti vykonání dotazu.

Optimalizace dotazů

Při tvorbě SQL dotazu pro fulltextové vyhledávání byla snaha dodržet pravidla pro optimalizaci dotazů, která byla uvedena v kapitole 5.2. Například v klauzuli `SELECT` jsou vyjmenovány pouze sloupce, ze kterých probíhá zobrazení výsledků na výstupu místo použití zástupného znaku *, dále v dotazech není použita klauzule `LIKE`, spojení tabulek je realizováno prostřednictvím indexovaných sloupců atp.

Pro zhodnocení vytvářeného SQL dotazu byla používána funkce `BENCHMARK()`, která vyhodnotí n-krát určitý výraz a následně zobrazí čas, který ukazuje rychlost vyhodnocení daného výrazu serverem. Ukázka dotazu:

```
SELECT BENCHMARK(1000, 'krém*');
```

K vyhodnocování celého SQL dotazu byl používán příkaz `EXPLAIN`, který byl již popsán v kapitole 5.2.

Optimalizace serverových nastavení

Testování probíhalo na lokálním webovém serveru Apache HTTP Server 2.2.11, PHP 5.3.0 a databázovém serveru MySQL 5.1.36, které byly nainstalovány na počítači s operačním systémem Windows 7 (64 bit) s procesorem Intel® Core™ 2 Duo 3GHz a s operační pamětí 4 GB RAM.

Vzhledem k požadavku pro indexování slov o minimální délce 3 znaky byl v konfiguračním souboru MySQL doplněn následující řádek do sekce [`mysqld`]

```
ft_min_word_len = 3
```

K aktivaci tohoto nového parametru, který již byl popisován v kapitole 3.2 bylo nutné aktualizovat indexy typu `FULLTEXT`. Indexy lze odstranit a vytvořit znovu nebo pro všechny tabulky, které mají indexy typu `FULLTEXT` použít následující příkaz, který zajistí následné automatické používání nového parametru:

```
REPAIR TABLE Tabulka USE_FRM;
```

Následně byly všechny již vytvořené, ale i nově definované indexy nastaveny na novou hodnotu.

6.4 Otestování návrhu fulltextového vyhledávání

Pro účely testování fulltextového vyhledávání byla vytvořena aplikace, jejíž úvodní stránku zobrazuje Obrázek 3.

Obrázek 3: Testovací aplikace, zdroj: [vlastní]

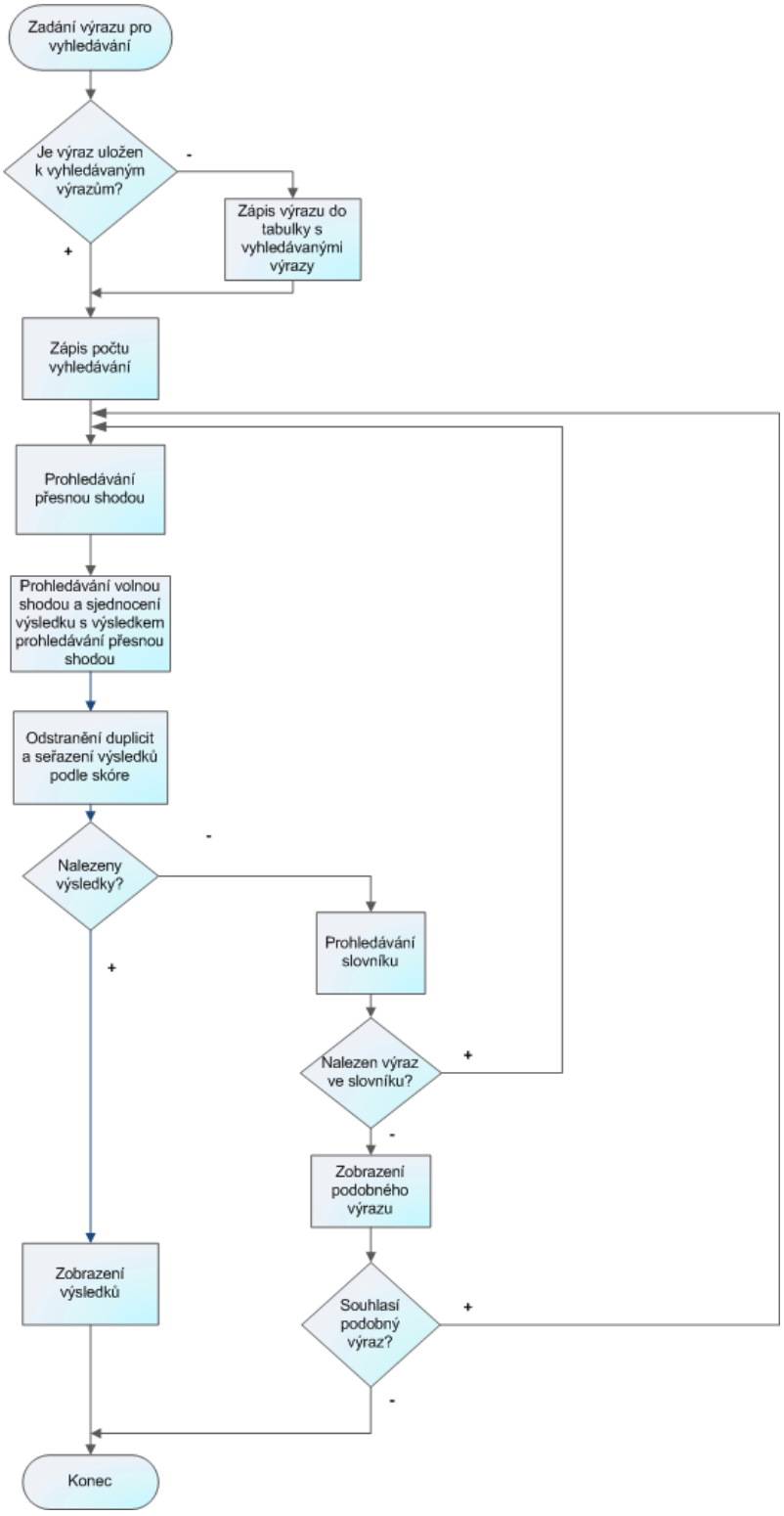
Tato aplikace zobrazuje fulltextové vyhledávací textové pole a dále pole pro výpis obsahu vyhledaných výrazů. Ve výpisu je zobrazena značka produktu, název produktu, kód produktu, odstín, kategorie, řada, popisek a popis produktu.

Fulltextové vyhledávání e-shopu Krasa.cz zobrazuje pouze produkty, které jsou online, neboli jsou dostupné. Toto je specifikováno jako jeden z atributů tabulky kosmetika_produkt, konkrétně se jedná o atribut `online` této entity.

Znamená to, že vyhledávání zobrazuje pouze odkazy na produkty, které je možné následně zákazníkem objednat a zákazníkovi odeslat. Aktualizace dat probíhá prostřednictvím administrace. Textové soubory od dodavatelů, obsahující dostupné zboží, jsou zpracovány php skripty. Tyto skripty zajistí „zapnutí“ produktu (hodnota 1) a jeho zobrazení na internetu nebo „vypnuti“ produktu (hodnota 0), pokud je nedostupný. Pro případ hromadného přenosu dat od dodavatelů je využíváno XML souborů. Toto zpracování umožňuje variabilitu zpracování dle druhu zboží a možnost využití dat i z jiných systémů.

6.4.1 Algoritmus fulltextového vyhledávání

Algoritmus realizace zobrazení výsledků dle zadaného výrazu, vytvořený v Microsoft Visio 2007, ukazuje následující Obrázek 4.



Obrázek 4: Algoritmus fulltextového vyhledávání, zdroj:[vlastní]

Prvním krokem pro realizaci fulltextového vyhledávání je zápis výrazu určeného k vyhledávání do textového pole. Při psaní je uživateli podsouván pomocí našeptávače možný hledaný výraz. Našeptávač nabízí výraz z pole název značky a název produktu. Podrobnější popis našeptávače ne uveden v kapitole 6.4.3.

V dalším kroku proběhne test, zda byl vyhledávaný výraz již vyhledáván a je uložen v tabulce kosmetika_fulltext, která obsahuje vyhledávané výrazy s počtem jejich vyhledávání. Pokud vyhledávaný výraz již v tabulce existuje, zvýší se hodnota počtu hledání o jednotku, pokud vyhledávaný výraz v tabulce neexistuje, dojde k jeho uložení s počtem vyhledávání jedna.

V tomto bodě již začíná vlastní realizace fulltextového vyhledávání, které dále probíhá ve dvou úrovních. Nejprve probíhá vyhledávání přesnou shodou se zadaným výrazem a následuje vyhledávání volnou shodou se zadaným výrazem. Oba způsoby vyhledávání jsou podrobněji popsány v kapitole 6.4.2

Výsledky obou dotazů jsou množinově sjednoceny, duplicity odstraněny (s konečným stavem tak, že pro *n* duplicit před sjednocením odpovídá po sjednocení a odstranění duplicit právě 1 záznam) a výsledky jsou seřazeny sestupně podle ohodnocení relevance.

V případě, že nebudou nalezeny žádné výsledky, je v dalším kroku zadaný výraz odeslán k porovnání do tabulky kosmetika_slovnik_vyraz, která obsahuje nestandardní výrazy. K zadanému výrazu je následně přiřazen ekvivalentní výraz z tabulky kosmetika_slovnik_slovo. Příkladem nestandardního výrazu může být výraz napsaný v cizím jazyce nebo netypické označení zboží, výrobce apod. Po vyhledání odpovídajícího výrazu probíhá opět vyhledávání tohoto výrazu přesnou shodou a následně volnou shodou. Např. pokud uživatel odešle k vyhledávání výraz Viši, nejsou nalezeny v databázi žádné výsledky, ale ve slovníku je tomuto výrazu přiřazen výraz Vichy. Následuje vyhledávání výrazu Vichy stejným postupem, jako v případě, kdy by uživatel zadal přímo výraz Vichy místo Viši.

Pokud uživatel odešle k vyhledávání výraz, pro který se v databázi nevyhledá žádná odpověď, ani není nalezen ekvivalentní výraz ve slovníku, tak je možné, že uživatel napsal překlep. Pro tento případ je fulltextový vyhledávač doplněn funkcí `levenshtein()` (neboli funkcí pro výpočet Levenshteinovy vzdálenosti), která nabídne uživateli podobný výraz, který možná uživatel zamýšlel napsat. Tato funkce je podrobněji popsána v kapitole 6.4.3. Pokud uživatel s nabídnutým výrazem nesouhlasí, je vyhledávání ukončeno. V případě souhlasu probíhá opět vyhledávání přesnou shodou a volnou shodou.

Výsledky vyhledávání se zobrazují dle určených vah pro vyhledávání. Nejvyšší váhy jsou

přiděleny značce produktu, následně názvu produktu, dále názvu řady, popisku produktu a popisu produktu.

Výstupem je seřazení výsledků sestupně, podle nejvyššího obdrženého skóre. Nejvýše jsou zobrazeny výsledky z prohledávání ve značce produktu, pokud již žádné další výsledky nejsou nalezeny, tak se zobrazí výsledky na základě shody s názvem produktu, dále následují výsledky obsahující výraz v názvu řady, v popisku a popisu produktu.

Nejprve jsou zobrazeny výsledky nalezené přesnou shodou, které mají přiděleny vyšší váhy, následují výsledky nalezené volnou shodou. Podrobnější popis vyhledávání přesnou shodou a volnou shodou je uveden v následující kapitole 6.4.2

Zobrazením relevantních výsledků je fulltextové vyhledávání ukončeno. Dalším možným scénářem je, že po zadání výrazu uživatelem nebude tento výraz nalezen ani přesnou shodou, ani volnou shodou, nebude ani obsažen ve slovníku kosmetika_slovnik_slovo a z případných nabídnutých podobných výrazů si uživatel nevybere, případně by nebyly žádné podobné nalezeny. V tomto případě hledané zboží pravděpodobně není k dispozici a uživateli je o tomto výsledku informován, čímž činnost algoritmu alternativně končí.

6.4.2 Možnosti prezentace výsledků a zajištění relevance

Data, která jsou vyhledána jako výsledky vyhledávání, musí být také správně zobrazena a musí být seřazena dle relevance – tj. co možná nejpřesnější shody se zadaným výrazem.

Zajištění relevance

Výsledky, které obdrží nejvyšší číslo relevance, budou zobrazeny nejvýše. Vyhledávací dotaz podle přirozeného jazyka určuje relevanci hledaného výrazu vzhledem k zadanému dotazu (ohodnocení desetinným číslem). Relevance se vypočítává na základě počtu slov v řádku, počtu unikátních slov v daném řádku a počtu dokumentů (řádků), které obsahují určité slovo. Dále se z vyhledávání vyřazují slova, která se vyskytují ve více než 50 % záznamů. Relevance je založena na počtu vyhovujících slov a četnosti jejich výskytu. Slova, která se vyskytují v mnoha záznamech, mají nižší váhu než slova, která se vyskytují vzácně. U booleovského vyhledávání se ohodnocují výsledky vyhledávání výhradně na základě toho, zda vyhledávaný termín vyhovuje zcela (hodnota 1), nebo vůbec (hodnota 0).

Dle návrhu je žádoucí, aby se na prvních místech zobrazovaly shody ve značce produktu, následně názvu produktu, dále názvu řady, popisku produktu a popisu produktu. V tomto případě je nutné napsat složitější dotazy, které přiřadí váhy jednotlivým produktům dle požadavku na jejich zobrazení. Vyhledávání přesnou shodou je realizováno přirozeným fulltextovým vyhledáváním, které standardně ohodnocuje dle relevance, což je nezáporné číslo. Řazení pak může být založeno

na těchto ohodnoceních (pro řazení sestupně dle relevance lze použít sestupné řazení podle ohodnocení). V případě neshody se vyhodnotí na 0. Pokud je žádoucí tuto relevanci změnit, je možné přidat číselné váhy, které se budou s číslem relevance násobit a tak lze relevanci zvýšit. Tímto způsobem lze určit, které výsledky se budou zobrazovat na prvních místech.

Jak již bylo uvedeno, vyhledávání probíhá ve dvou úrovních. V obou případech vyhledávání jsou zobrazeny výsledky pouze s produkty, které jsou online – tj. jsou dostupné. Nejprve je provedeno vyhledávání na přesnou shodu se zadaným výrazem, následuje vyhledávání volnou shodou se zadaným výrazem. V obou případech vyhledávání výrazu předchází úprava vyhledávaného výrazu odstraněním pomlček, ampersandu, tečky, středníku a lomítka. Tuto úpravu zpracovává PHP skript pomocí funkce `preg_replace()`.

SQL dotaz uvedený ve skriptu fulltext.php začíná klauzulí `SELECT`, kde je uvedeno, jaké sloupce a z jakých tabulek si přejeme zobrazit jako výsledek vyhledávání.

```
SELECT kosmetika_produkt.id_kosmetika_produkt, kosmetika_produkt.kosmetika_kod,
kosmetika_zarazeni.kategorieID, kosmetika_firmy.nazev_firmy,
kosmetika_produkt.kosmetika_nazev, kosmetika_rada.rada_nazev,
kosmetika_odstiny.odstin_nazev,kosmetika_produkt.kosmetika_popisek,
kosmetika_produkt.kosmetika_popis
```

Pomocí operátoru `MATCH` definujeme sloupce, které se mají prohledávat a hledaný řetězec zadáme pomocí `AGAINST()`. Výraz `MATCH` se vyskytuje v dotazu celkem dvakrát - jednou v klauzuli `SELECT` a jednou v klauzuli `WHERE`. V případě klauzule `SELECT` slouží k výpočtu celkového skóre, zatímco v případě klauzule `WHERE` slouží k omezení výsledku pouze na případy, kdy není dané skóre nulové (tj. žádné z hledaných slov se v tom případě v daném sloupci nevyskytuje).

Následující část skriptu ukazuje také rozložení vah. Nejvýše se budou zobrazovat výsledky, které se shodují v názvu firmy, proto je zde nastavena váha pro násobení 6, dále výsledky z názvu kosmetiky, jejichž ohodnocení relevance se násobí číslem 5 atd. Pokud je výraz nalezen vícekrát, např. v názvu produktu a zároveň i v popisku produktu, sečtou se jednotlivá skóre a výsledné skóre je vyšší, než když je výraz nalezen pouze v názvu produktu.

```
MATCH(kosmetika_firmy.nazev_firmy)AGAINST("'.$search.'") * 6 +
MATCH(kosmetika_produkt.kosmetika_nazev)AGAINST("'.$search.'") * 5 +
MATCH(kosmetika_rada.rada_nazev)AGAINST("'.$search.'") * 4 +
MATCH(kosmetika_odstiny.odstin_nazev)AGAINST("'.$search.'") * 3 +
MATCH(kosmetika_produkt.kosmetika_popisek)AGAINST("'.$search.'") * 2 +
MATCH(kosmetika_produkt.kosmetika_popis)AGAINST("'.$search.'") * 1)
as score FROM kosmetika_produkt
```

Dalším krokem je spojení tabulek, ze kterých probíhá vyhledávání pomocí jejich identifikatorů.

```
LEFT JOIN kosmetika_odstiny ON
kosmetika_produkt.id_kosmetika_produkt = kosmetika_odstiny.id_produkt_c
LEFT JOIN kosmetika_firmy ON
kosmetika_firmy.id_kosmetika_firmy = kosmetika_produkt.id_firmy_c
LEFT JOIN kosmetika_rada ON kosmetika_rada.rada_id = kosmetika_produkt.rada_id
LEFT JOIN kosmetika_zarazeni ON
kosmetika_zarazeni.produktID=kosmetika_produkt.id_kosmetika_produkt
WHERE  kosmetika_produkt.online = 1
```

Dále již probíhá realizace vyhledávání přesnou shodou. K jeho realizaci je nutné hledaný výraz uzavřít do uvozovek. V klauzuli `WHERE` je dále zajištěno, že dotaz vrátí pouze ty výsledky, u nichž alespoň pro jeden sloupec platí, že jeho relevance zadanému výrazu je nenulová.

Klauzule `ORDER BY` zajišťuje sestupné řazení podle skóre. Klauzule `LIMIT` poté omezuje celkový počet možných výsledků tohoto dotazu na hodnotu 50.

```
WHERE  kosmetika_produkt.online = 1  AND
  MATCH(kosmetika_firmy.nazev_firmy)AGAINST("'.$search.'") OR
  MATCH(kosmetika_produkt.kosmetika_nazev)AGAINST("'.$search.'") OR
  MATCH(kosmetika_rada.rada_nazev)AGAINST("'.$search.'") OR
  MATCH(kosmetika_odstiny.odstin_nazev)AGAINST("'.$search.'") OR
  MATCH(kosmetika_produkt.kosmetika_popisek)AGAINST("'.$search.'") OR
  MATCH(kosmetika_produkt.kosmetika_popis)AGAINST("'.$search.'") AND
 kosmetika_produkt.online = 1
ORDER BY score DESC LIMIT 50
```

Po vyhledání na přesnou shodu následuje vyhledávání podle volné shody. Vyhledávání volnou shodou je založeno na použití modifikátorů (znaky se specifickým významem, které jsou součástí hledaného výrazu). Význam znaku plus je takový, že výskyt hledaného slova z výrazu, před kterým se tento znak vyskytuje je povinný. Význam znaku hvězdička je takový, že místo něj může existovat libovolný řetězec (i nulové délky).

```
MATCH(kosmetika_firmy.nazev_firmy)
AGAINST("+'.$search.'*" IN BOOLEAN MODE)*6 +
MATCH(kosmetika_produkt.kosmetika_nazev)
AGAINST("+'.$search.'*" IN BOOLEAN MODE)*5 +
MATCH(kosmetika_rada.rada_nazev)
AGAINST("+'.$search.'*" IN BOOLEAN MODE)*4 +
MATCH(kosmetika_odstiny.odstin_nazev)
AGAINST("+'.$search.'*" IN BOOLEAN MODE)*3 +
MATCH(kosmetika_produkt.kosmetika_popisek)
AGAINST("+'.$search.'*" IN BOOLEAN MODE)*2 +
MATCH(kosmetika_produkt.kosmetika_popis)
AGAINST("+'.$search.'*" IN BOOLEAN MODE) *1) as score
```

Dotaz pro vyhledávání přesnou shodou a dotaz pro vyhledávání volnou shodou jsou sjednoceny pomocí klauzule `UNION`. Protože zadaný výraz může být nalezen přesnou shodou a zároveň volnou shodou, může se stát, že se uživateli zobrazí duplicity. Přestože `UNION` duplicity standardně odstraňuje, v tomto případě se tomu tak nestane. Ačkoliv se výrazy shodují, obdrží ale

každý výraz jiné skóre pro přesnou shodu a jiné skóre pro volnou shodu a tak je výsledek vyhodnocen jako dva různé výsledky. Z tohoto důvodu obsahuje druhý SQL dotaz vnořený dotaz, ve kterém se zjišťuje, zda již identifikátor produktu není vyhledán přesnou shodou. V případě, kdy je produkt vyhledán přesnou shodou, zobrazí se na vyšší pozici, než v případě, kdy by byl vyhledán volnou shodou a zobrazí se pouze jedenkrát.

```
ORDER BY score DESC LIMIT '.$stranka.', '.$strankovatPo)
```

Klauzule `ORDER BY` zajišťuje sestupné řazení podle skóre poté, co jsou výsledky obou dotazů sjednoceny. Klauzule LIMIT poté omezuje celkový počet možných výsledků tohoto dotazu podle $stranka a $strankovatPo.

Sestavení tohoto dotazu předcházelo dlouhé testování různých SQL dotazů. Nejprve byl sestavený dotaz testován ve Workbech, kde byla jednotlivá skóre vypisována do sloupců, jak je ukázáno na Obrázek 5. Ve výpisu skóre bylo kontrolováno, zda se výsledky řadí správně dle relevance, tj. zda obdržely skóre takové, jaké je potřeba pro správné zobrazení. V případě, kdy se výsledky zobrazovaly dle požadavků, byl SQL dotaz vložen do skriptu fulltext.php a dále byl dotaz testován v testovací aplikaci, kde byla možnost rychle zadávat výrazy pro vyhledávání a tak rychleji kontrolovat, zda je dotaz sestaven správně. V případě testování v SQL editoru by bylo nutné vždy pro každý jednotlivý hledaný výraz upravit celý SQL dotaz, což by bylo časově náročné.

nazev_firmy	kosmetika_nazev	rada_nazev	odstin_nazev	score_1	score_2	score_3	score_4	score_5	score_6
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Tulle	0	5	0	0	2	0
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Vintage	0	5	0	0	2	0
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Lucky	0	5	0	0	2	0
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Jewel	0	5	0	0	2	0
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Bubbles	0	5	0	0	2	0
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Limelight	0	5	0	0	2	0
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Sparkle	0	5	0	0	2	0
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Topaz	0	5	0	0	2	0
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Moonbeam	0	5	0	0	2	0
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Vanilla	0	5	0	0	2	0
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Gold	0	5	0	0	2	0
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Party	0	5	0	0	2	0
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Gala	0	5	0	0	2	0
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Aura	0	5	0	0	2	0
Elizabeth Arden	Oční stíny (Color Ingrigue Eyeshad...	Dekorativní kosmetika	Siren	0	5	0	0	2	0

Obrázek 5: Skórování výsledků ve Workbench, zdroj: [vlastní]

Stránkování

Zobrazovaných výsledků vyhledávání může být mnoho a uživatel by měl mít možnost prohlédnout si všechny vyhledané reference. Toto je hlavní důvodu k řešení stránkování. Ke stránkování je nutné znát počet řádků, které nám databáze při volání dotazu vrátí a také počet řádků, které lze vypsat na každou stranu.

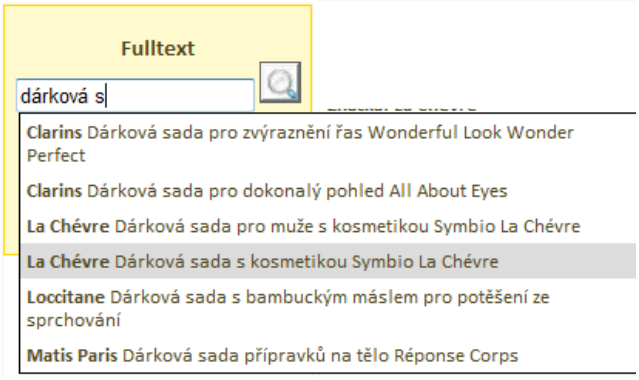
Na konci dotazu je v klauzuli `LIMIT` uveden počet požadovaných řádků s výsledky, které se mají zobrazit na jednu stranu.

6.4.3 Doplnění fulltextu

Pro uživatele je důležité, aby vyhledaly výsledky, které požadují. V některých případech se může stát, že si uživatel již nepamatuje přesný výraz – například celý název produktu nebo přesný název značky tak, jak se správně píše. Pro tyto účely je fulltextové vyhledávání doplněno našeptávačem a nabídkou podobného výrazu.

Našeptávač

Při psaní výrazu do textového pole se po stisku klávesy nabízí aktualizované názvy produktů odpovídající zapisovanému výrazu, jak ukazuje Obrázek 6. V případě, že není nalezen v databázi žádný relevantní výsledek pro „našeptávání“, našeptavač nic nezobrazí. Našeptávač je explicitně zapnutý. Pokud si uživatel nepřeje našeptávač využívat, může jej vypnout.



Obrázek 6: Našeptávač, zdroj: [vlastní]

Realizace našeptávání probíhá pomocí JavaScriptu s využitím JavaScriptové knihovny jQuery. Dokumentace API je k dispozici na http://jquery.com/. Pro správnou funkčnost je nutné mít v prohlížeči povolen JavaScript. Dalším požadavkem pro použití jQuery je uložit soubor jquery-1.5.1.js do složky s JavaScript kódem. Následující řádky vysvětlují zdrojový kód našeptávače, který je částí JavaScriptového kódu akce.js.

Při každém uvolnění stisknuté klávesy ve vyhledávacím políčku je vykonána funkce.

```
$("#naseptavac").bind("click keyup", function(){
```

Tato funkce kontroluje, zda je našeptávač aktivován. Pokud ano, tak proměnná genTimestamp vygeneruje náhodné číslo. Důvodem tohoto kroku je vyřešení případu, kdy se nám

vrací odpovědi od serveru v přeházeném pořadí, než ve kterém byly na server odeslány, protože na serveru vzniká různé zpoždění (aktuální vytížení serveru, databáze apod.)

```
if ($("#naseptavac").val() != "" && $("#nastaveniFulletext").val() == "1"){
     var genTimestamp = Math.random()+1;
      $.post("naseptavac.php", {datainput: $("#naseptavac").val(),   timestamp:
          genTimestamp}, function(data)
```

Následně jsou data odeslána prostřednictvím `datainput` php skriptu naseptavac.php. Zde jsou ze vstupních dat odstraněny nežádoucí znaky.

```
return preg_replace("/[\&\-\;\.\']/", " ", $vstup);
```

Do textové proměnné `$returnHtml.` se následně bude vypisovat celý text, který je požadováno vypsat. Funkce `strip_tags()` odstraní HTML a PHP značky.

```
$returnHtml = "";
$search  = nahradit(strip_tags($_POST["datainput"]));
```

Výpis dat se realizuje pomocí SQL dotazu, který provádí výběr názvu značky a názvu produktu podle shody s $search. Výsledky tohoto vyhledávání budou později zobrazeny v boxu jako „výsledky našeptávání“.

Dále jsou nalezeny všechny neviditelné znaky a jsou nahrazeny mezerou. Následující příkaz `echo` kóduje textově výstup do JSON formátu.

```
$returnHtml = preg_replace("/\s+/", " ", $returnHtml);
    echo "{\"html\": \"".$returnHtml."\", \"timestamp\":
    \"".$_POST["timestamp"]."\"}";
```

Takto zakódovaný text je dále zpracováván skriptem akce.js. V případě, kdy se kontrolní náhodné číslo z dat vrácených z php skriptu rovná vygenerovanému náhodnému číslu v JavaScript kódu, vypíše se přes funkci `html()` textové html, které se přeformátuje, zpracuje a jako HTML kód se vloží do boxu, ve kterém se uživateli vypíší „výsledky našeptávání“. V opačném případě (neshoda čísel) se výsledky našeptávání“ skryjí. Pokud data.html neobsahují žádnou hodnotu jsou výsledky také skryty.

```
$.post("naseptavac.php", {datainput: $("#naseptavac").val(), timestamp:
genTimestamp}, function(data){
         if (data.timestamp == genTimestamp){
            if(data.html != ""){
              $("#vysledkyNaseptavani").html(data.html).show();
            } else {
                    $("#vysledkyNaseptavani").hide();
            }
         }
      }, "json");
    } else {
       $("#vysledkyNaseptavani").hide();
    }
```

```
});
$("body").click(function(){
    $("#vysledkyNaseptavani").hide();
});
```

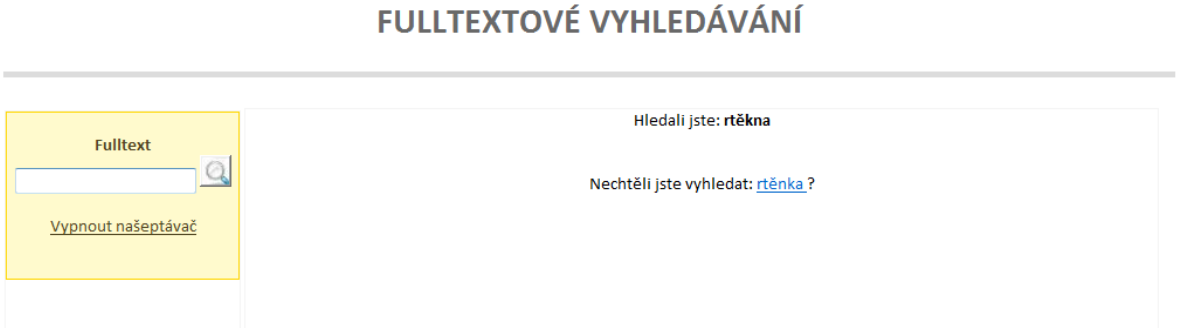
Tento našeptávač nezobrazuje výsledky dle počtu vyhledávání, tj. dle nejvyhledávanějších výrazů. K jeho realizaci by bylo možné využít tabulky kosmetika_fulltext, do které jsou ukládány jednotlivé výrazy s počtem jejich vyhledávání, další alternativou přidat pole do tabulky kosmetika_produkt a do tohoto pole ukládat počet kolikrát byl daný produkt vyhledán.

Levenshtein

Pomocí funkce `levenshtein()` se vypočítá Levenshteinova vzdálenost definovaná jako minimální počet znaků, které je možné nahradit, vložit nebo odstranit, aby se výraz změnil na požadovaný.

V případě zadání slova obsahující překlep do vyhledávače je toto slovo porovnáváno se slovy uloženými v databázi a dle výpočtu Levenshteinovy vzdálenosti je nabídnuto uživateli slovo ve kterém je Levenshteinova vzdálenost nejmenší, tj. liší se v nejmenším počtu znaků, jak je vidět na Obrázek 7

Oprava by měla být nabídnuta pouze v případě, že nebyly nalezeny žádné výsledky vyhledávání a v databázi existuje podobné slovo, pro které se mohou výsledky zobrazit. Pokud se uživatel rozhodne pro vyhledávání nabídnutého výrazu, probíhá dále fulltextové vyhledávání výrazu standardní cestou.



Obrázek 7: Levenshtein, zdroj: [vlastní]

Realizaci hledání podobného výrazu představuje fulltext.php. Proměnná `shortest` ukazuje na nejbližší přesnou shodu. Nejvyšší shoda je 0, proto je výchozí hodnota nastavena na -1.

```
function najdiPreklep($kde){
    if ($kde != ""){
       $shortest = -1;
```

Nejprve je vypočítán počet mezer. Dále je vytvořeno prázdné pole, do kterého se budou předávat porovnávané výrazy. V dalším příkazu se pomocí funkce `explode` rozdělí řetězec

s výrazy z databáze dle mezer na jednotlivá slova a je určen jejich index pomocí proměnné `pozice`. `Ceil` zaokrouhluje desetinné číslo nahoru. Do proměnné `pocetPole` je uložen počet slov, které bude osahovat jedno pole.

```
$mezerZadano = substr_count($kde, " ");
$pole = array();
$exp = explode(" ", $data["kosmetika_nazev"]);
$pozice = 0;
$pocetPole = @ceil(count($exp) / $mezerZadano);
```

Dále následuje cyklus, který zajistí naplnění pole (dle počtu slov) výrazy z databáze. Číslo 0 určuje jedno slovo, neboť neobsahuje mezeru.

Další cyklus prování porovnání výrazu zadaného uživatelem a výrazu z databáze, který je uložen v poli. Následuje funkce `levenshtein()`, pomocí které je vypočítána vzdálenost neboli počet znaků, které se neshodují s výrazem zadaným uživatelem. Nejbližší výraz shodující se s výrazem, který zadal uživatel je ukládán do proměnné `closest`. Proměnná `shortest` obsahuje počet písmen, která se neshodují. Pokud se např. liší ve 3 znacích, tak je její hodnota 3.

```
foreach ($pole as $hodnota){
        $lev = levenshtein($kde, $hodnota);
        if ($lev == 0) {
          $closest = $hodnota;
          $shortest = 0;
          break;
        }
```

Když je Levenshteinova vzdálenost menší než počet písmen, které se neshodují (proměnná `shortest`), je tato hodnota uložena do proměnné `closest` a ukazuje výraz, který má nejvyšší shodu s výrazem, který zadal uživatel. Zároveň dojde ke změně proměnné `shortest`, která bude obsahovat novou, nejmenší hodnotu Levenshteinovy vzdálenosti.

```
if ($lev <= $shortest || $shortest < 0) {
    $closest  = $hodnota;
    $shortest = $lev;
```

V závěru je výraz z proměnné `closest` vypsán uživateli jako nabídka výrazu pro vyhledávání.

6.5 Externí fulltextové vyhledávání v MySQL

Dalším způsobem fulltextové vyhledávání v MySQL je použití externího fulltextového vyhledávače, jakým může být např. Sphinx (http://www.sphinxsearch.com) nebo Mnogosearch (http://www.mnogosearch.org/products.html).

Sphinx je open source fulltextový vyhledávač. Je možné jej využít jako užitečný doplněk k MySQL. Sphinx nabízí API pro několik programovacích jazyků včetně PHP. Sphinx je prospěšný

především pro prohledávání velmi rozsáhlé databáze. Je velmi rychlý a podporuje distribuované vyhledávání, je možné jej využít pro rychlá seskupování a setřiďování. Verze 0.9.9. podporuje také češtinu. [18]

Sphinx má dva hlavní programy. Jedním z nich je indexer, který získává dokumenty ze specifikovaných zdrojů (např. výsledků dotazů MySQL) a vytváří nad nimi fulltextový index. Druhou částí je searchd – démon, který obsluhuje vyhledávací dotazy z indexů, který vybudoval indexer. [18]

Verze SphinxSE má vlastní úložný engine, který umožňuje přistupovat k Sphinx přímo prostřednictvím MySQL. Umí indexovat kromě MyISAM i tabulky InnoDB, používá algoritmus pro vracení relevantnějších výsledků (nejprve přesná shoda, pak volná), dále filtrování a přiděluje pořadí dle blízkosti slov. K dalším funkcionalitám Sphinxu patří podpora pro indexování HTML, podpora tvarosloví a synonym. [18]

ZÁVĚR

Cílem této práce bylo charakterizovat a porovnat vyhledávácí metody používané na internetu. Dalším cílem bylo charakterizovat současný stav fulltextového vyhledávání webových aplikací realizovaných nad databázovým serverem MySQL, představit varianty fulltextového vyhledávání v MySQL a provést jejich porovnání.

V úvodní části práce je uvedena charakteristika vyhledávacích metod používaných na internetu, včetně jejich porovnání dle zvolených kriterií. Následně je charakterizován současný stav fulltextového vyhledávání v MySQL. Tato práce popisuje jednotlivé varianty, které jsou k dispozici pro zajištění fulltextového vyhledávání s použitím MySQL, vzájemně je porovnává a poukazuje na některé případy, kdy lze jednotlivé varianty aplikovat. Zároveň vymezuje prostor pro aplikovatelnost těchto variant vhodnou volbou ukládacího enginu MySQL a upozorňuje na možná rizika a potenciálně problematické vlastnosti vybraného ukládacího enginu. Zároveň navrhuje některé postupy, které jsou obecně aplikovatelné za účelem snížení nebo eliminace takovýchto vlastností, které se jeví jako problematické.

Dalším cílem práce bylo vytvořit návrh fulltextového vyhledávání internetového obchodu a tento návrh otestovat. Pro rychlé zobrazení relevantních výsledků vyhledávání je nutné vytvořit správný návrh databáze, nebo v případě již existujícího databázového modelu provést jeho kontrolu a případné úpravy a dále provést optimalizaci pro fulltextové vyhledávání včetně zajištění relevance zobrazených výsledků. Dalším cílem bylo provést návrh na doplnění fulltextového vyhledávače.

Tato práce popisuje otestování vlastního návrhu fulltextového vyhledávání internetového obchodu Krasa.cz. Databáze tohoto internetového obchodu byla importována do MySQL se stávající databázovou strukturou. Pro správu serveru MySQL byla používána Workbench 5.2.31.

Pro stanovení, zda je tato existující databázová struktura vhodná pro splnění cíle zadání, bylo zpětně provedeno její ověření prostřednictvím kontroly některých obecných formálních požadavků, které jsou při vytváření databázové struktury běžně používány. Následně byla tato struktura podrobena kontrole z hlediska integritního omezení, normalizace dat a referenční integrity. Na základě výsledků této kontroly bylo stanoveno, že není potřeba vytvářet kompletně novou databázovou strukturu, avšak stávající stačí změnit do podoby, kdy bude pro dosažení stanoveného cíle spíše použitelnější a to především z důvodu tendence minimalizovat čas dotazů (optimalizace). To vedlo především k požadavkům na přidání specifických typů indexů pro vybrané atributy některých entit. Efektivita některých těchto změn byla prokázána v rámci testování neupravené a upravené verze databáze, přičemž volbou vhodných opatření bylo také zabezpečeno, aby výsledky těchto experimentů byly dostatečně statisticky průkazné.

Pro účely vyhledávání nestandardních výrazů a evidenci vyhledávaných výrazů bylo navrženo do databáze přidat další tabulky. V práci je uveden popis entit a vztahů entit databázové struktury, ve kterém jsou popsány i nově přidané tabulky. Práce obsahuje také grafické znázornění pomocí ER diagramu.

Následně byla provedena optimalizace pro fulltextové vyhledávání, která byla realizována definováním indexů, optimalizací SQL dotazu a serverovým nastavením.

Práce dále popisuje algoritmus návrhu fulltextového vyhledávání a další možnosti doplnění fulltextového vyhledávání, jakými mohou být našeptávač nebo nabídka podobného výrazu. Pro účely testování fulltextového vyhledávání dle vlastního návrhu byla vytvořena aplikace zobrazující fulltextové políčko a výpis produktů. Zdrojové kódy této aplikace jsou součástí příloh této práce.

Pro realizaci fulltextového vyhledávání je důležitý správně sestavený SQL dotaz, který zajistí vyhledání relevantních výsledků. Tvorba tohoto SQL dotazu není jednoduchá a lze narazit na řadu problémů. Jedním z nich může být zajištění relevance, které je explicitně podporováno pouze v jedné variantě fulltextového vyhledávání v MySQL – v přirozeném fulltextovém vyhledávání. Po dlouhém testování několika různých SQL dotazů ve Workbench a sledování relevance vyhledávání dle požadavků byly v dotazu přidány číselné váhy, které zajistí zvýšení relevance výsledků dle požadavku – tj. zobrazí výsledky nejprve dle shody ve značce produktu, následně dle shody v názvu produktu, dle shody v popisu produktu apod.

Další problém může nastat při vyhledávání složeného výrazu. Fulltextové vyhledávání v MySQL standardně vyhledává nejprve přesnou shodou celé slovní spojení, kterému je přiřazena nevyšší váha, ale také záznamy, ve kterých je nalezeno jakékoliv ze slov z vyhledávaného slovního spojení. Tímto se může stát, že se uživateli zobrazí ve výsledcích produkt, který se neshoduje s hledaným výrazem. Tento problém je možné řešit pomocí Booleovského vyhledávání, které nabízí používání modifikátorů, s jejichž pomocí lze nadefinovat požadavek na výskyt veškerých slov z hledaného výrazu.

Pomocí vytvořené aplikace bylo provedeno otestování našeptávače a zobrazení nabídky podobného výrazu, které minimalizuje chyby uživatele při zadání výrazu pro vyhledání.

Ačkoliv MySQL nabízí přímo podporu fulltextového vyhledávání definováním fulltextových indexů, pro realizaci kvalitního fulltextového vyhledávače je nutné použít složitě poskládané SQL dotazy a v některých případech i s využitím regulárních výrazů. Další možností je vyzkoušet externí fulltextový vyhledávač, kterým může být například Sphinx.

Domnívám se, že cíle práce uvedené v úvodu práce byly splněny a práce přiblížila problematiku fulltextového vyhledávání.

SEZNAM POUŽITÉ LITERATURY

[1] BÍLA, Jiří, KRÁL, František, HLAVÁČ, Vladimír. *Informační technologie: databázové a znalostní systémy.* 2. vyd., přeprac. Praha: Vydavatelství ČVUT, 2003. 126 s. ISBN 80-01-02790-2

[2] BOLDIŠ, Petr. *Základy vyhledávání na internetu.* [online]. 2004 [cit. 2011-03-09]. Praha: Studijní a informační centrum České zemědělské univerzity, 2004. Dostupné z WWW: <http://dl.webcore.czu.cz/file/ZGFRY0VseU83OU09>

[3] CONOLLY, Thomas, BEGG, Carolyn E, HOLOWCZAK, Richard. *Mistrovství – databáze: profesionální průvodce tvorbou efektivních databází*. 1. vyd. Brno: Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7

[4] Český statistický úřad. *Internet a komunikace* [online]. 2010 [cit. 2010-10-31]. Dostupné z WWW: <http://www.czso.cz/csu/redakce.nsf/i/internet_a_komunikace>.

[5] Český statistický úřad. *Internet a jeho využití* [online]. 2010 [cit. 2010-10-09]. Dostupné z WWW: <http://www.czso.cz/csu/redakce.nsf/i/7_elektronicky_nakup>.

[6] DUBOIS, Paul. *MySQL profesionálně: Komplexní průvodce použitím, programováním a správou MySQL.* 2. vyd. Praha: Mobil Media a.s., 2003. 1071 s. ISBN 80-86593-41-X.

[7] HLAVENKA, Jiří. *Mistrovství ve vyhledávání na Internetu.* 1. vyd. Praha: Computer Press, 2002. 195 s. ISBN 80-7226-759-0.

[8] HUB, Miloslav. *Technologie internetu – PHP5 : distanční opora*. 1. vyd. Pardubice: Univerzita Pardubice, 2009. 88 s. ISBN 978-80-7395-163-4

[9] KOFLER, Michael. *Mistrovství s MySQL: Kompletní průvodce webového vývojáře.* 1. vyd. Brno: Computer Press, a.s., 2007. 805 s. ISBN 978-80-251-1502-2.

[10] LACKO, Luboslav. *PHP5 a MySQL5*. 1. vyd. Brno: Cumputer Press, a.s., 2007. 320 s. ISBN 978-80-251-1695-1.

[11] NÁDBĚLA, Josef. *Velký počítačový slovník*: *výklad pojmů, výrazů a zkratek z počítačové terminologie.* 1. vyd. Kralice na Hané: Computer Media, 2004. 455 s. ISBN 80-866836-21-3.

[12] Netcraft Ltd. *Web Server Survey* | Netcraft [online]. 2010 [cit. 2011-02-22]. September 2010 Web Server Survey. Dostupné z WWW: <http://news.netcraft.com/archives/category/web-server-survey/>.

[13] Netscape Communications Corporation. *Open Directory Project: Who We Are And What We Do* [online]. 2008, 6.2.2008 [cit. 2011-02-05]. Dostupné z WWW: <http://www.dmoz.org/help/geninfo.html>.

[14] OPPEL, Andreaw J, KRÁSENSKÝ, David. *Databáze bez předchozích znalostí: [průvodce pro samouky]*. 1. vyd. Brno : Computer Press, 2006. 319 s. ISBN 80-251-1199-7

[15] Oracle Corporation and/or its affiliates. *MySQL* [online]. 2010 [cit. 2010-10-21]. 11.9. Full-Text Search Functions. Dostupné z WWW: <http://dev.mysql.com/doc/refman/5.0/en/fulltext-search.html>.

[16] Oracle Corporation and/or its affiliates. *Overview of MySQL Storage Engine Architecture* [online]. 2011 [cit. 2011-03-15]. Dostupné z WWW: <http://dev.mysql.com/doc/refman/5.5/en/pluggable-storage-overview.html>.

[17] SCHNEIDER, D.,Robert: MySQL: *Oficiální průvodce tvorbou, správou a laděním databází.* 1. vyd. Praha: Grada Publishing, a.s., 2006. 372s. ISBN 80-247-1516-3.

[18] SCHVARTZ Baron, ZAITSEV Peter, TKACHENKO Vadim a kol. *MySQL profesionálně. Optimalizace pro vysoký výkon.* 1. vyd. Brno: ZONER software, a.s., 2009. 712s. ISBN 978-80-7413-035-9.

[19] ŠIMONOVÁ, Stanislava; PANUŠ, Jan. *Databázové systémy I: pro kombinovanou formu studia.* 1. vyd. Pardubice: Univerzita Pardubice, 2007. 106 s. ISBN 978-80-7194-988-6

[20] ŠIMONOVÁ, Stanislava; PANUŠ, Jan; NAIMAN, Karel. *Databázové systémy II – SQL, přístup k datovým zdrojům: pro kombinovanou formu studia.* 1. vyd. Pardubice: Univerzita Pardubice, 2006. 100 s. ISBN 80-7194-845-4

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

SEZNAM POUŽITÝCH ZKRATEK A POJMŮ

API Application Programming Interface označuje rozhraní pro programování aplikací. Jedná se o sbírku procedur, funkcí a tříd knihovny.

DOM Document object model – objektový model dokumentu. DOM je API umožňující přístup či modifikaci obsahu, struktury nebo stylu dokumentu nebo jeho částí.

ER diagram Entity relationship diagram

HTML Hypertext mark up language – značkovací jazyk používaný na tvorbu webových stránek

HTTP Hypertext Transfer Protocol

jQuery JavaScript knihovna

JSON JavaScript Object Notation – JavaScript objekt určený pro přenos dat. Vstupem je libovolná datová struktura a výstupem je vždy řetězec.

MySQL My Structured Query Language - systém pro řízení databází

PHP Hypertext Preprocessor, původně Personal Home Page – skriptovací programovací jazyk

RMD Relační model dat

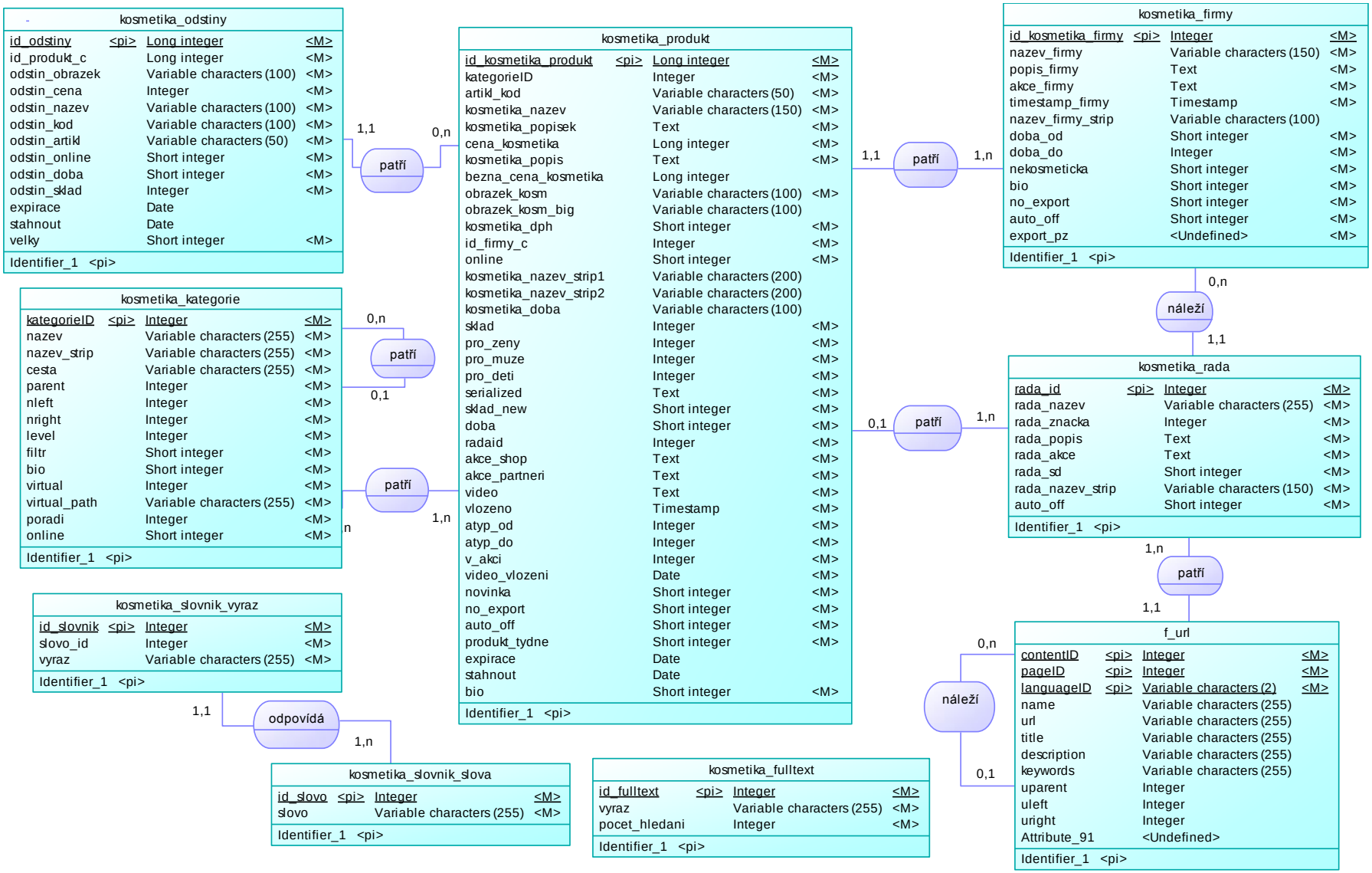
RSŘBD Relační systém řízení báze dat

SQL Structured Query Language - strukturovaný dotazovací jazyk

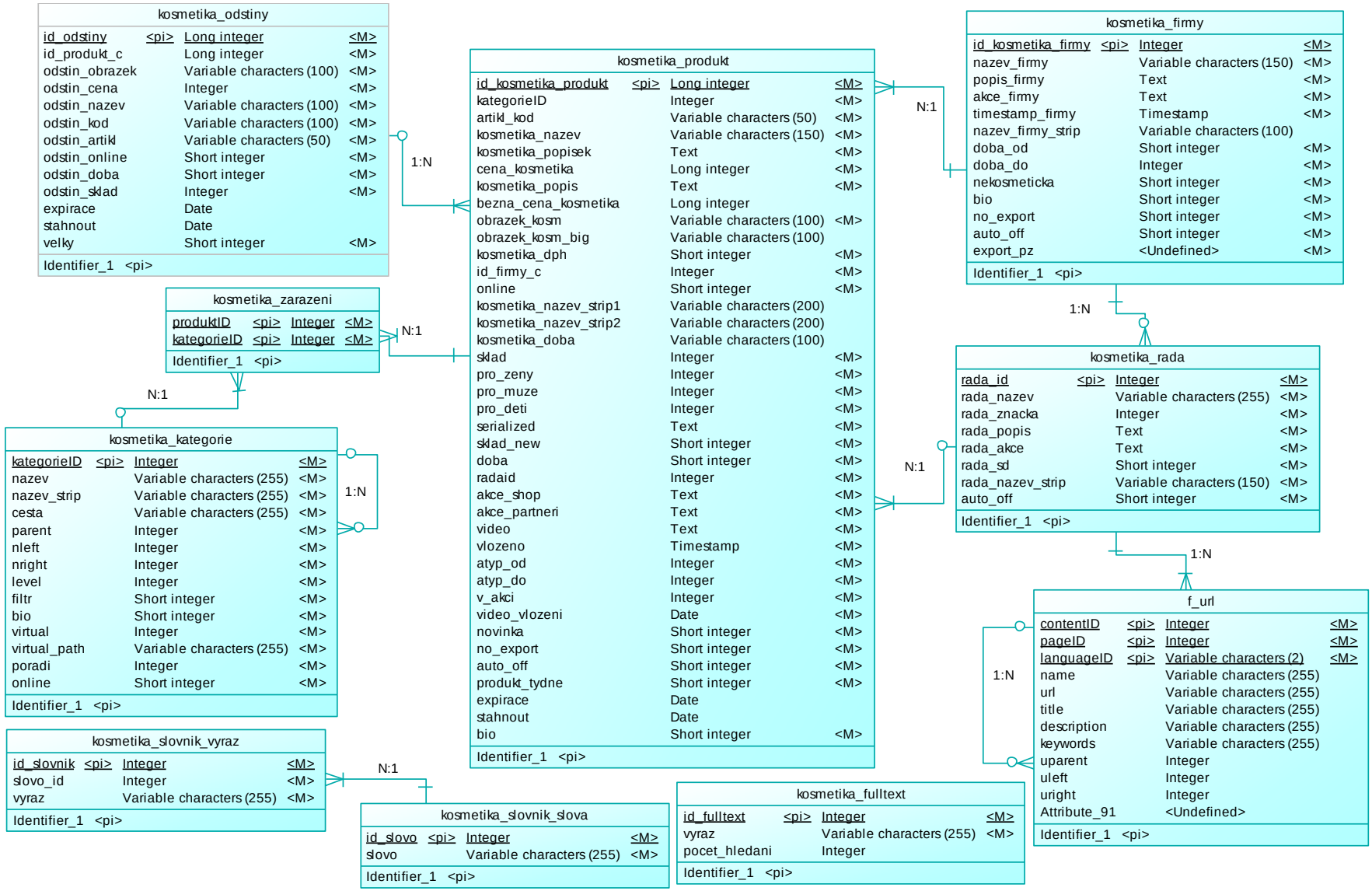
SŘBD Systém řízení báze dat

Entity 1 a její ID	Entita 2 a její ID	Vztah mezi entitami	Textové vyjádření
kosmetika_produkt (id_kosmetika_produkt)	kosmetika_odstiny (id_produkt_c)	0,N - 1,1	Kosmetický produkt nemusí mít přiřazen odstín nebo může mít odstínů několik. Odstín musí být přiřazen právě jednomu produktu. Výsledný vztah relací kosmetika_produkt a kosmetika_odstiny je 1:N.
kosmetika_produkt (id_kosmetika_produkt)	kosmetika_kategorie (kategorieID)	1,N - 1,N	Produkt musí být zařazen do jedné nebo více kategorií. Kategorie musí obsahovat jeden nebo více produktů. Výsledný vztah relací kosmetika_produkt a kosmetika_kategorie má vztah N:M. Vznikla vztahová tabulka kosmetika_kategorie s ID obou relací.
kosmetika_kategorie (kategorieID)	kosmetika_kategorie (parent)	0,N - 0,1	V tabulce kosmetika_kategorie je samoreferenční vztah mezi kategorií a podkategorií. Kategorie nemusí mít podkategorii, ale může jich mít více. Každá podkategorie musí patřit do jedné kategorie. Kategorie na nejvyšší úrovni hierarchie nemá nadřazenou kategorii. Výsledný vztah kategorie a podkategorie je 1:N.
kosmetika_produkt (id_firmy_c)	kosmetika_firmy (id_kosmetika_firmy)	1,1 – 1,N	Produkt musí mít určenou firmu (značku) a to pouze jednu firmu. Firma musí mít přiřazen nějaký produkt a může mít produktů více. Výsledný vztah relací kosmetika_produkt a kosmetika_firmy je 1:N.
kosmetika_produkt (rada_id)	kosmetika_rada (rada_id)	0,1 – 1,N	Produkt může mít určenou řadu, ale maximálně jednu. Řada musí mít přiřazen produkt a může jich mít přiřazeno i více. Výsledný vztah relací kosmetika_produkt a kosmetika_rada je 1:N.
kosmetika_firmy (id_kosmetika_firmy),	kosmetika_rada (rada_znacka)	0,N – 1,1	Firma může mít definovanou řadu a může jich mít definovaných několik. Řada musí náležet pouze jedné firmě. Výsledný vztah relací kosmetika_firmy a kosmetika_rada má vztah 1:1.
kosmetika_slovnik_vyraz (id_slovnik)	kosmetika_slovnik_slova (id_slovo)	1,1 – 1,N	Každému výrazu musí odpovídat jedno slovo ve slovníku. Slovo ve slovníku musí být přiřazeno nějakému výrazu a může být přiřazeno i více výrazům. Výsledný vztah relací kosmetika_slovnik_vyraz a kosmetika_slovnik_slova má vztah 1:N.
kosmetika_rada (rada_id)	f_url (contentID)	1,N – 1,1	Každá řada má přidělenou speciální url adresu a může existovat více url adres pro danou řadu. Url adresa musí být přiřazena jedné řadě. Výsledný vztah relací kosmetika_rada a f_url má vztah 1:N.
f_url (contentID)	f_url (uparent)	0,N-0,1	V tabulce f_url je samoreferenční vztah mezi nadřazenou kategorií a podkategorií. Url nemusí mít podřazenou url, ale může jich mít několik. Podřazená url musí mít jednu nadřazenou url. Url na nejvyšší úrovni hierarchie už nemá nadřazenou url. Výsledný vztah relací je 1:N.

Příloha 1: Integritní omezení vztahů, zdroj: [vlastní]



Příloha 2: ER diagram, zdroj: [vlastní]



Příloha 3: Model databáze, formát IDEF1X, zdroj: [vlastní]

Příloha 4: Zdrojové kódy, zdroj: [vlastní]

db.php

Skript se pokusí vytvořit připojení k databázi a v případě neúspěchu se ukončí s definovanou chybou.

```
<?php
$spojeni = mysql_connect("localhost","root","") or die ('Na serveru se nyni
pracuje, prosime o strpeni. Zkuste to za chvilku. ;)');
mysql_select_db("krasa_utf", $spojeni) or die ('Na serveru se nyni pracuje,
prosime o strpeni. Zkuste to za chvilku....');
mysql_query("SET character_set_client=utf8");
mysql_query("SET character_set_connection=utf8");
mysql_query("SET character_set_results=utf8");
?>
```

---

index. html

Úvodní strana aplikace.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Language" content="cs"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" >
    <meta name="author" content="Ivana"/>
    <title>FULLTEXT</title>
  </head>
  <body>
    <center><br>
      <h1> FULLTEXTOVÉ VYHLEDÁVÁNÍ</h3>
      <h3> Vypracovala: Ivana Broklová</h3>
      <h3> 2011</h3> <br>
      <a href="vyhledavani.php"><img src="image/vpred.jpg" alt="vstupte">
    </center>
  </body>
</html>
```

---

uvod.php

Pokud je navázáno spojení s databází, tento skript zobrazí prvních 30 produktů z databáze.

```
<?php
  require "db.php";  {
  echo "<div id='vysledkyFulltext'>";
  $data = mysql_query("select * from kosmetika_produkt
      left join kosmetika_odstiny on
      kosmetika_produkt.id_kosmetika_produkt=kosmetika_odstiny.id_produkt_c
```

```
      left join kosmetika_firmy on
      kosmetika_firmy.id_kosmetika_firmy=kosmetika_produkt.id_firmy_c
      left join kosmetika_rada on
      kosmetika_rada.rada_id=kosmetika_produkt.rada_id
      left join kosmetika_zarazeni on
      kosmetika_zarazeni.produktID=kosmetika_produkt.id_kosmetika_produkt
    limit 0, 30") or die (mysql_error());
   while ($zaznam = mysql_fetch_array($data) ){
    echo '
        <div class="box">
            <table><tr><td><div class="znacka"><strong>Značka:
            </strong>'.$zaznam["nazev_firmy"].'</div></td>
            <td><div class="odstin"><strong>Kód:
            </strong>'.$zaznam["kosmetika_kod"].'</div></td></tr>
            <tr><td colspan="2"><div class="kategorie"><strong>Kategorie:
            </strong> '.$zaznam["kategorieID"].'</div></td></tr>
            <tr><td colspan="2"><div class="nazev">
            <strong>Název: </strong> '.$zaznam["kosmetika_nazev"].'</div>
            </td></tr>
            <tr><td><div class="rada">
            <strong>Řada: </strong> '.$zaznam["rada_nazev"].'</div></td>
            <td><div class="odstin"><strong>Odstín:
            </strong>'.$zaznam["odstin_nazev"].'</div></td></tr>
            <tr><td colspan="2"><div class="popisek"><strong>Popisek:
            </strong> '.$zaznam["kosmetika_popisek"].'</div></td></tr>
            <tr><td colspan="2">
            <a href="#" class="zobraz">Zobrazit Popis</a></td></tr>
            <tr><td colspan="2"><div class="popis"><strong>Popis:
            </strong> '.$zaznam["kosmetika_popis"].'</div></td></tr>
            </table>
           <hr width="400" size="1" color="#dcdcdc" align="center" noshade>
      </div>
    ';
  }
  echo "</div>";
   }
    ?>
```

---

detail.php

Tento skript zobrazí detail produktu, na který uživatel klikl v boxu našeptávače při „našeptávání“.

```
<?php
  echo "<div id='vysledkyFulltext'>";
  $data = mysql_query("select * from kosmetika_produkt
      left join kosmetika_odstiny on
      kosmetika_produkt.id_kosmetika_produkt=kosmetika_odstiny.id_produkt_c
      left join kosmetika_firmy on
      kosmetika_firmy.id_kosmetika_firmy=kosmetika_produkt.id_firmy_c
      left join kosmetika_rada on
      kosmetika_rada.rada_id=kosmetika_produkt.rada_id
      left join kosmetika_zarazeni on
      kosmetika_zarazeni.produktID=kosmetika_produkt.id_kosmetika_produkt
      WHERE kosmetika_produkt.id_kosmetika_produkt = '".$_GET["id"]."'") or
die (mysql_error());
   $zaznam = mysql_fetch_array($data);
```

```
    echo '
        <div class="box">
            <table><tr><td><div class="znacka">
            <strong>Značka: </strong>'.$zaznam["nazev_firmy"].'</div></td>
            <td><div class="odstin"><strong>Kód:
            </strong>'.$zaznam["kosmetika_kod"].'</div></td></tr>
            <tr><td colspan="2"><div class="kategorie"><strong>Kategorie:
            </strong> '.$zaznam["kategorieID"].'</div></td></tr>
            <tr><td colspan="2"><div class="nazev"><strong>Název:
            </strong> '.$zaznam["kosmetika_nazev"].'</div></td></tr>
            <tr><td><div class="rada"><strong>Řada:
            </strong> '.$zaznam["rada_nazev"].'</div></td>
            <td><div class="odstin"><strong>Odstín:
            </strong>'.$zaznam["odstin_nazev"].'</div></td></tr>
            <tr><td colspan="2"><div class="popisek"><strong>Popisek:
            </strong> '.$zaznam["kosmetika_popisek"].'</div></td></tr>
            <tr><td colspan="2"><a href="#" class="zobraz">Zobrazit
            Popis</a></td></tr>
            <tr><td colspan="2"><div class="popis"><strong>Popis:
            </strong> '.$zaznam["kosmetika_popis"].'</div></td></tr>
           </table>
           <hr width="400" size="1" color="#dcdcdc" align="center" noshade>
      </div>
    ';
  echo "</div>";

    ?>
```

---

fulltext.php

Skript vykonává fulltextové vyhledávání a výpis vyhledaných výsledků uživateli. Zobrazené výsledky jsou zobrazeny po 10 produktech na každé straně a v případě zobrazení více výsledků má uživatel možnost přecházet mezi stránkami.

Pokud nejsou nalezeny žádné výsledky, pomocí funkce `projdiSlovnik()` zkontroluje, zda se zvláštní výraz nenachází ve slovníku. Pokud ano, přiřadí mu odpovídající výraz a probíhá vyhledávání tohoto nového výrazu. Pokud ve slovníku odpovídající výraz není nalezen, pomocí funkcí `najdiPreklep()` a `levenshtein()` nalezne nejvíce odpovídající výraz z databáze a nabídne jej uživateli k vyhledání.

Tento skript dále pomocí funkce `ulozVyhledavani()` ukládá vyhledávané výrazy do databáze, a pokud je tento výraz v databázi nalezen, zvýší jeho číslo vyjadřující počet vyhledávání. V případě, že nalezen není, tak k nově vloženému uloží hodnotu počtu vyhledávání 1.

```
<?php
require "db.php";

function fulltextDotaz($search, $stranka, $strankovatPo){
$dotaz = mysql_query(
```

```
'(SELECT kosmetika_produkt.id_kosmetika_produkt,
kosmetika_produkt.kosmetika_kod, kosmetika_zarazeni.kategorieID,
kosmetika_firmy.nazev_firmy, kosmetika_produkt.kosmetika_nazev,
kosmetika_rada.rada_nazev,
kosmetika_odstiny.odstin_nazev,kosmetika_produkt.kosmetika_popisek,
kosmetika_produkt.kosmetika_popis,

 (MATCH(kosmetika_firmy.nazev_firmy)
  AGAINST("'.$search.'" IN BOOLEAN MODE) * 6 +
  MATCH(kosmetika_produkt.kosmetika_nazev)
  AGAINST("'.$search.'" IN BOOLEAN MODE) * 5 +
  MATCH(kosmetika_rada.rada_nazev)
  AGAINST("'.$search.'" IN BOOLEAN MODE) * 4 +
  MATCH(kosmetika_odstiny.odstin_nazev)
  AGAINST("'.$search.'" IN BOOLEAN MODE) * 3 +
  MATCH(kosmetika_produkt.kosmetika_popisek)
  AGAINST("'.$search.'" IN BOOLEAN MODE) * 2 +
  MATCH(kosmetika_produkt.kosmetika_popis)
  AGAINST("'.$search.'" IN BOOLEAN MODE) * 1) as score

FROM kosmetika_produkt
  LEFT JOIN kosmetika_odstiny ON
  kosmetika_produkt.id_kosmetika_produkt = kosmetika_odstiny.id_produkt_c
  LEFT JOIN kosmetika_firmy ON
  kosmetika_firmy.id_kosmetika_firmy = kosmetika_produkt.id_firmy_c
  LEFT JOIN kosmetika_rada ON
  kosmetika_rada.rada_id = kosmetika_produkt.rada_id
   LEFT JOIN kosmetika_zarazeni ON
   kosmetika_zarazeni.produktID=kosmetika_produkt.id_kosmetika_produkt

WHERE  kosmetika_produkt.online = 1
AND MATCH(kosmetika_firmy.nazev_firmy)AGAINST
    ("'.$search.'" IN BOOLEAN MODE) >= 1 or
    MATCH(kosmetika_produkt.kosmetika_nazev)
    AGAINST("'.$search.'" IN BOOLEAN MODE) >= 1 OR
    MATCH(kosmetika_rada.rada_nazev)
    AGAINST("'.$search.'" IN BOOLEAN MODE) >= 1 OR
    MATCH(kosmetika_odstiny.odstin_nazev)
    AGAINST("'.$search.'" IN BOOLEAN MODE) >= 1 OR
    MATCH(kosmetika_produkt.kosmetika_popisek)
    AGAINST("'.$search.'" IN BOOLEAN MODE) >= 1 OR
    MATCH(kosmetika_produkt.kosmetika_popis)
    AGAINST("'.$search.'" IN BOOLEAN MODE) >= 1 AND
kosmetika_produkt.online = 1
ORDER BY score DESC LIMIT 50

  )UNION(
SELECT kosmetika_produkt.id_kosmetika_produkt,
kosmetika_produkt.kosmetika_kod, kosmetika_zarazeni.kategorieID,
kosmetika_firmy.nazev_firmy, kosmetika_produkt.kosmetika_nazev,
kosmetika_rada.rada_nazev,
kosmetika_odstiny.odstin_nazev,kosmetika_produkt.kosmetika_popisek,
kosmetika_produkt.kosmetika_popis,
      MATCH(kosmetika_firmy.nazev_firmy)
      AGAINST("+'.$search.'*" IN BOOLEAN MODE)*6 +
      MATCH(kosmetika_produkt.kosmetika_nazev)
      AGAINST("+'.$search.'*" IN BOOLEAN MODE)*5 +
      MATCH(kosmetika_rada.rada_nazev)
      AGAINST("+'.$search.'*" IN BOOLEAN MODE)*4 +
      MATCH(kosmetika_odstiny.odstin_nazev)
      AGAINST("+'.$search.'*" IN BOOLEAN MODE)*3 +
```

```
      MATCH(kosmetika_produkt.kosmetika_popisek)
      AGAINST("+'.$search.'*" IN BOOLEAN MODE)*2 +
      MATCH(kosmetika_produkt.kosmetika_popis)
      AGAINST("+'.$search.'*" IN BOOLEAN MODE) *1) as score
      FROM kosmetika_produkt
LEFT JOIN kosmetika_odstiny ON
kosmetika_produkt.id_kosmetika_produkt = kosmetika_odstiny.id_produkt_c
LEFT JOIN kosmetika_firmy ON
kosmetika_firmy.id_kosmetika_firmy = kosmetika_produkt.id_firmy_c
LEFT JOIN kosmetika_rada ON
kosmetika_rada.rada_id = kosmetika_produkt.rada_id
LEFT JOIN kosmetika_zarazeni ON
kosmetika_zarazeni.produktID=kosmetika_produkt.id_kosmetika_produkt
       WHERE kosmetika_produkt.online = 1
       AND kosmetika_produkt.id_kosmetika_produkt NOT IN (
                  SELECT kosmetika_produkt.id_kosmetika_produkt
                  FROM kosmetika_produkt
                  LEFT JOIN kosmetika_odstiny ON
kosmetika_produkt.id_kosmetika_produkt = kosmetika_odstiny.id_produkt_c
                  LEFT JOIN kosmetika_firmy ON
kosmetika_firmy.id_kosmetika_firmy = kosmetika_produkt.id_firmy_c
                  LEFT JOIN kosmetika_rada ON kosmetika_rada.rada_id =
kosmetika_produkt.rada_id
                  LEFT JOIN kosmetika_zarazeni ON
kosmetika_zarazeni.produktID=kosmetika_produkt.id_kosmetika_produkt
                  WHERE kosmetika_produkt.online = 1 AND
                  MATCH(kosmetika_firmy.nazev_firmy)
                  AGAINST("'.$search.'" IN BOOLEAN MODE) >= 1 OR
                  MATCH(kosmetika_produkt.kosmetika_nazev)
                  AGAINST("'.$search.'" IN BOOLEAN MODE) >= 1 OR
                  MATCH(kosmetika_rada.rada_nazev)
                  AGAINST("'.$search.'" IN BOOLEAN MODE) >= 1 OR
                  MATCH(kosmetika_odstiny.odstin_nazev)
                  AGAINST("'.$search.'" IN BOOLEAN MODE) >= 1 OR
                  MATCH(kosmetika_produkt.kosmetika_popisek)
                  AGAINST("'.$search.'" IN BOOLEAN MODE) >= 1 OR
                  MATCH(kosmetika_produkt.kosmetika_popis)
                  AGAINST("'.$search.'" IN BOOLEAN MODE) >= 1    )

  AND (
      MATCH(kosmetika_firmy.nazev_firmy)
      AGAINST("+'.$search.'*" IN BOOLEAN MODE) OR
      MATCH(kosmetika_produkt.kosmetika_nazev)
      AGAINST("+'.$search.'*" IN BOOLEAN MODE) OR
      MATCH(kosmetika_rada.rada_nazev)
      AGAINST("+'.$search.'*" IN BOOLEAN MODE) OR
      MATCH(kosmetika_odstiny.odstin_nazev)
      AGAINST("+'.$search.'*" IN BOOLEAN MODE) OR
      MATCH(kosmetika_produkt.kosmetika_popisek)
      AGAINST("+'.$search.'*" IN BOOLEAN MODE) OR
      MATCH(kosmetika_produkt.kosmetika_popis)
      AGAINST("+'.$search.'*" IN BOOLEAN MODE))
  ORDER BY score DESC LIMIT 50) ORDER BY score DESC LIMIT '.$stranka.',
'.$strankovatPo) or die (mysql_error());
  return $dotaz;
}

function nahradit ($vstup){
   return preg_replace("/[\&\-\;\.\']/", " ", $vstup);
}
```

```
function ulozVyhledavani($vyraz){
  if (mysql_num_rows(mysql_query("SELECT id FROM kosmetika_fulltext WHERE
vyraz='".$vyraz."'")) == 0)
    mysql_query("INSERT INTO kosmetika_fulltext
      VALUES (null, '".$vyraz."', 1)") or die (mysql_error());
  else
    mysql_query("UPDATE kosmetika_fulltext SET pocet_hledani=pocet_hledani
+ 1 WHERE vyraz='".$vyraz."'") or die (mysql_error());
}

function projdiSlovnik($slovnik, $stranka, $strankovatPo){
  $dotaz = mysql_query("SELECT kosmetika_slovnik_slova.slovo AS slovo
                        FROM kosmetika_slovnik_slova,
                        kosmetika_slovnik_vyraz
                        WHERE kosmetika_slovnik_vyraz.slovo_id =
                        kosmetika_slovnik_slova.id_slovo AND
                        kosmetika_slovnik_vyraz.vyraz = '".$slovnik."'")
        or die (mysql_error());
  if (mysql_num_rows($dotaz) > 0){
      $vysledek = mysql_fetch_array($dotaz);
      return fulltextDotaz($vysledek["slovo"], $stranka, $strankovatPo);
  }
}

function najdiPreklep($kde){
     if ($kde != ""){
        $shortest = -1;
        $query = mysql_query("SELECT kosmetika_nazev FROM kosmetika_produkt
WHERE kosmetika_nazev != '".$kde."'") or die (mysql_error());
        while ($data = mysql_fetch_array($query)) {
            $mezerZadano = substr_count($kde, " ");
            $pole = array();
            $exp = explode(" ", $data["kosmetika_nazev"]);
            $pozice = 0;
            $pocetPole = @ceil(count($exp) / $mezerZadano);
            if ($pocetPole >= 0){
              for($i = 0; $i < count($exp);$i++){
                  $kolikSlov = $mezerZadano;
                  $doPole = "";

                 for($a = 0; $a <= $kolikSlov; $a++){
            $doPole .= (isset($exp[$pozice]) ? $exp[$pozice] : "")." ";
            $pozice++;
                  }
                  $pozice -= $kolikSlov;
                  $pole[] = $doPole;
              }
            }else{
               $pole[] = $exp[$pozice];
            }
            foreach ($pole as $hodnota){
              $lev = levenshtein($kde, $hodnota);
              if ($lev == 0) {
              $closest = $hodnota;
              $shortest = 0;
                break;
              }

            if ($lev <= $shortest || $shortest < 0) {
            $closest  = $hodnota;
                  $shortest = $lev;
```

```
            }
          }
          if ($shortest == 1) break;              }
        echo "<br />";
        echo "Nechtěli jste vyhledat: <a
href='vyhledavani.php?s=fulltext&text=$closest'>$closest</a>?\n";
        echo "<br />";
     }
}
if (isset($_POST["fulltext"]))
  $_SESSION["hledano"] =  $_POST["fulltext"];
if (isset($_GET["text"]))
  $_SESSION["hledano"] =  $_GET["text"];
  $strankovatPo = 10;
if (!isset($_GET["stranka"]))
  $stranka = 0;
else
  $stranka = (intval($_GET["stranka"])-1) * $strankovatPo;
if (isset($_POST["fulltext"]) || isset($_SESSION["hledano"])){
  echo "<div id='vysledkyFulltext'>";
$search  = nahradit(strip_tags($_SESSION["hledano"]));
  echo "Hledali jste: <strong>". $search ."</strong><br /><br />";
ulozVyhledavani($search);

 $dotaz = fulltextDotaz($search, $stranka, $strankovatPo);
 $a = mysql_fetch_array(mysql_query("SELECT FOUND_ROWS() AS pocet"));

    if ($a["pocet"] == 0){
      $dotaz = projdiSlovnik($search, $stranka, $strankovatPo);
      $a = mysql_fetch_array(mysql_query("SELECT FOUND_ROWS() AS pocet"));
      if ($a["pocet"] == 0)
        najdiPreklep($search);
    }

    if ($a["pocet"] > 0){

          $pocetVysledku =  intval($a["pocet"]);
        if($stranka > 0)
          echo "<a
href='/vyhledavani.php?s=fulltext&stranka=".($stranka/$strankovatPo)."'>&la
quo; předchozí</a>";
        echo " | ";
        if($stranka + $strankovatPo < $pocetVysledku)
          echo "<a
href='/vyhledavani.php?s=fulltext&stranka=".($stranka/$strankovatPo
+2)."'>následující &raquo;</a>";
        while ($zaznam = mysql_fetch_array($dotaz)){
            echo '
         <div class="box">
          <table><tr><td><div class="znacka">
            <strong>Značka: </strong>'.$zaznam["nazev_firmy"].'</div></td>
            <td><div class="odstin"><strong>Kód:
            </strong>'.$zaznam["kosmetika_kod"].'</div></td></tr>
            <tr><td colspan="2"><div class="kategorie"><strong>Kategorie:
            </strong> '.$zaznam["kategorieID"].'</div></td></tr>
            <tr><td colspan="2"><div class="nazev"><strong>Název:
            </strong> '.$zaznam["kosmetika_nazev"].'</div></td></tr>
            <tr><td><div class="rada"><strong>Řada:
            </strong> '.$zaznam["rada_nazev"].'</div></td>
            <td><div class="odstin"><strong>Odstín:
            </strong>'.$zaznam["odstin_nazev"].'</div></td></tr>
```

```
            <tr><td colspan="2"><div class="popisek"><strong>Popisek:
            </strong> '.$zaznam["kosmetika_popisek"].'</div></td></tr>
            <tr><td colspan="2"><a href="#" class="zobraz">Zobrazit
            Popis</a></td></tr>
            <tr><td colspan="2"><div class="popis"><strong>Popis:
             </strong> '.$zaznam["kosmetika_popis"].'</div></td></tr>
           </table>
          <hr width="400" size="1" color="#dcdcdc" align="center" noshade>
         </div>
            ';
        }
        echo "<div class='strankovani'>";
        for($i = 1; $i <= ceil($pocetVysledku / $strankovatPo);$i++){
echo "<a href='/vyhledavani.php?s=fulltext&stranka=".$i."' ".($i ==
($stranka / $strankovatPo) + 1? "class='aktivni'" : "").">".$i."</a> - ";
        }
        echo "</div>";
        echo "</div>";
    }
}
ob_end_flush();
?>
```

---

vyhledavani.php

Skript zobrazí uživateli úvodní stranu této aplikace. Dále provádí kontrolu zadání nepovolených znaků.

```
<?php
  session_start();
  require "db.php";
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Language" content="cs"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" >
    <meta name="author" content="Ivana"/>
    <title>vyhledavani</title>
 <link rel="stylesheet" href="css/styly.css" type="text/css" media="all" />
    <script type="text/javascript" src="js/jquery-1.5.1.js"></script>
    <script type="text/javascript" src="js/akce.js"></script>
  </head>
  <body>
  <center>
   <br />
      <div class="nadpis"><a href="vyhledavani.php">FULLTEXTOVÉ
      VYHLEDÁVÁNÍ</a></div><br />
  <hr width="1000" size="3" color="#dcdcdc" align="center">
  <div class="stranka"><br />
  <div id="obsah">
   <?php
      if (isset($_GET['s'])){
            $soubor=$_GET['s'];
            $kontrola=similar_text("http://",$soubor);
```

```
            $kontrola2=similar_text("https://",$soubor);
            $kontrola3=similar_text("ftp://",$soubor);
            $kontrola4=similar_text("php://",$soubor);
            if ($kontrola>4 or $kontrola2>4 or $kontrola3>4 or
            $kontrola4>4){
            echo "<h3>Upozornění</h3>V odkazu je nepovolený znak!";
           exit;
         }
         $soubor2= dirname($_SERVER['SCRIPT_FILENAME'])."/".$soubor.".php";
             if(file_exists($soubor2)){
                    if(substr_count($soubor,"../")>0){
                           echo "<h3>Upozornění</h3>
                           Nelze nahrát soubor v nadřazeném adresáři!";
                    }elseif($soubor=="vyhledavani"){
                    echo "<h3>Upozornění</h3>Index nemůže načíst sám sebe!";
                    }else{
                       include $soubor2;
                     }
             }else{
                  include "error.php";
             }
      }elseif(isset($_GET['id'])){
        include "detail.php";
      }else{
        include "uvod.php";
      }
     ?>
       </div>
  <div id="levy">
      <div id='vysledkyNaseptavani'></div>
       <div class="povinne"><br />
       <strong>Fulltext</strong><br />
       <form action="vyhledavani.php?s=fulltext" method="POST"
id="formular">
        <input type="hidden" value='1' id='nastaveniFulletext' />
        <input type="text" size="20" name="fulltext" id='naseptavac' />
        <input type="submit" value="" style="background-
image:url(image/lupa.jpg); width: 27px; height: 27px;cursor:pointer;
margin:0px;padding:0px;" /> <br /><br />
        <span class='nastaveniLink'>Vypnout našeptávač</span>
        </form> <br /><br /></div> <br /><br />
    </div>
      </div>
  </body>
</html>
```

---

naseptavac.php

Skript porovnává zadávané znaky s názvy v databázi a zobrazuje box s odpovídajícími produkty.

```
<?php
header("content-type: text/html; charset=utf-8");
  if (isset($_POST["datainput"], $_POST["timestamp"])){
    require "db.php";
    function nahradit ($vstup){
       return preg_replace("/[\&\-\;\.\']/", " ", $vstup);
    }
```

```
    $returnHtml = "";
    $search  = nahradit(strip_tags($_POST["datainput"]));
    $dotaz = mysql_query(
'(SELECT kosmetika_produkt.id_kosmetika_produkt,
kosmetika_firmy.nazev_firmy, kosmetika_produkt.kosmetika_nazev,
            (MATCH(kosmetika_firmy.nazev_firmy)
             AGAINST("'.$search.'") * 6 +
             MATCH(kosmetika_produkt.kosmetika_nazev)
             AGAINST("'.$search.'") * 5) as score
 FROM kosmetika_produkt
 LEFT JOIN kosmetika_firmy ON
 kosmetika_firmy.id_kosmetika_firmy = kosmetika_produkt.id_firmy_c
 WHERE  kosmetika_produkt.online = 1  AND
         MATCH(kosmetika_firmy.nazev_firmy)
         AGAINST("'.$search.'") >= 1 or
         MATCH(kosmetika_produkt.kosmetika_nazev)
         AGAINST("'.$search.'")
 ORDER BY score DESC

  )UNION(
SELECT kosmetika_produkt.id_kosmetika_produkt, kosmetika_firmy.nazev_firmy,
kosmetika_produkt.kosmetika_nazev,
              (MATCH(kosmetika_firmy.nazev_firmy)
               AGAINST("+'.$search.'*" IN BOOLEAN MODE)*6 +
               MATCH(kosmetika_produkt.kosmetika_nazev)
               AGAINST("+'.$search.'*" IN BOOLEAN MODE)*5
              ) as score
       FROM kosmetika_produkt
            LEFT JOIN kosmetika_firmy ON
      kosmetika_firmy.id_kosmetika_firmy = kosmetika_produkt.id_firmy_c
      WHERE kosmetika_produkt.online = 1
            AND kosmetika_produkt.id_kosmetika_produkt
              NOT IN (
               SELECT kosmetika_produkt.id_kosmetika_produkt
               FROM kosmetika_produkt
               LEFT JOIN kosmetika_firmy ON
      kosmetika_firmy.id_kosmetika_firmy = kosmetika_produkt.id_firmy_c
              WHERE kosmetika_produkt.online = 1
          AND
            MATCH(kosmetika_firmy.nazev_firmy)
            AGAINST("'.$search.'" IN BOOLEAN MODE) >= 1 OR
            MATCH(kosmetika_produkt.kosmetika_nazev)
            AGAINST("'.$search.'" IN BOOLEAN MODE) >= 1)AND
            (MATCH(kosmetika_firmy.nazev_firmy)
            AGAINST("+'.$search.'*" IN BOOLEAN MODE) OR
            MATCH(kosmetika_produkt.kosmetika_nazev)
            AGAINST("+'.$search.'*" IN BOOLEAN MODE)
            )
        ORDER BY score DESC
        )
  ORDER BY score DESC LIMIT 0, 6') or die (mysql_error());
    while ($zaznam = mysql_fetch_array($dotaz)){

            $returnHtml .= "
              <div class='box'>
                 <table style='width:395px;'>
                  <tr><td colspan='2'><div class='nazev'><a
                  href='vyhledavani.php?id=".$zaznam
                  ['id_kosmetika_produkt']."'><strong>".$zaznam
                  ['nazev_firmy']."</strong> ".$zaznam['kosmetika_nazev']."
                  </a></div></td></tr>
```

```
                    </table>
                  </div>
                ";
            }
        $returnHtml = preg_replace("/\s+/", " ", $returnHtml);
        echo "{\"html\": \"".$returnHtml."\", \"timestamp\":
        \"".$_POST["timestamp"]."\"}";
    }
ob_end_flush();
?>
```

---

akce.js

Tento skript zobrazí nebo skryje popisek produktu. Další část kódu představuje našeptávač, který posílá data skriptu fulltext.php potřebná pro zobrazení boxu pro našeptávání. Skript dále provádí zapnutí nebo vypnutí našeptávače.

```
$(function() {
  $(".zobraz").click(function(){
    if($(this).parents(".box").find(".popis").css("display") == "none") {
      $(this).parents(".box").find(".popis").slideDown(1000);

      $(this).text("Skrýt Popis");
    }else{
      $(this).parents(".box").find(".popis").slideUp(1000);
      $(this).text("Zobrazit Popis");
    }

    return false;
  });

   //naseptavac
  $("#naseptavac").bind("click keyup", function(){
    if ($("#naseptavac").val() != "" && $("#nastaveniFulletext").val() ==
"1"){
      var genTimestamp = Math.random()+1;
$.post("naseptavac.php", {datainput: $("#naseptavac").val(), timestamp:
genTimestamp}, function(data){
         if (data.timestamp == genTimestamp){
if(data.html != ""){
             $("#vysledkyNaseptavani").html(data.html).show();
           } else {
                 $("#vysledkyNaseptavani").hide();
           }
         }
      }, "json");
    } else {
       $("#vysledkyNaseptavani").hide();
    }
  });
  $("body").click(function(){
     $("#vysledkyNaseptavani").hide();
  });

    var textyNastaveni = ["Zapnout našeptávač", "Vypnout našeptávač"];
  $(".nastaveniLink").click(function
      var neviditelnaHodnota = $("#nastaveniFulletext
```

```
      if(neviditelnaHodnota.val() == "1"){
      neviditelnaHodnota.val("0");
     }else{
            neviditelnaHodnota.val("1");
      }
      $(".nastaveniLink").text(textyNastaveni[parseInt(neviditelnaHodnota.v
al())]);
  });
});
```

---

styly.css

Kaskádové styly dokumentu.

```
/* CSS Document */
body {
      font-family: Calibri, Times New Roman, Arial CE;
      font-style: normal;
      font-size: 11pt;

      }
fieldset {
  border:0px;
}
option {
      font-family: Calibri, Times New Roman, Arial CE;
      font-style: normal;
      font-size: 11pt;
}
label.error { float: none; color: red; padding-left: .5em; vertical-align:
top; }
.stranka {
  width: 1000px;
  position: relative;
  text-align: center;
  margin: auto;
  }
.stranka img {
      border: 0px solid #CD853F;
}
#levy {
  position: absolute;
      width: 200px;
      top: 20px;
      height: 650px;
  text-align: center;
  margin: auto;
  border-color: #f5f5f5;
  left: 0px;
      border-width: 1px;
      border-style: solid;

}
#obsah {
  width: 780px;
  height: 650px;
      top: 90px;
      left: 202px;
```

```
  text-align: center;
  margin: 0 0 0 205px;
  border-color: #f5f5f5;
      border-width: 1px;
      border-style: solid;
}
.nadpis {
 color: #696969;
 font-variant: uppercase;
 font-style: Geneva CE, Calibri, Arial CE;
 font-weight:bold;
 font-size:30px;
}
.nadpis a, .nadpis a:hover{
 color: #696969;
 font-variant: uppercase;
 font-style: Geneva CE, Calibri, Arial CE;
 font-weight:bold;
 font-size:30px;
 text-decoration:none;
}
.povinne {
      background: #fffacd;
      color: #514721;
      border-color: #ffd700;
      border-width: 1px;
      border-style: solid;
      }
.povinne a{
  color: #0066cc;
}
.ramecek {
  background: #e6e6fa;
      color: #514721;
      border-color: #add8e6;
      border-width: 1px;
      border-style: solid;
}
.ramecek a {
  color: #0066cc;
}
.box {
  color: #514721;
      border-color: #dcdcdc;
      border-width: 0px;
      border-style: solid;
      text-align: left;
      padding: 1px
      }
.box table {
  width:770px;
}
.znacka {
  font-size: 10pt;
  text-align: left;
  font-weight: bold;
  padding: 1px
}
.odstin {
  font-size: 10pt;
  text-align: right;
```

```
  font-weight: bold;
  padding: 1px
}
.nazev {
  font-size: 10pt;
  text-align: left;
  padding: 1px
}
.popis {
  font-size: 10pt;
  text-align: left;
  padding: 1px;
}
#vysledkyFulltext .popis{
  display:none;
}
.popisek {
  font-size: 10pt;
  text-align: left;
  padding: 1px
}
.kategorie {
  font-size: 10pt;
  text-align: left;
  padding: 1px
}
.popisek p, .popis p {
padding: 0px;
margin: 0;

}
.rada {
  font-size: 10pt;
  text-align: left;
  padding: 1px
}

.strankovani{
  width: 760px;
  margin-bottom:10px;
}
.strankovani a {
  color:#000;
  text-decoration: none;
}

.strankovani a:hover{
  text-decoration: underline;
}

.strankovani .aktivni{
  font-weight:bold;
  color: blue;
}
#vysledkyNaseptavani{
  position:absolute;
  width:400px;
  margin-left: 10px;
  margin-top: 70px;
  background-color: #fff;
  text-align: left;
```

```
  border: 1px solid #000;
  display:none;
}
#vysledkyNaseptavani .box{
  padding: 0px;
}
#vysledkyNaseptavani .box:hover{
  background-color: #dcdcdc;
}
#vysledkyNaseptavani a{
  color: #514721;
  text-decoration: none;
}

.nastaveniLink{
      cursor: pointer;
      text-decoration:underline;
}
```
