

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií



BAKALÁŘSKÁ PRÁCE

Liberec 2009

Pavel Neděla

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 - Elektrotechnika a informatika
Studijní obor: 1802R022 - Informatika a logistika

Statistika WWW stránek v TLD „cz“

The statistics of WWW pages in TLD „cz“

Bakalářská práce

Autor: **Pavel Neděla**
Vedoucí práce: Mgr. Jiří Vraný

V Liberci 29.5.2009

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Pavel NEDĚLA**
Studijní program: **B2612 Elektrotechnika a informatika**
Studijní obor: **Informatika a logistika**

Název tématu: **Statistika WWW stránek v TLD ".cz"**

Z á s a d y p r o v y p r a c o v á n í :

1. Seznamte se s problematikou automatického zpracování www stránek a hromadného zpracování dokumentů.
2. Za pomoci dostupného software zpracujte statistiku WWW stránek v top level doméně ".cz". Výstupem z této statistiky by měl být mimo jiné - celkový počet stránek, zastoupení jednotlivých verzí html, zastoupení jazyků, verze kódování, využití metaznaků v hlavičce a další.
3. BP zpracujte v publikačním systému Latex či DocBookRozsah grafických prací: dle potřeby dokumentace

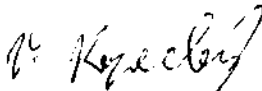
Rozsah grafických prací: **dle potřeby**
Rozsah pracovní zprávy: **cca 40 stran**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

- [1] <http://www.wikipedia.org/>
- [2] Další zdroje na Internetu.

Vedoucí bakalářské práce: **Mgr. Jiří Vraný**
Ústav nových technologií a aplikované informatiky

Datum zadání bakalářské práce: **30. října 2008**
Termín odevzdání bakalářské práce: **29. května 2009**


prof. Ing. Václav Kopecký, CSc.
děkan




prof. Dr. Ing. Jiří Maryška, CSc.
vedoucí ústavu

V Liberci dne 31. října 2008

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé BP a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce.

Datum: 29.5.2009

Podpis:

Poděkování

Tímto bych chtěl vyjádřit poděkování vedoucímu práce Mgr. Jiřímu Vranému za pomoc a rady poskytnuté při tvorbě této bakalářské práce.

Pavel Neděla

Abstrakt

Bakalářská práce se zmiňuje o základních principech webového prohledávání a především provádí statistiku webových stránek v top level doméně „cz“. Práce sleduje některé specifické údaje o webových stránkách, jako je například verze HTML, nebo druh kódování.

Abstract

Bachelor work mentions basic principles of web searching and mainly do statistics of the pages in top level domain „cz“. This work is monitoring some specific datas about web pages such as version of HTML or kind of coding.

Klíčová slova

web, prohledávání, statistika, HTML

Keywords

web, searching, statistics, HTML

Obsah

1	Úvod	8
2	O vyhledávání na internetu.....	9
2.1	Problematika webového prohledávání	9
2.1.1	Taktika výběru	10
2.1.2	Taktika znovunavštívení	10
2.1.3	Taktika zdvořilosti	11
2.1.4	Taktika paralelizace	12
2.2	Architektura webového prohledávače	12
2.3	Příklady některých webových prohledávačů	13
2.3.1	Prohledávače s veřejně publikovanou strukturou	13
2.3.2	Prohledávače komerčních společností	14
2.3.3	České prohledávací služby	16
3	Statistika webových stránek	17
3.1	Vstupní data	17
3.2	Statistický vzorek	19
3.3	Sledované informace	20
3.3.1	Kódování dokumentu	20
3.3.2	Verze HTML	21
3.3.3	Počet odkazů z hlavní stránky	22
3.3.4	Použití klíčových slov	22
3.3.5	Použití rámců	22
3.3.6	Použití CSS	22
3.3.7	Použití JavaScriptu	23
3.3.8	Použití VBScriptu	23
3.3.9	Použití Flash animací	23
3.3.10	Použití ActiveX	23
3.4	Získání statistických dat	24
3.4.1	Program pro získání dat	24
3.4.2	Vyhledávací hlediska	25
3.4.3	Průběh získání dat	25
3.4.4	Infiltrace	26
3.4.5	Nekomunikující stránky	26
3.4.6	Způsob vyhodnocení	27
3.4.7	Chybné údaje	27
3.4.8	Shrnutí	28
3.5	Závěr	31

Seznam symbolů, zkratk, termínů

Meta-značka: značka HTML kódu umístěna do záhlaví webové stránky. Je v ní umístěn například popis obsahu webu, titulek, klíčová slova pro prohledávání, apod. Meta-značky nevidí uživatel v prohlížeči, může je vidět pouze pokud si zobrazí kód HTML.

Top level doména: tzv. doména 1. řádu, pro Českou republiku je to „cz“.

PageRank: algoritmus pro ohodnocení důležitosti webových stránek. Tento algoritmus využívá strukturu hypertextových odkazů jako vzájemné „doporučování“ stránek. Hodnocení stránky se nepočítá z prostého počtu odkazů, které na ni vedou, ale bere v úvahu i hodnocení odkazujících stránek. Je základem vyhledávače Google.

Python: objektově orientovaný, multiplatformní programovací jazyk, nabízí významnou podporu k integraci s jinými programovacími jazyky a nástroji, je volně šiřitelný a díky jednoduché syntaxi se dá naučit během několika dní.

Webcrawling: (angl.) prohledávání webových stránek.

XML jazyk: „*Extensible Markup Language*“ – rozšiřitelný značkovací jazyk. Velmi obecný jazyk pro vytváření dokumentů obsahujících alespoň částečně strukturovaná data. Je vhodný hlavně pro výměnu dokumentů (např. objednávky a faktury), komunikaci (mezi aplikačními servery v internetu), nebo pro prezentaci na www stránkách.

1 Úvod

Webové technologie, tak jako celý obor informačních technologií, procházejí bouřlivým vývojem. Stávající technologie jsou nahrazovány technologiemi novějšími a každý, kdo se zabývá tvorbou, nebo správou webových stránek, musí tento vývoj sledovat. O tom, jak tento vývoj postupuje, hodně napoví použitá technologie na stávajících webových stránkách. Proto byla provedena tato statistika, která je zaměřena na použití některých technologií na webových stránkách. A protože nás samozřejmě nejvíce zajímá, jak je to v České republice, byla statistika zaměřena na webové stránky v „*top level*“ doméně „cz“. Jelikož získání dat pro takovéto statistiky přímo souvisí s problematikou prohledávání a prohledávačů na internetu, jsou zde také stručně zmíněny základy této problematiky.

Kapitola 2 popisuje problematiku vyhledávání na internetu a přibližuje stručné základy použitých technologií. Jsou zde také zmíněny některé problémy, které musí vyhledávače řešit. Jedná se o informace, které přímo souvisí s tématem bakalářské práce a následné zpracování statistik na ně navazuje.

V kapitole 3 je popsáno vlastní zpracování statistik, dále jsou zde popsány použité metody a postupy a jsou zde také popsány sledované veličiny, včetně důsledků, které mají na vlastnosti webových stránek. V závěru jsou také nastíněny další možné kroky do budoucna v souvislosti s touto statistikou.

2 O vyhledávání na internetu

Vyhledávání na internetu je specializovaná služba, která nám umožňuje vyhledat požadované údaje. Realizuje se zadáním vyhledávacího klíče (požadovaný údaj) do jednoduchého formuláře vyhledávacího nástroje a to prostřednictvím internetového prohlížeče tak, že výsledkem dotazu je seznam odkazů na internetové stránky, kde se požadovaný údaj nachází.

Služba převádí miliony hypertextových stránek do uspořádané a setříděné struktury. Tato struktura je vytvořena pomocí prohlížečů („robotů“ a „pavouků“ = „crawler“), což jsou speciální programy, které procházejí web a vytvářejí databázi internetových stránek – tzv. „index“. Lze to přirovnat k obrovské knize obsahující částečnou kopii každé webové stránky, kterou prohlížeč najde. Některé prohlížeče kladou větší důraz na klíčová slova, která se nacházejí v titulku, v nadpisech první úrovně, nebo v „meta“ informacích, než na ta, která se nacházejí v těle dokumentu. Důležitým faktorem je také frekvence výskytu klíčových slov. Kvalita získaného seznamu se měří množstvím vrácených adres, ale i poměrem duplicitních odkazů a subjektivním ohodnocením relevance odkazů, že vrácené odkazy skutečně odpovídají tomu, co jsme hledali. Prohlížeče mohou pracovat velmi rychle, protože pracují pouze s textem a nemusí načítat obrázky.

Vyhledávače fungují jednak jako nástroj vědy, výzkumu, případně jako zájmově orientované – často při univerzitách, nebo se jedná o servery komerčních organizací, které jsou nejčastěji dotovány reklamou umístěnou na stránkách této organizace.

2.1 Problematika webového prohlédávání

Hlavní dvě úskalí v problematice webového prohlédávání jsou jednak velké objemy dat a dále velká četnost změn webových stránek. Obrovské množství stránek je každým dnem přidáno, změněno, nebo zrušeno. Takovéto množství změn musí webový prohlížeč přidat, změnit, nebo odebrat každý den. Velké množství změn znamená, že webový prohlížeč může v daném čase stáhnout pouze zlomky webových stránek, takže musí dojít ke stanovení priorit při jejich stahování. Velké množství změn způsobuje, že mezitím co prohlížeč stáhl poslední verzi stránek webu, je velmi pravděpodobné, že současně byly již nové stránky na web přidány, nebo byly aktualizovány, či dokonce smazány. Poslední

stránky, které byly staženy, jsou s velkou pravděpodobností aktuální, ale nejstarší stažené stránky již pravděpodobně byly aktualizovány. Protože šířka pásma pro provádění „*webcrawlingu*“ není neomezená, je tedy nutné stanovit omezený, ale efektivní poměr mezi kvalitou a čerstvostí dat. Prohledávač musí pečlivě vybírat, kterou stránku navštíví v dalším kroku. Chování webového prohledávače je postaveno na kombinaci několika postupů (taktik):

- **Taktika výběru** („*selection policy*“) – které části jakých webových stránek stáhnout
- **Taktika znovunavštívení** („*re-visit policy*“) – které stránky kdy zkontrolovat, jestli tam nedošlo ke změnám
- **Taktika zdvořilosti** („*politeness policy*“) – u kterých stránek se vyvarovat přílišného zatížení downloadem
- **Taktika paralelizace** („*parallelization policy*“) – jak během prohledávání koordinovat rozložení prohledávačů mezi webovými stránkami

2.1.1 Taktika výběru

Díky rozlehlosti webu je dáno, že i velké vyhledávací stroje pokrývají jen část veřejně přístupného obsahu webu. Je třeba, aby stahované části webu obsahovaly jen ten nejdůležitější obsah stránek a ne jen náhodné ukázky webu. Tento požadavek je tou nejdůležitější prioritou. K tomu se používají speciální algoritmy, jako je například prohledávání do šířky („*breadth-first*“), analýza zpětných odkazů („*backlink-count*“), nebo dílčí „*PageRank*“.

2.1.2 Taktika znovunavštívení

Protože web je velmi dynamický, tak prohledání části webu může trvat velmi dlouho, řádově třeba i v týdnech, nebo měsících. Než prohledávač ukončí svou práci, nastane mnoho událostí, ať už se jedná o vytvoření nových stránek, aktualizace stávajících, nebo zrušení stránek. Nejčastějšími hodnotícími znaky pohlížejícími na tuto problematiku jsou „čerstvost“ („*freshness*“) a „stáří“ („*age*“). Čerstvost je binární míra, která indikuje zda uschovaná stránka je v současném čase aktuální, nebo ne. Stáří je míra, která indikuje, jak zastaralá je v současné době uchovaná stránka.

Snahou je, aby prohlédávач měl co nejvíce dat „čerstvých“, nebo-li mít co nejmenší „stáří“ uložených stránek. Tato hodnocení nejsou ekvivalentní, protože v první řadě se prohlédávач soustředí na to, kolik stránek není „čerstvých“ a teprve potom se zajímá o to, jaké je „stáří“ lokálních kopií stránek. V taktice znovunavštívení www stránek se používají dvě základní metody: „**rovnoměrná**“ metoda – zahrnuje znovunavštívení všech stránek pravidelně u všech stejně často, nezávisle na četnosti jejich změn a „**proporcionální**“ metoda, která zahrnuje častější znovunavštívení u stránek s četnější frekvencí aktualizací. Frekvence návštěv je u těchto stránek přímo úměrná četnosti aktualizací. V obou případech může být opakované prohlédání stránek také náhodné, nebo v pevném pořadí. Při pokusech byl zjištěn zajímavý výsledek, a to že doba průměrné čerstvosti rovnoměrné metody překonává proporcionální metodu. Vysvětlením je skutečnost, že když jsou stránky aktualizovány příliš často, prohlédávач bude marnit čas příliš častým navštěvováním velmi často se měnících stránek a nebude schopen udržovat jejich kopii v čerstvém stavu. Optimální řešení je někde mezi oběma metodami. Optimální metoda také zachovává určitou průměrnou frekvenci návštěv a ignoruje stránky, které se aktualizují příliš často. Optimální metoda je blíže rovnoměrné metodě, než k metodě proporcionální.

2.1.3 Taktika zdvořilosti

Používání webových „robotů“ se hodí pro řadu úloh, ale mají vliv na celou webovou komunitu. Používání webových robotů zahrnuje i negativní stránky věci:

- **Velká potřeba síťových zdrojů** - robot potřebuje velkou šířku pásma, protože pracuje s vysokým stupněm paralelismu během dlouhého časového úseku.
- **Zatížení serverů** – zvláště pokud je frekvence přístupů velmi velká.
- **Špatné naprogramování robotů** – mohou zavinit zhroucení serverů, či routerů.
- **Osobní roboty** - pokud se umístí u mnoha uživatelů mohou narušit síť a webové servery.

Částečné řešení těchto problémů je vyloučení vstupu robotů. Což je standardní postup administrátorů, kteří neumožní webovým robotům přístup do určitých částí svých webových serverů.

2.1.4 Taktika paralelizace

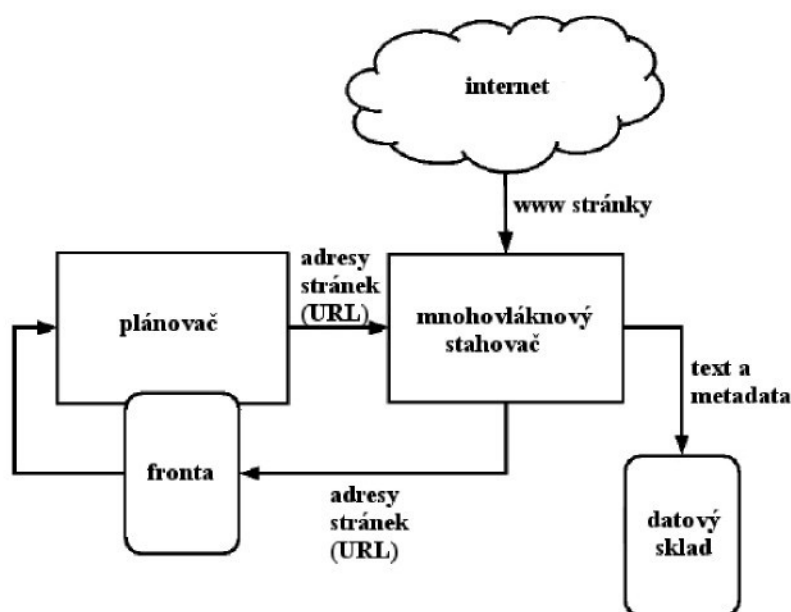
Paralelní prohlížeč je takový, který pracuje s více procesy současně. Cílem je maximalizace četnosti downloadu s minimalizací výpočetní režie. Je třeba se také vyvarovat opakovanému stažení té samé stránky, což vyžaduje koordinovanou techniku pro přiřazení nových webových adres, objevených během procesu procházení, protože ty samé stránky mohly být zároveň nalezeny paralelními procesy. Jedná se hlavně o tyto dvě techniky:

- **„Dynamické přiřazení“** – centrální server přiřadí novou adresu různým prohlížečům dynamicky. To znamená, že centrální server dynamicky vyrovnává zatížení každého z prohlížečů. S dynamickým přiřazením se typicky setkáme u systémů, které mohou přidat, nebo odebrat stahující procesy. Centrální server se může stát přetížený, tak některé z pracovních zátěží musí být přeneseny k rozložení prohlížečích procesů mezi více prohlížečích úloh.
- **„Statické přiřazení“**, kde jsou už od začátku prohlížení stanovená pevná pravidla, která definují jak přiřadit nové webové adresy prohlížečům. Efektivní přiřazovací funkce musí mít tři hlavní vlastnosti:
 1. Každý prohlížeč musí dostat přibližně ten samý počet odkazů.
 2. Pokud počet prohlížečích procesů narůstá, množství odkazů přiřazených každému z prohlížečů se musí zmenšit.
 3. Přiřazení musí být schopno dynamicky přidat, nebo odebrat prohlížečský proces.

2.2 Architektura webového prohlížeče

Prohlížeč musí mít dobrou prohlížečskou strategii, ale také musí mít vysoce optimalizovanou architekturu. Webové prohlížeče jsou základní částí

vyhledávacích strojů a detaily jejich algoritmů a jejich architektury jsou uchovávány jako obchodní tajemství. Pokud je konstrukce prohlédače publikována, tak je to velmi často bez detailů a podrobností, což je prevence před okopírováním výsledků práce. Je to však také z obavy před zveřejněním jejich třídícího algoritmu.



Obr. 1: Typická architektura webového prohlédače – zahrnuje plánovač i mnohováknový stahovač, dvě hlavní datové struktury – sklad webových stránek (text) a jejich adresy ve frontě.

2.3 Příklady některých webových prohlédačů

2.3.1 Prohlédače s veřejně publikovanou strukturou

- **Pavuk** – prohlédač vytvořený slovenským autorem, používá příkazovou řádku, nebo grafické rozhraní X-Windows. Jeho vývoj sice již nějakou dobu nepokračuje, ale i tak je dobře použitelný.

URL: < <http://pavuk.sourceforge.net/>>

- **WebSPHINX** – je vystavěn na základě knihoven tříd Java, kde je implementováno vícevláknové vyhledávání a rozbor webových stránek. Má grafické uživatelské rozhraní.

URL: < <http://www.cs.cmu.edu/~rcm/websphinx/>>

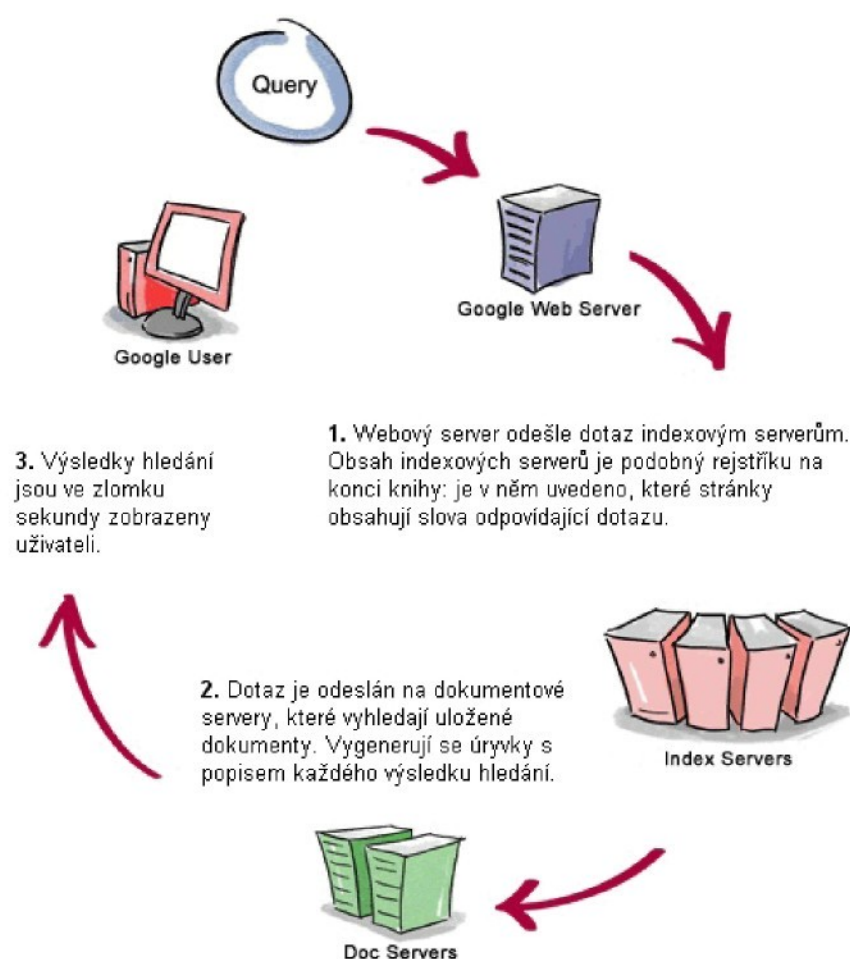
- **DataparkSearch** – i v současné době vyvíjený webový prohlížeč napsaný v jazyce C, s rozsáhlou dokumentací. Podporuje „*http*“, „*ftp*“, „*mtp*“, používá virtuální URL schémata pro indexování SQL databází, dokáže indexovat vícejazyčné stránky použitím „*content negotiation*“, což je mechanismus definovaný v „*http*“ specifikaci, který umožňuje obsluhovat různé verze dokumentu v rámci stejné URL adresy. K hodnocení oblíbenosti stránky využívá neuronový model sítě.
URL:<<http://www.dataparksearch.org/>>
- **Nutch** – prohlížeč kompletně napsaný v jazyce „*Java*“, ale jeho data jsou jazykově nezávislá. Má vysoce modulární strukturu umožňující vývojářům vytvořit zásuvné moduly („*pluginy*“) například pro rozbor typu médií, vyhledávání dat, či dotazování a znovuzískávání dat.
URL:<<http://lucene.apache.org/nutch/>>

2.3.2 Prohlížeče komerčních společností

- **Google** – momentálně zřejmě nejpoužívanější prohlížeč, jehož historii začali psát v roce 1996 dva studenti Stanfordské univerzity v Kalifornii – jako součást své závěrečné práce vytvořili hypotézu, podle které by vyhledávače mohly analyzovat vztahy mezi servery a podle toho by se určovalo pořadí výsledků. Jejich hypotéza byla správná a otevřela jim cestu k úspěchu a obrovskému bohatství. Postupně se začala prodávat inzerce, která souvisí s vyhledáváním klíčových slov. V současnosti nabízí Google tolik různých aktivit, že má postarano o průnik do všech možných sfér lidského života.
URL:<<http://www.google.cz>>
- **Yahoo!** - také za tímto prohlížečem stáli v počátcích dva studenti, kteří dostali v roce 1994 nápad vytvořit jakýsi webový katalog úplně nové koncepce. Zprvu do něj odkazy zařazovali sami, ale později díky jednoduchému skriptu mohli do katalogu zařazovat odkazy s krátkým popisem i uživatelé z celého světa. V roce 1995 byl katalog pojmenován na *Yahoo*, což je akronym názvu „*Yet Another Hierarchical Official Oracle*“, což je volně přeloženo něco jako: „Ještě jedno nadřazené dotěrné orákulum“. Současně slovo *Yahoo* odkazuje na bytost z knihy

„Gulliverovy cesty“ od Jonathana Swifta. Koncem roku 1995 již služba *Yahoo* zodpovídala 4 miliony dotazů denně. Od roku 1996 je z *Yahoo* víceúčelový projekt, ne jen katalog. Vytvořili svůj vlastní fulltextový vyhledávač. V roce 1997 se Jerry Yang, jeden ze zakladatelů společnosti, stává ve svých 29-ti letech nejmladším miliardářem na světě. V současné době firma zaměstnává osm tisíc lidí a jeho tržní hodnota je 50 miliard USD. *Yahoo* si drží druhou pozici mezi největšími vyhledávači.

URL:<<http://www.yahoo.com>>



Obr. 2: Vyhledávání serverem Google

zdroj: <http://www.lupa.cz>

2.3.3 České prohlédávací služby

- **Atlas** – provozuje ho společnost „Centrum Holdings“, která spravuje například také portál **Centrum**, vlastníkem je americká společnost. Vyhledávajícím uživatelům prezentuje hlavně reklamu (placené přednostní zápisy), nikoliv přirozené výsledky na předních místech hledání. Internetoví uživatelé si uvědomují, že hledání zde není to pravé a od těchto portálů se odklánějí. Portál je na ústupu.

URL:<<http://www.atlas.cz>>

- **Seznam** – v provozu od května 1996, byl postaven na základech portálu *Yahoo*, disponuje vlastním fulltextovým vyhledáváním „*Kompas*“. „*Kompas*“ je zaměřen na česká specifika ohledně kódování češtiny. Investuje do kvalitního vyhledávání a následuje zahraniční trendy. Je naším nejpoužívanějším vyhledávacím portálem.

URL:<<http://www.seznam.cz>>

3 Statistika webových stránek

3.1 Vstupní data

Vstupními daty pro vytvoření statistiky je seznam adres webových stránek, za které bude statistika provedena. V současné době je v české doméně „.cz“ zaregistrováno více jak 550 tisíc domén (zdroj: <http://www.nic.cz>). Původním záměrem bylo získat tento seznam od správce české domény, sdružení „CZ.NIC, z.s.p.o.“. Nicméně žádost o poskytnutí tohoto seznamu byla sdružením poměrně rezolutně odmítnuta. Nakonec byl seznam adres získán z katalogu: „<http://odkazy.seznam.cz>“. Jednalo se o „csv“ soubor se zhruba 100 tisíci záznamy. Tento soubor byl včetně popisků s položkami oddělenými čárkou. Tento soubor byl součástí dat poskytnutých společností Seznam, a.s., pro výzkum v oblasti www stránek na TUL.

```
"Cestovani","Bohušov","Neoficiální stránky vesnice Bohušov na Svitavsku.", "http://bohuvov.kvalitne.cz"
"Cestovani","Bohušovice","Aktuality, sportovní a kulturní akce, odkazy a diskuzní fórum.", "http://www.bohuvovice.i
"Cestovani","Bojnice","Atrakce, ubytování, služby a informace.", "http://www.bojnice.net/"
"Cestovani","Bojnice","Dovolená v Bojnících v samostatně stojící chatě na okraji lesa v chatové osadě Jánošík.", "h
"Cestovani","Bojnice","Informace o historii, programech a výstavách na zámku Bojnice.", "http://www.bojnicecastle.s
"Cestovani","Bojnice","O zámku, historii, expozice.", "http://www.skonline.sk/hrady/bojnice/zamok-bojnice.html"
"Cestovani","Bojnice","Ubytování v soukromí. Celoroční ubytování v centru Bojnic. Kompletně zařízený apartmán se s
"Cestovani","Bolína","Neoficiální stránky vesnice Bolína, která je částí obce Vlašim.", "http://web.telecom.cz/boli
"Cestovani","Bolíkovice","Neoficiální stránky obce Bolíkovice okres Třebíč. Obecní úřad Babice.", "http://www.obecb
"Cestovani","Bolívie","Co by vás mohlo o Bolívii zajímat.", "http://www.mujsweb.cz/www/bubak22"
"Cestovani","Bolívie","Putování po Bolívii, včetně popisu cesty z hor do hlavního města pěšky během blokády silnic
"Cestovani","Bolívie","Stránky popisují cestu napříč Bolívií.", "http://cestbol.blog.cz"
"Cestovani","Bolívie","Základní informace, poloha, historie, rozsáhlá fotogalerie.", "http://zeme.sopka.cz/zeme_det
"Cestovani","Bolívie a Chile 2003","Toulání po Bolívii a Chile v létě 2003. Humorný deník, lokální ceny, důležité
"Cestovani","Bolívie a Peru 2000","Cestování po Bolívii a Peru, jak je prožili Alfons, Hlad, Diki a Krup. Zápis z
"Cestovani","Bolívie, poslední indiánská země","Bolivijské Altiplano, džungle, jezero Titicaca, indiáni. Fotografie
"Cestovani","Bonesovy stránky o Egyptě","Pár fotek z Egypta.", "http://bones.websdarma.cz"
"Cestovani","Bor","Neoficiální www stránky města. Okres Tachov.", "http://mujsweb.cz/www/bor.u.tachova/"
"Cestovani","Bora Bora","Zážitky z cesty na ostrov Bora Bora.", "http://www.cafegalerie.cz/borabora"
"Cestovani","Borkovany","Malebná vesnička na jihu Moravy.", "http://www.sweb.cz/obec.borkovany"
"Cestovani","Borkovany","Neoficiální stránky malebné vinařské vesničky na jihu Moravy.", "http://borkovany.unas.cz"
"Cestovani","Borkovany","Neoficiální stránky obce Borkovany.", "http://www.borkovany.eu"
"Cestovani","Borneo 2000","Deník z cesty po Malajské části Bornea, fotky, zážitky a postřehy.", "http://www.geociti
"Cestovani","Bornholm","Deník a foto z dovolené na Bornholmu (2002). Cyklistika a koupání.", "http://www.naseverana
"Cestovani","Borovník","Jsme malá vesnička na úpatí Českomoravské vrchoviny.", "http://www.borovnik.vbs.cz"
"Cestovani","Borová Lada","Neoficiální stránka obecního úřadu Borová Lada. Informace o centrální Šumavě. Okres Pra
"Cestovani","Borová u Náchoda","Neoficiální stránky: z historie obce, zajímavosti, tipy na výlety do okolí, fotogr
"Cestovani","Borsko","Stránky obcí mikroregionu Borsko.", "http://www.borsko.jz.cz"
```

Obr. 3: Ukázka vzorku neupravených adres

Soubor bylo nutno upravit a získat z něj samostatné adresy. Postup úpravy vstupních dat proběhl následujícím způsobem:

1. **Odstranění popisků a komentářů** – k úpravě „csv“ souboru s oddělovači se nabízí Microsoft Excel, ale ten nedokáže zpracovat více jak 65.536 řádků. Takže byl využit Microsoft Access, který celý soubor načetl a

dokázal využít oddělovače tak, že oddělil odkazy od komentářů. Sloupce s komentářem byly potom v tabulce zrušeny, takže zůstal pouze sloupec s odkazy. Poté byla tabulka exportována zpět do textového formátu, kde každý odkaz je na samostatném řádku.

2. **Získání domén 2.řádu** – protože chceme statistiku stránek pod hlavní doménou, bylo třeba najít „čisté“ domény. Pro tento účel jsem napsal krátký konverzní program v programovacím jazyce „Python“. Jeho algoritmus je takový, že ze vstupního souboru bere řádek po řádku a v každém z nich hledá řetězec „.cz“. Pokud ho nenajde, pokračuje dalším řádkem. Tím zároveň vyloučíme odkazy jiných domén. Pokud tento řetězec najde, sestavuje zprava jméno domény 2.řádu až k další tečce a k němu zleva připojí řetězec „http://www“ a celý sestavený odkaz uloží do výstupního souboru. Program použitý v tomto kroku se jmenuje „konv_a.py“ – viz příloha A.
3. **Vyloučení duplicit** – pro tento účel nemělo smysl se zdržovat naprogramováním další utility, takže jsem použil nástroj, který znám z dřívější doby. Jedná se o „prastarý“ databázový nástroj „FoxPro 2.0“, který svému účelu výborně posloužil. Nejprve jsem textový soubor s adresami příkazem „IMPORT“ nahrál do databáze, příkazem „SORT“ setřídil a příkazem „TOTAL“ sjednotil duplicitní adresy. Výsledek jsem poté opět vyexportoval do textového souboru. Celá tato akce trvala asi čtyři minuty. Bylo jen ještě potřeba jednoduchým konverzním programem v textovém souboru „odříznout“ pravostranné mezery, které si doplnil databázový program. Poměrně nepříjemným překvapením byl fakt, že duplicitních záznamů bylo zhruba 75%. Největší počet duplicit i počty odkazů jsou v následujících tabulkách.

Tab. 1: Počty vět odkazů po úpravách

	Počet vět	%		
Neupravená data	103 561	100		100
Po vyloučení jiných domén	86 468	84	100	
Po vyloučení duplicit	20 860		24	20

Tab. 2: Adresy s největším počtem duplicit

Adresa	Počet duplicit	%
www.wz.cz	15 508	18
www.sweb.cz	7 414	9
www.mujweb.cz	5 982	7
www.estranky.cz	3 725	4

```

http://www.alenkaobchod.cz
http://www.ales-vesely.cz
http://www.alesnovak.cz
http://www.alesotypka.cz
http://www.alesradl.cz
http://www.alesvalenta.cz
http://www.alev.cz
http://www.alexfish.cz
http://www.alexneu.cz
http://www.alf.cz
http://www.alfa-proj.cz
http://www.alfacomp.cz
http://www.alfashop.cz
http://www.alfasport.cz
http://www.alfaunion.cz

```

Obr. 4: Ukázka vzorku upravených adres

3.2 Statistický vzorek

Po úpravách jsme k vypracování statistiky získali 20 860 webových adres z TLD „cz“. Budeme-li počítat, že v doméně „cz“ je registrováno 550 tisíc domén druhé úrovně, je náš statistický vzorek přibližně 3,8 %. Statistické metodiky zveřejňované na internetu, včetně stránek Českého statistického úřadu, se vyhýbají určení procenta velikosti vhodného statistického vzorku. Samozřejmě, že čím větší vzorek, tím relevantnější výsledky, ale se zvětšením statistického vzorku prudce narůstá časová náročnost zpracování a zvětšení objemu zpracovávaných dat. Vzhledem ke všem okolnostem, pokládám získaný počet adres pro provedení statistiky za dostatečný a tím výsledky statistiky za průkazné.

3.3 Sledované informace

3.3.1 Kódování dokumentu

Počítač chápe každé písmeno jako číslo od 0 do 255 (jeden bajt). Anglicky mluvící uživatelé si vystačí se 128 znaky. Těmto základním znakům bez háček a čárek se říká ASCII. Dalšími čísly od 128 do 255 se označily znaky s diakritikou jazyků západní Evropy, a tak vznikl kód „*Latin-1*“. V něm kromě znaků „á, í, š, ž“ žádné jiné české znaky nejsou. Pro neazbukové jazyky střední a východní Evropy vznikly různé konvence, které zachovávají význam prvních 128 znaků ASCII a dalších 128 znaků si definují po svém. A právě různá přiřazení znaků s diakritikou číslům od 128 do 255 se označují jako kódování (případně jako znaková sada). Vzniklo několik různých kódování češtiny a každé to vyřešilo jinak. Dnes nejčastěji používaná kódování jsou:

- **ISO 8859-1** – jedná se o západoevropské kódování ISO (nebo též „*Latin1*“), které neobsahuje všechny české znaky (viz předchozí odstavec). Pokud jde o webové stránky v češtině, je jeho použití chybou.
- **ISO 8859-2** – kódování používané na Unixu a na Linuxu, ale i v řadě programů pro Windows. Někdy se označuje jako „*Latin2*“. Microsoft ho označuje jako „Středoevropské jazyky ISO“. Jeho použití na webových stránkách psaných českým jazykem je v pořádku.
- **Windows-1250** – je preferováno pod operačními systémy Windows. Jeho obliba na webových stránkách spočívá zejména v tom, že její většina editorů www stránek pod tímto operačním systémem používá jako základní kódování (např. FrontPage). Jeho použití na českém webu je v pořádku.
- **UTF-8** – jedná se o tzv. znakovou sadu „*Unicode*“ a je v současnosti nejčastěji používaným kódováním. „*Unicode*“ je na rozdíl od předešlých znakových sad určeno pro všechny světové jazyky

najednou, protože znakům přiřazuje čísla až do 16-ti miliónů (zapisuje se dvěma bajty). Jde o nejmodernější kódování a všeobecná podpora je teprve hudbou budoucnosti, ale v internetových prohlížečích je podporované dobře. Má velkou výhodu v tom, že v něm jde psát ve více jazycích najednou. Pro použití na stránkách českého webu je nejvhodnější.

3.3.2 Verze HTML

HTML jako značkovací jazyk pro popis webové stránky prošel velkým vývojem a na internetu se objevuje v různých verzích. Nejvíce se objevuje v těchto verzích:

- **HTML 3.2** – dnes již dosti stará verze (byla vydána v lednu 1997), dříve dosti používaná. Pořád ji na internetu můžete vidět.
- **HTML 4.0** – byla vydána v prosinci 1997. Do specifikace jazyka přibylly nové prvky pro tvorbu tabulek, formulářů a nově byly začleněny „rámy“ („frames“).
- **HTML 4.01** – vydána v prosinci 1999. Opravuje některé chyby verze 4.0. Je to poslední verze jazyka HTML.
- **XHTML 1.0** – nová forma jazyka pro popis webové stránky, jedná se v podstatě o jinou skupinu jazyků. XHTML je aplikací jazyka XML nad HTML. Specifikace byla vydána v roce 2000 a revidována byla v roce 2002. Jeho podpora je v současných internetových prohlížečích stejná jako podpora HTML. Je výrazně přísnější na validitu napsaného kódu. Řada tvůrců webových stránek ho proto nemá ráda. Používá se ve verzích „*Transitional*“ (přechodová), kde lze psát i značky HTML, verze „*Strict*“ (striktní), která je přísnější na dodržování specifikace a verze „*Frameset*“, která je totožná s verzí „*Transitional*“, ale přidává ještě podporu pro rámce.

- **XHTML 1.1** – Je velice podobné „*XHTML 1.0 Strict*“. Je to modulově založená forma jazyka. Zde je již také řada věcí zakázána.
- **XHTML 2.0** – Vývojové stádium jazyka. Není zamýšleno, aby bylo zpětně kompatibilní se svými předchůdci.

3.3.3 Počet odkazů z hlavní stránky

Měřítko provázanosti mezi webovými stránkami. Často ho používají prohlížeče k ohodnocení „bonity“ webové stránky. V řadě případů jde špatně automaticky rozlišit mezi odkazy na jiný web a odkazy na jinou část těch samých webových stránek, proto tento údaj bude trochu nepřesný.

3.3.4 Použití klíčových slov

Klíčová slova jsou určena pro vyhledávače a měla by krátce a výstižně charakterizovat „o čem“ web je. Použití klíčových slov ještě neznamená, že stránka bude na vrcholu při vyhledávání.

3.3.5 Použití rámců

Rámy, nebo také rámce, nebo-li „*frames*“ je technologie, která rozdělí webovou stránku do obdélníkových oblastí, které mohou být na sobě nezávislé a lze do každé z nich načítat jinou webovou stránku. Dříve velmi oblíbené, v dnešní době již zastaralá technologie.

3.3.6 Použití CSS

CSS, nebo-li „kaskádové styly“ jsou technologií, která vznikla přibližně v roce 1997 a řeší podobu („webdesign“) webových stránek (pozadí, barvu a styl písma, zarovnání, atd.) tím způsobem, že definice, která platí pro celý web je nadefinována v malém parametrickém souboru a v HTML kódu se již tyto záležitosti neřeší. Jedná se o moderní technologii.

3.3.7 Použití JavaScriptu

JavaScript je programovací jazyk, který se používá na webových stránkách. Zapisuje se přímo do HTML kódu. Má syntaxi podobnou jazyku Java. Jeho použití je jednoduché. Jedná se o klientský skript, což znamená, že se program odesílá se stránkou do internetového prohlížeče uživatele a tam se teprve vykonává. Nelze ho spustit jinde, než na webových stránkách. Jeho možnosti jsou velmi široké. Výrazně rozšiřuje funkčnost webu. Tím, že se vykonává až v prohlížeči uživatele může být potenciálně nebezpečný a zneužitelný. V internetovém prohlížeči ho lze vypnout.

3.3.8 Použití VBScriptu

Funguje stejně jako JavaScript, ale je založen na syntaxi jazyka Visual Basic. Již starší technologie. Oblíbena mezi tvůrci virů. Téměř se již nepoužívá.

3.3.9 Použití Flash animací

Flash animace „rozhýbávají“ a oživují webové stránky nejrůznějšími pohyblivými výtvary. Jedná se o kontroverzní záležitost. Často se v nich tvoří reklamy a jiné „vlezlé“ a rádobý zábavné klipy. Možnosti Flash technologie jsou velmi široké a lze v nich vytvořit i jednoduchou hru.

3.3.10 Použití ActiveX

Další technologie, která se na webu vedle JavaScriptu, VBScriptu a Flash technologie stará o interaktivitu. Prvky ActiveX se stahují ze stránek jako doplňky a integrují se do prohlížeče. S ActiveX pracují různé služby jako online antivirové programy, různé přehrávače. Velkým problémem je riziko zanesení viru, nebo jiného škodlivého softwaru (spyware). Jakmile totiž uživatel spustí doplněk, otevře se tím pomyslná brána do systému. Protože se jedná o technologii firmy Microsoft, podporu pro tyto doplňky najdeme jenom v Internet Exploreru.

3.4 Získání statistických dat

3.4.1 Program pro získání dat

Pro získání statistických dat byl v jazyce „*Python*“ vytvořen program, který bere webové adresy z vytvořeného seznamu a z těchto webů sbírá informace. Program pro získání dat se jmenuje „ziskdat.py“ – viz příloha A. Algoritmus programu je následující:

- Program načte adresu webové stránky ze vstupního souboru
- Proběhne test dostupnosti webové stránky. Pokud je webová stránka nedostupná, zapíše se tato informace do log souboru a načte se další adresa ze vstupního souboru.
- Pokud je web dostupný, načte se výchozí stránka tohoto webu do paměti.
- Ze stránky jsou získány požadované údaje dle vyhledávacích hledisek – viz níže. Program se snaží zohlednit i některé možné chyby v kódu stránky. Například pokud narazí na ukončovací znak „>“ místo očekávaného apostrofu.
- Program samozřejmě počítá i s tím, že požadované údaje na stránce nenalezne.
- Následuje zápis zjištěných údajů do výstupního souboru, který je v textovém formátu. Program také pro urychlení vyhodnocení, většinu údajů (kromě verze HTML a kódování) průběžně napočítává.
- Program aktuální data po zpracování každé stránky ukládá i s informací o poslední adrese zpracované stránky. Tyto údaje se po zpracování každé stránky přepisují. Provádí se to z toho důvodu, aby v případě havárie celého procesu nepřišla nazmar mnohahodinová práce a dalo se pokračovat od následující adresy.

- Po zpracování poslední adresy se napočítávané údaje zapíší do log souboru

3.4.2 Vyhledávací hlediska

Zde je seznam vyhledávacích hledisek pro vyhledávání statistických údajů. Vyhledává se podle textových řetězců, jejichž výskyt je typický pro přítomnost určité technologie.

Tab. 3: Seznam vyhledávacích hledisek

Údaj	Vyhledávací hledisko
Verze HTML	Údaj, který následuje po řetězci: „ W3C //DTD “ k druhé mezeře
Kódování	Údaj, který následuje po řetězci: „ charset =“ k dalšímu apostrofu
Počet odkazů	Počet výskytů řetězce: „ href =”http://“
Použití klíčových slov	Existence řetězce: „ keywords “
Použití rámců	Existence řetězce: „ frameset “
Použití CSS	Existence řetězce: „ text/css “
Použití JavaScript	Existence řetězce: „ text/javascript “
Použití VBScript	Existence řetězce: „ text/vbscript “
Použití Flash	Existence řetězce: „ .swf “
Použití ActiveX	Existence řetězce: „ activex “

3.4.3 Průběh získání dat

Veškeré zpracování, včetně získání dat z webových stránek, vyhodnocení i přípravy webových adres proběhly na přenosném počítači (notebooku) s již postarší konfigurací: Intel Pentium M 750 / 1862 MHz, 1024 Mb RAM, HDD 80Gb. Operační systém Windows XP Pro SP3. Připojen k internetu technologií ADSL o rychlosti 8192/512 kbit/s.

Získání statistických dat neproběhlo najednou, protože program během zpracování narazil na několik webových stránek, které byly zaheslovány. Protože to však nebyly stránky, které nekomunikují, nezafungovalo opatření, které řeší tzv. „timeout“ (čímž máme na mysli časovou prodlevu, která je delší než přijatelná doba načítání stránky) a zpracování na nich „uvázlo“. Bylo nutné program ukončit a pokračovat následující webovou adresou. Zaheslované stránky byly ze zpracování vyloučeny. Zpracování tedy proběhlo ve čtyřech částech a celkový výsledek byl sloučen z těchto dílčích částí. Tento sběr dat trval přibližně 9 hodin.

3.4.4 Infiltrace

Při sběru dat z webových stránek byly získány i statistické údaje, se kterými se předem nepočítalo, ale byla by škoda je nezmínit. Jedním z nich je i počet zachycených infiltrací. Během zpracování rezidentní štít používaného antivirového programu (NOD32 Antivirus 4 Business Edition) zachytil na 17-ti stránkách virové infiltrace. Jednalo se vždy o tzv. „Trojské koně“. Vzhledem k počtu prohledaných stránek se jedná přibližně o 0,09%. Vypadá to sice jako malé číslo, ale vzhledem k obrovskému množství stránek na internetu je pravděpodobnost napadení dosti velká.

3.4.5 Nekomunikující stránky

Další kapitolou jsou stránky, se kterými se nepodařilo navázat spojení. Internet je takové stále živé médium, kde neustále vznikají stránky nové a jiné zase naopak zanikají. Nenalezení některých stránek ze seznamu adres svádí k interpretaci, že se jedná o již zaniklé stránky. Ale rozhodně tomu tak být nemusí. Může jít o dočasnou nedostupnost zaviněnou například výměnou webové technologie, technickými problémy serveru, dočasnou ztrátou konektivity, apod. Ať je tomu tedy z jakýchkoliv důvodů, to číslo je dosti zajímavé.

3.4.6 Způsob vyhodnocení

Jak již bylo sděleno v předchozích částech, většina údajů statistiky byla průběžně načítána během sběru dat. To ale jde dobře u údajů, které dávají výsledek typu ANO / NE, nebo třeba u výpočtu průměrného počtu odkazů. Ne však u údajů o typu kódování a verzi HTML. Zpracování těchto dvou údajů by bylo v textovém formátu velmi komplikované. Programovací jazyk „*Python*“ má vestavěnou podporu pro práci s SQL databází, takže jsem se rozhodl tato data převést do databáze SQL a zde je vyhodnotit. Jedná se o řešení, které je velmi rychlé a efektivní. Stačilo napsat několik drobných programů pro import do SQL a vyhodnocení. Program pro převod z textového formátu do SQL se jmenuje „do_sql.py“ – viz příloha A. Programy pro vyhodnocení uvedených veličin se jmenují „vyh_ver.py“ a „vyh_kod.py“ – viz příloha A.

3.4.7 Chybné údaje

Při vyhodnocování údajů o přítomnosti, nebo nepřítomnosti určité technologie na webových stránkách je to celkem jednoduché. Buď to tam je, nebo není a nic mezi tím. Ale v případě údajů o verzi HTML a kódování tomu tak není. Bylo zajímavé sledovat jak v řadě webových stránek nejsou validní údaje. Bylo možné se podívat nad údajem o kódování, který byl: „czech1250“, „iso-1250“, „latin2“, „utf=8“, „windows-c“, nebo údaj o verzi s hodnotou: „XHTML 4.01“. Na níže uvedeném obrázku je malá ukázka, jak na některých webech může vypadat zápis kódování „iso-8859-2“:

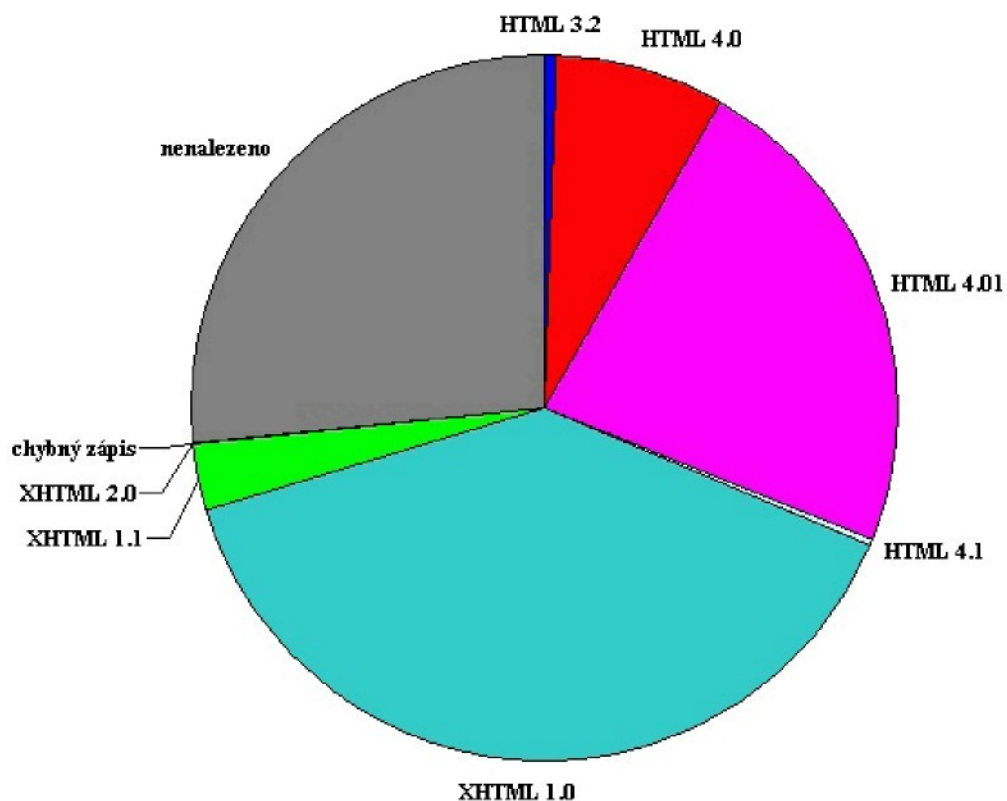
```
'iso 8859-2'  
'iso 8859-2 '  
'iso-8859-2;  
'iso_8859-2'
```

Obr. 5: Ukázka chybného zápisu kódování

3.4.8 Shrnutí

Tab. 4: Statistika verzí HTML

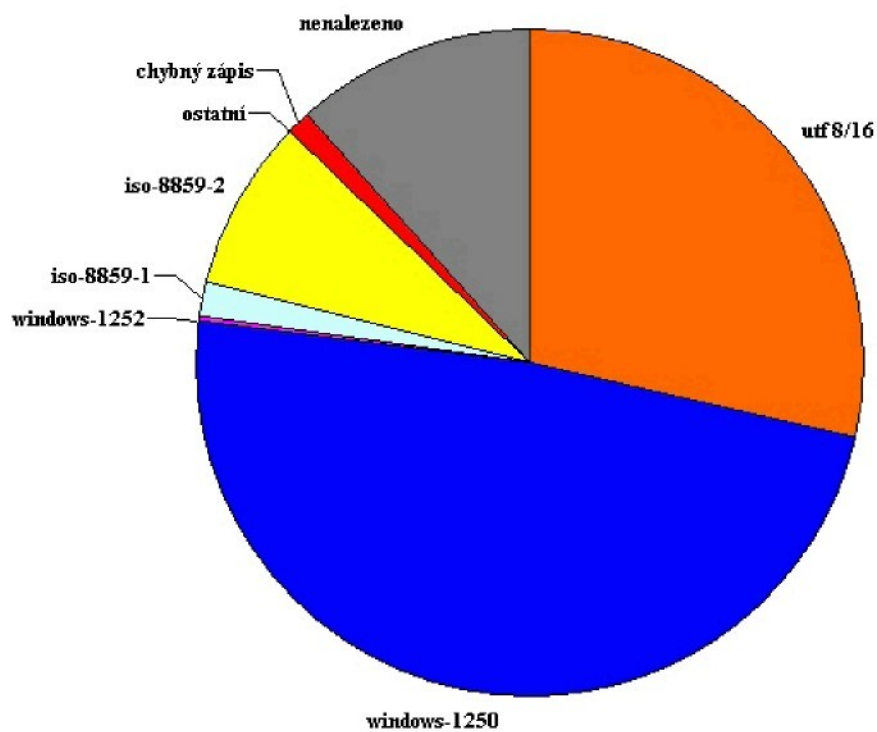
Verze HTML	Počet	%
HTML 3.2	125	0,641
HTML 4.0	1477	7,580
HTML 4.01	4486	23,022
HTML 4.1	6	0,031
XHTML 1.0	7583	38,915
XHTML 1.1	597	3,064
XHTML 2.0	1	0,005
chybný zápis	10	0,051
nenalezeno	5201	26,691



Graf. 1: Statistika verzí HTML

Tab. 5: Statistika kódování

Kódování	Počet	%	Typ kódování
utf-8	5596	28,719	Unicode
utf-16	2	0,010	Unicode
windows-1250	9409	48,286	Windows – východní Evropa
windows-1251	2	0,010	Windows – azbuka
windows-1252	63	0,323	Windows – západní Evropa
windows-1254	1	0,005	Windows – turecké kódování
iso-8859-1	318	1,632	ISO – západoevropské
iso-8859-2	1594	8,180	ISO – středoevropské
us-ascii	12	0,062	ASCII
chybný zápis	227	1,165	
nenalezeno	2262	11,608	

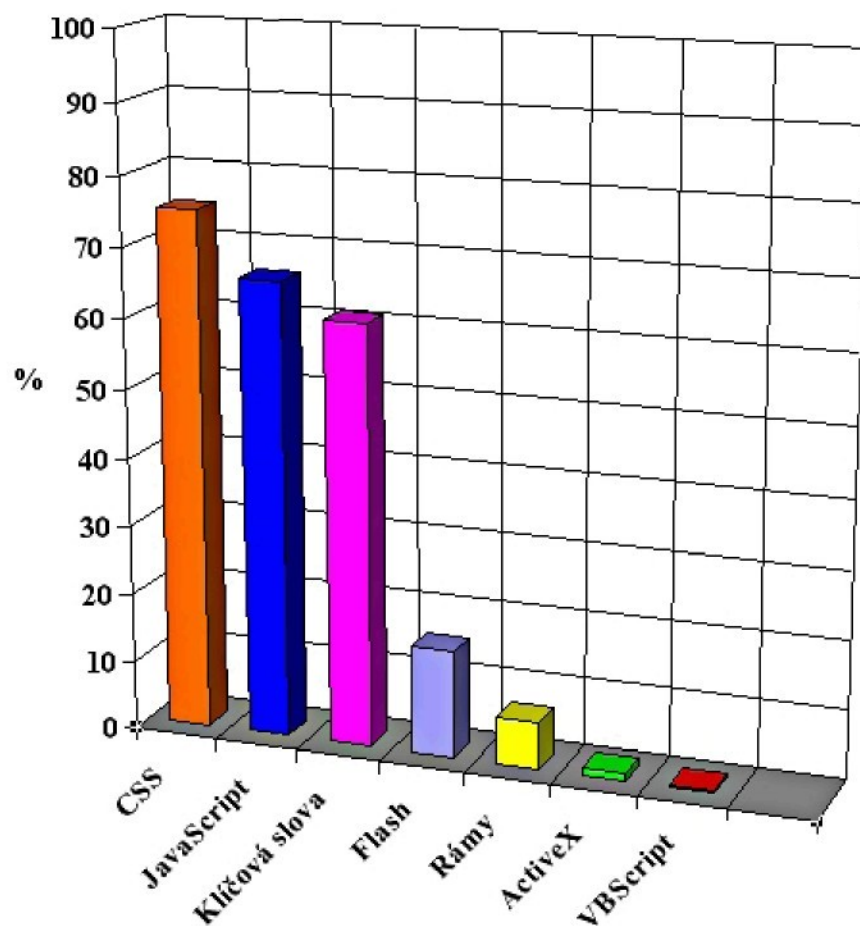


Graf. 2: Statistika kódování

Tab. 6: Rekapitulace

POČET ADRES CELKEM:	20 860	100 %
Z TOHO NEKOMUNIKUJÍCÍCH:	1 368	7 %
ZAHESLOVANÝCH:	6	
ZPRACOVANÝCH:	19 486	93 %
Průměrný počet odkazů z hlavní stránky:	28	
Počet stránek používajících klíčová slova:	11 868	60,91 %
Počet stránek používajících rámy:	1 332	6,84 %
Počet stránek používajících CSS:	14 626	75,06 %
Počet stránek používajících JavaScript:	12 845	65,92 %
Počet stránek používajících VBScript:	70	0,36 %
Počet stránek používajících Flash technologii:	3 036	15,58 %
Počet stránek používajících ActiveX:	199	1,02 %

Poznámka: procentní míra udávaná u jednotlivých technologií je počítána vzhledem k počtu zpracovaných stránek.



Graf. 3: Přehled použití technologií

3.5 Závěr

- Co se týká verzí HTML, je vidět, že se lze dnes setkat i s dosti starými verzemi, dále je vidět, že verze HTML 4.01 je i dnes poměrně oblíbená, jelikož ji používá zhruba jedna čtvrtina webů. Z těchto výsledků je zřejmé, že poměrně velká část majitelů webových stránek je ukolébána vlastní spokojeností, možná i konzervativismem a nemá příliš zájem na používání nových technologií. Ze statistiky je vidět postupný nástup XHTML, která je již použita téměř u poloviny webů. Poměr webů používajících XHTML bude postupem doby čím dál více narůstat. Poměrně zarážející je velký počet webů (více jak čtvrtina), kde tento údaj nelze dohledat. Na tyto „nezařazené“ weby můžeme téměř s jistotou pohlížet jako na weby se staršími verzemi HTML, protože technologie XHTML je přísnější na validitu kódu. Dalším důvodem je velká benevolence prohlížečů vůči chybám, přičemž spousta „rádobywebmasterů“ nemá tušení o tom, že existuje něco jako validátor kódu, a tudíž nepřijdou ani na to, že mají něco špatně – vždyť stránka se zobrazí správně. Tento pohled nám však výrazně mění poměr mezi technologií HTML a XHTML, jenže to nemůžeme přímo dokázat bez nahlédnutí do kódu stránky.
- Z přehledu kódování je vidět výrazná převaha (téměř polovina) kódování „*windows-1250*“, což znamená jasnou převahu operačních systémů Microsoft Windows. Lze ale již také vypožorovat nástup kódování „*Unicode*“, které je použito téměř v jedné třetině případů a dá se očekávat, že jeho podíl bude čím dál více narůstat. Zajímavé je nalezení několika zástupců u nás exotického kódování, jako je azbuka, nebo turečtina. Těžko se dá bez nahlédnutí do kódu stránek odhadnout jestli se jedná o úmysl, nebo o omyl.
- Průměrný počet odkazů vedoucí z hlavní stránky (28 odkazů) je překvapivě velký. Mnohé materiály pojednávající v poslední době o

internetových technologiích počítají s průměrným počtem sedmi odkazů – například [4]. Zřejmě to je známka toho, že internet se stává více provázanější, nebo je možné, že si autoři webů uvědomují důležitost odkazů vzhledem k umístění ve vyhledávačích. Namátkově jsem zjistil, že rozdělení počtu odkazů v různých webech je velmi nevyrovnané. Existují weby, které mají na hlavní stránce stovky odkazů.

- Používání klíčových slov je u nás dle výsledků již téměř samozřejmé a jsou použita u více než šedesáti procent webů. O jejich užitečnosti snad nikdo nepochybuje.
- Nízký počet webů, které používají rámy, „VBScript“, nebo „ActiveX“ odráží jejich zastaralost. U technologie „ActiveX“ jsou to i jeho nevýhody, které již byly popsány výše.
- Zato kaskádové styly (CSS) jsou aktuální technologií a je to vidět i ve statistice – 75% mluví samo za sebe. Nelze se ovšem divit, protože v tomto případě jsou výhody pro tvorbu stránek naprosto zřejmé.

Na závěr se můžeme zamyslet nad tím, jak s těmito statistikami pokračovat dál a jak na ně navázat. Určitě může být zajímavé celý test provést znovu, třeba po půl roce. Také můžeme rozšířit okruh testovaných veličin. Ale můžeme také celý test povýšit o dimenzi výš. Můžeme rozšířit funkčnost programu pro sběr dat tak, aby nepočítal jen počet odkazů, ale aby tyto odkazy ze stránek extrahoval a tím celou adresní bázi dat rozšiřoval. Další statistiky by se již mohly provádět s čím dál větším počtem adres. A tady nám vzniknou další veličiny pro statistické sledování - přírůstek a úbytek adres, což nám může poskytnout další, velmi zajímavé statistické údaje.

Seznam použité literatury

- [1] PÁNEK Karel. *Architektury a modely webových strojů*. Lupa.cz 6.2.2002
URL: < <http://www.lupa.cz/clanky/architektury-a-modely-webovych-stroju/>>
- [2] CASTILLO Carlos. *Effective web crawling*. University of Chile. 2004
URL: <http://chato.cl/research/crawling_thesis>
- [3] ŠEBÍK Antonín. *Historie společnosti Yahoo!* Vseohw.net. 27.7.2007
URL: <<http://vseohw.net/clanky/historie/yahoo>>
- [4] BARABÁSI A.-L.. *V pavučině sítí*. Paseka. 2005
- [5] internet, zejména stránky:
URL: < <http://www.wikipedia.cz>>
URL: < <http://www.jakpsatweb.cz>>
URL: < <http://www.vyhledavace.info/>>
URL: < <http://www.lupa.cz/>>
URL: < <http://google.pogar.info/historie-spolecnosti-google/>>

DataparkSearch Engine 4.51

Reference manual

DataparkSearch Engine 4.51: Reference manual

Copyright © 2003-2008 OOO DataPark

Copyright © 2001-2003 Lavtech.com corp.

This project is dedicated to Noémie.

Table of Contents

1. Introduction.....	1
1.1. DataparkSearch Features.....	1
1.2. Where to get DataparkSearch.	3
1.3. Disclaimer.....	3
1.4. Authors.....	3
1.4.1. Contributors.....	3
2. Installation.....	5
2.1. SQL database requirements	5
2.2. Supported operating systems	5
2.3. Tools required for installation	6
2.4. Installing DataparkSearch.....	7
2.5. Possible installation problems.....	9
2.6. Installation registration.....	10
3. Indexing	11
3.1. Indexing in general.....	11
3.1.1. Configuration.....	11
3.1.2. Running indexer	11
3.1.3. How to create SQL table structure	11
3.1.4. How to drop SQL table structure.....	11
3.1.5. Subsection control	12
3.1.6. How to clear database.....	12
3.1.7. Database Statistics	12
3.1.8. Link validation.....	13
3.1.9. Parallel indexing	13
3.2. Supported HTTP response codes	14
3.3. Content-Encoding support	15
3.4. Stopwords.....	16
3.4.1. StopwordFile command	16
3.5. Clones.....	16
3.5.1. DetectClones command.....	16
3.6. Specifying WEB space to be indexed	16
3.6.1. Server command.....	19
3.6.2. Realm command	19
3.6.3. Subnet command	19
3.6.4. Using different parameter for server and it's subsections	20
3.6.5. Default indexer behavior	20
3.6.6. Using indexer -f <filename>	21
3.6.7. ServerDB , RealmDB , SubnetDB and URLDB commands	21
3.6.8. URL command.....	21
3.7. Aliases.....	21
3.7.1. Alias indexer.conf command.....	22
3.7.2. Different aliases for server parts.....	22
3.7.3. Using aliases in Server commands	22
3.7.4. Using aliases in Realm commands	23
3.7.5. AliasProg command.....	23

3.7.6. ReverseAlias command	24
3.7.7. Alias command in <code>search.htm</code> search template	25
3.8. Servers Table	25
3.8.1. Loading servers table	25
3.8.2. Servers table structure	26
3.8.3. Flushing Servers Table	26
3.9. External parsers	26
3.9.1. Supported parser types	26
3.9.2. Setting up parsers	27
3.9.3. Avoid indexer hang on parser execution	27
3.9.4. Pipes in parser's command line	28
3.9.5. Charsets and parsers	28
3.9.6. <code>DPS_URL</code> environment variable	28
3.9.7. Some third-party parsers	28
3.10. Other commands are used in <code>indexer.conf</code>	30
3.10.1. Include command	31
3.10.2. DBAddr command	31
3.10.3. VarDir command	32
3.10.4. NewsExtensions command	32
3.10.5. SyslogFacility command	32
3.10.6. Word length commands	33
3.10.7. MaxDocSize command	33
3.10.8. MinDocSize command	33
3.10.9. IndexDocSizeLimit command	33
3.10.10. URLSelectCacheSize command	33
3.10.11. URLDumpCacheSize command	34
3.10.12. UseCRC32URLId command	34
3.10.13. HTTPHeader command	34
3.10.14. Allow command	34
3.10.15. Disallow command	35
3.10.16. CheckOnly command	35
3.10.17. HrefOnly command	36
3.10.18. CheckMp3 command	36
3.10.19. CheckMp3Only command	36
3.10.20. IndexIf command	36
3.10.21. NoIndexIf command	37
3.10.22. HoldBadHrefs command	37
3.10.23. DeleteOlder command	37
3.10.24. UseRemoteContentType command	38
3.10.25. AddType command	38
3.10.26. Period command	38
3.10.27. PeriodByHops command	39
3.10.28. ExpireAt command	39
3.10.29. UseDateHeader command	39
3.10.30. MaxHops command	39
3.10.31. TrackHops command	39
3.10.32. MaxDepth command	40
3.10.33. MaxDocsPerServer command	40

3.10.34. MaxNetErrors command	40
3.10.35. ReadTimeOut command	40
3.10.36. DocTimeOut command	41
3.10.37. NetErrorDelayTime command	41
3.10.38. Cookies command	41
3.10.39. Robots command	41
3.10.40. RobotsPeriod command	42
3.10.41. CrawlDelay command	42
3.10.42. Section command	42
3.10.43. HrefSection command	43
3.10.44. FastHrefCheck command	43
3.10.45. Index command	43
3.10.46. ProxyAuthBasic command	43
3.10.47. Proxy command	44
3.10.48. AuthBasic command	44
3.10.49. ServerWeight command	44
3.10.50. OptimizeAtUpdate command	45
3.10.51. SkipUnreferred command	45
3.10.52. Bind command	45
3.10.53. ProvideReferer command	45
3.10.54. LongestTextItems command	45
3.11. Extended indexing features	45
3.11.1. Indexing SQL database tables (htdb: virtual URL scheme)	46
3.11.1.1. HTDB indexer.conf commands	46
3.11.1.2. HTDB variables	47
3.11.1.3. Creating full text index	48
3.11.1.4. Indexing SQL database driven web server	49
3.11.2. Indexing binaries output (exec: and cgi: virtual URL schemes)	50
3.11.2.1. Passing parameters to cgi: virtual scheme	50
3.11.2.2. Passing parameters to exec: virtual scheme	51
3.11.2.3. Using exec: virtual scheme as an external retrieval system	51
3.11.3. Mirroring	51
3.11.4. Data acquisition	53
3.12. Using syslog	53
3.13. Storing compressed document copies	54
3.13.1. Configure stored	55
3.13.2. How stored works	56
3.13.3. Using stored during search	56
4. DataparkSearch HTML parser	58
4.1. Tag parser	58
4.2. Special characters	58
4.3. META tags	58
4.4. Links	59
4.5. Comments	59
4.6. Body patterns	60
4.7. Sub-documents	60

5. Storing data	61
5.1. SQL storage types	61
5.1.1. General storage information	61
5.1.2. Various modes of words storage	61
5.1.3. Storage mode - single	61
5.1.4. Storage mode - multi	61
5.1.5. Storage mode - crc	61
5.1.6. Storage mode - crc-multi	62
5.1.7. SQL structure notes	62
5.1.8. Additional features of non-CRC storage modes	62
5.2. Cache mode storage	62
5.2.1. Introduction	62
5.2.2. Cache mode word indexes structure	62
5.2.3. Cache mode tools	63
5.2.4. Starting cache mode	63
5.2.5. Optional usage of several splitters	64
5.2.6. Using run-splitter script	64
5.2.7. Doing search	65
5.2.8. Using search limits	65
5.3. DataparkSearch performance issues	65
5.3.1. searchd usage recommendation	66
5.3.2. Memory based filesystem (mfs) usage recommendation	66
5.3.3. URLInfoSQL command	66
5.3.4. MarkForIndex command	66
5.3.5. CheckInsertSQL command	66
5.3.6. MySQL performance	67
5.3.7. Post-indexing optimization	67
5.3.8. Asynchronous resolver library	68
5.4. SearchD support	68
5.4.1. Why using searchd	68
5.4.2. Starting searchd	68
5.5. Oracle notes	69
5.5.1.	69
5.5.1.1. Why Oracle?	69
5.5.1.2. DataparkSearch+Oracle8 Installation Requirements	70
5.5.1.3. Currently supported/tested platforms	70
5.5.2. Compilation, Installation and Configuration	70
5.5.2.1. Compilation	71
5.5.2.2. Installation and Configuration	71
6. Subsections	74
6.1. Tags	74
6.1.1. Tag command	74
6.1.2. TagIf command	74
6.1.3. Tags in SQL version	74
6.2. Categories	75
6.2.1. Category command	76
6.2.2. CategoryIf command	77

7. Languages support.....	78
7.1. Character sets	78
7.1.1. Supported character sets	78
7.1.2. Character sets aliases	79
7.1.3. Recoding	82
7.1.4. Recoding at search time	??
7.1.5. Document charset detection	82
7.1.6. Automatic charset guesser	82
7.1.6.1. LangMapFile command	??
7.1.6.2. Build your own language maps	??
7.1.7. Default charset	??
7.1.8. Default Language	??
7.1.9. LocalCharset command	??
7.1.10. ForceIISCharset1251 command	??
7.1.11. RemoteCharset command	??
7.1.12. URLCharset command	??
7.1.13. CharsToEscape command	??
7.2. Making multi-language search pages	??
7.2.1. How does it work?	??
7.2.2. Possible troubles	87
7.3. Segmenters for Chinese, Japanese, Korean and Thai languages	??
7.3.1. Japanese language phrase segmenter	??
7.3.2. Chinese language phrase segmenter	??
7.3.3. Thai language phrase segmenter	??
7.3.4. Korean language phrase segmenter	??
7.4. Multilingual servers support	??
8. Searching documents	??
8.1. Using search front-ends	??
8.1.1. Performing search	??
8.1.2. Search parameters	??
8.1.3. Changing different document parts weights at search time	??
8.1.4. Using front-end with an.shtml page	??
8.1.5. Using several templates	??
8.1.6. Search operators	??
8.1.7. Advanced boolean search	??
8.1.8. The Verity Query Language, VQL	??
8.1.9. How search handles expired documents	??
8.2. mod_dpsearch module for Apache httpd	??
8.2.1. Why using mod_dpsearch	??
8.2.2. Configuring mod_dpsearch	??
8.3. How to write search result templates	??
8.3.1. Template sections	98
8.3.1.1. <i>TOP</i> section	??
8.3.1.2. <i>BOTTOM</i> section	102
8.3.1.3. <i>RESTOP</i> section	??
8.3.1.4. <i>RES</i> section	??
8.3.1.5. <i>CLONE</i> section	104

8.3.1.6. <i>RESBOT</i> section.....	105
8.3.1.7. <i>navleft</i> , <i>navleft_nop</i> section	??
8.3.1.8. <i>navbar0</i> section.....	??
8.3.1.9. <i>navright</i> , <i>navright_nop</i> section	??
8.3.1.10. <i>navbar1</i> section.....	??
8.3.1.11. <i>notfound</i> section.....	106
8.3.1.12. <i>noquery</i> section.....	??
8.3.1.13. <i>error</i> section.....	??
8.3.2. Variables section.....	??
8.3.3. Includes in templates	109
8.3.4. Conditional template operators.....	??
8.3.5. Security issues	??
8.4. Designing <i>search.html</i>	??
8.4.1. How the results page is created	??
8.4.2. Your HTML	??
8.4.3. Forms considerations.....	??
8.4.4. Relative links in <i>search.htm</i>	??
8.4.5. Adding Search form to other pages	114
8.5. Relevance	??
8.5.1. Ordering documents	??
8.5.2. Relevance calculation	115
8.5.2.1. A full method of relevance calculation.....	??
8.5.2.2. A fast method of relevance calculation.....	??
8.5.3. Popularity rank	??
8.5.3.1. "Goo" popularity rank calculation method	??
8.5.3.2. "Neo" popularity rank calculation method	??
8.5.4. Boolean search.....	118
8.5.5. Crosswords	??
8.5.6. The Summary Extraction Algorithm (SEA).....	??
8.6. Search queries tracking	??
8.7. Search results cache	??
8.8. Fuzzy search.....	??
8.8.1. Ispell	??
8.8.1.1. Two types of ispell files	120
8.8.1.2. Using Ispell	??
8.8.1.3. Customizing dictionary.....	??
8.8.1.4. Where to get Ispell files	??
8.8.2. Aspell	??
8.8.3. Synonyms	??
8.8.4. Accent insensitive search.....	??
8.8.5. Acronyms and abbreviations	??
9. Miscellaneous	??
9.1. Reporting bugs	??
9.1.1. Core dump reports	??
9.2. Using <i>libdpsearch</i> library.....	??
9.2.1. <i>dps-config</i> script.....	126
9.2.2. <i>DataparkSearch</i> API.....	126

9.3. Database schema	??
A. Donations	??
Index.....	??

List of Tables

3-1. Verbose levels	53
5-1. Cache limit types	65
7-1. Language groups	78
7-2. Charsets aliases.....	79
8-1. Available search parameters	??
8-2. VQL operators supported by DataparkSearch.....	??
8-3. Configure-time parameters to tune relevance calculation (switches for configure)	??
9-1. <code>server</code> table schema	??
9-2. Several server's parameters values in <code>srvinfo</code> table	??

Chapter 1. Introduction

DataparkSearch is a full-featured web search engine. DataparkSearch consists of two parts. The first part is an indexing mechanism (the **indexer**). The indexer walks over hypertext references and stores found words and new references into the database. The second part is a CGI front-end to provide the search service using the data collected by the indexer.

DataparkSearch was cloned from the 3.2.16 CVS version of mnoGoSearch at 27 November 2003 as DataparkSearch 4.16. The mnoGoSearch's first release took place in November 1998. The search engine had the name of UDMSearch until October 2000 when the project was acquired by Lavtech.Com Corp. and changed its name to mnoGoSearch.

The latest change log of DataparkSearch can be found on our website (<http://www.dataparksearch.org/ChangeLog>).

1.1. DataparkSearch Features

Main DataparkSearch features are as follows:

- MySQL (libz library required), PostgreSQL, iODBC, unixODBC, EasySoft ODBC-ODBC bridge, InterBase, Oracle (see), MS SQL back-ends support.
- HTTP support.
- HTTP proxy support.
- HTTPS support.
- FTP support.
- NNTP support (both news:// and nntp:// URL schemes).
- HTDB virtual URL scheme support. One may build index and search through the big text fields/blobs of SQL database.
- Mirroring features.
- text/html, text/xml, text/plain, audio/mpeg (MP3) and image/gif built-in support.
-

External parsers support for other document types.

-
- Ability to index multilingual sites using content negotiation.
-
- Searching all of the word forms using ispell affixes and dictionaries
-
- Basic authorization support. One may index password protected intranet HTTP servers.
-
- Proxy authorization support.
-
- Reentry capability. One may use several indexing and searching processes at the same time even on the same database. Multi-threaded indexing support.
-
- Stop-list support.
-
- <META NAME="robots" content="..."> and robots.txt support.
-
- C language CGI web front-end.
-
- Boolean query language support.
-
- Results sorting by relevancy, popularity rank, last modified date and by importance (a multiplication of relevancy and popularity rank).
-
- Fuzzy search: different word forms, spelling corrections, synonyms, acronyms and abbreviations.
-
- Various character sets support.
-
- HTML templates to easily customize search results.
-
- Advanced search options like time limits, category and tags limits etc.
-
- Phrases segmenting for Chinese, Japanese, Korean and Thai languages.
-
- Accent insensitive search.

- **mod_dpsearch** - search module for Apache (<http://httpd.apache.org/>) web server.
- Internationalized Domain Names support.
- The Summary Extraction Algorithm (SEA).

1.2. Where to get DataparkSearch.

Check for the latest version of DataparkSearch at: <http://www.dataparksearch.org/>, as well at Google Code: <http://code.google.com/p/dataparksearch/>.

A mirror in Moscow, Russia: <http://dataparksearch.newsru.ru/>.

DataparkSearch is also available in FreeBSD ports collection, see www.freshports.org/www/dpsearch (<http://www.freshports.org/www/dpsearch>) and in the T2 Linux SDE (<http://www.t2-project.org/packages/dpsearch.html>).

DataparkSearch's source is available via SVN at Google Code:

```
svn checkout http://dataparksearch.googlecode.com/svn/trunk/ dataparksearch-read-only
```

1.3. Disclaimer

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. See COPYING file for details.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

1.4. Authors

Maxim Zakharov <maxime@maxime.net.ru>, homepage (<http://www.maxime.net.ru/>)

1.4.1. Contributors

Michael Kynast <kynast@newslookup.com>; First DataparkSearch user. Testing on Linux Red Hat.

Jean-Gerard Paillancy: Testing on OpenBSD.

Amit Joshi: Testing on CentOS, packaging for Debian, some ideas to improve the scalability for several PC and using several DBAddr.

mnoGoSearch developers and contributors <devel@mnogosearch.org>: Development and contributions for mnoGoSearch versions up to 3.2.15.

Chapter 2. Installation

2.1. SQL database requirements

Note that if you want to compile DataparkSearch with one of supported SQL database you must have this database already installed before installing DataparkSearch.

It is possible to use DataparkSearch with several SQL databases.

You also should have enough permission to create new database or to write into already existing one.

MySQL notes: If you want to build DataparkSearch with MySQL, 4.1 or later release required. libz library must be installed from **zlib-devel** RPM to successfully compile DataparkSearch with MySQL.

PostgreSQL notes: If you want to build DataparkSearch with PostgreSQL, 7.3.x or later release required. PostgreSQL 8.1 is recommended for better performance.

iODBC notes: iodbc-2.50.22a is known to work.

unixODBC notes: unixODBC-1.7 is known to work.

InterBase notes:

- Interbase 4.0 is known to work.
- InterBase CS 6.0 is known to work.
- FirebirdCS-0.9-4 is known to work.

FreeTDS notes: 0.52 version is known to work with MS SQL 7.0.

Oracle8 notes: 8.0.5.X is known to work.

Oracle8i notes: 8.1.6 R2 EE is known to work.

2.2. Supported operating systems

We use GNU Autoconf so it is possible to compile and use DataparkSearch on almost every modern UNIX system with a C compiler without any modifications. We develop the software on FreeBSD 5.x using PostgreSQL 8.1.

Currently known systems where DataparkSearch has been successfully compiled and tested on are:

- CentOS 3.1, CentOS 3.3
-

Debian GNU/Linux (i386)

•

FreeBSD 2.2.5, 3.x, 4.x, 5.x, 6.x

•

Linux Fedora Core-1, Kernel 2.4.22-1.2174

•

Linux Mandrake 10.2

•

Linux Red Hat 8.0, 9.0

•

Solaris 9

•

Solaris 10 x86, gcc

•

Ubuntu Linux 6.10

•

Gentoo Linux 2007.0 amd64

We hope DataparkSearch will work on other Unix platforms as well. Please report successful platforms to maxime@maxime.net.ru (<mailto:maxime@maxime.net.ru>).

NFS notes: There are some problems reported running DataparkSearch over NFS v4 on Linux 2.6.17. Although, everything is OK on this system when NFS v3 is used.

2.3. Tools required for installation

You need the following tools to build and install DataparkSearch from source:

•

Bzip2 (<http://www.bzip.org/>) to uncompress the distribution.

•

A reasonable tar to unpack the distribution. GNU tar (<http://www.gnu.org/software/tar/tar.html>) is known to work.

•

A working ANSI C compiler. GNU gcc (<http://gcc.gnu.org/>) is known to work.

•

A good make program. GNU make (<http://www.gnu.org/software/make/make.html>) is recommended and sometimes required.

- A `sed` - stream editor.
- A `perl` interpreter, if `install.pl` will be used for installation.
- To build documentation from XML sources, you need `jade` or `openjade` installed.
- You need also `jadetex` installed to build documentation in PDF. Use `make book.pdf` command in `doc/` subdirectory to make that documentation.

2.4. Installing DataparkSearch

Unpack the distribution and change directory into the top-level directory of the unpacked distribution.

```
tar -xyf dpsearch-x.x.tar.bz2
```

To simplify configuration process we included a configuration script with the package - `install.pl`. Run `install.pl` and select DataparkSearch configuration options in a question-and-answer manner. After you specify all the configuration options, the script will run `./configure` with the options you chose. It will also create `install.options` file containing your configuration preferences that you can use to run the script later bypassing questions. After configuration is finished, build and install the package as described in section 3.

In case you would like to configure DataparkSearch manually without using the configuration script, do the following:

If you would like to configure the package with SQL database support:

```
sh$ ./configure --with-mysql
```

or

```
sh$ ./configure --with-pgsql
```

or with another depending on what database you prefer,

or with multiple databases:

```
sh$ ./configure --with-mysql --with-pgsql --with-msql --with-freetds
```

By default, DataparkSearch is installed in `/usr/local/dpsearch` in the following subdirectories:

Directory	Contents
bin	<code>search.cgi</code> , <code>storedoc.cgi</code> , <code>dps-config</code>
lib	<code>libdpsearch.a(so)</code> , <code>libdpcharset.a(so)</code>
sbin	<code>indexer</code> , <code>cached</code> , <code>run-splitter</code> , <code>searchd</code> , <code>splitter</code> , <code>stored</code>

etc	indexer.conf-dist, search.htm-dist, langmap.conf-dist, searchd.conf-dist, stopwords.conf-dist, stored.conf-dist, storedoc.htm-dist
share	various documentation and sql scripts

If you have no permission to write to that directory or just want to install DataparkSearch to another location, please use configure with `--prefix` option, e.g.

```
./configure --prefix=/user/home/data --with-mysql
```

To install DataparkSearch with HTTPS support use configure with the following option:

```
./configure --with-openssl
```

or in case the OpenSSL library is installed in a non-standard location:

```
./configure --with-openssl=/path/to/library
```

Note: Please note that OpenSSL library installed on your system is required for HTTPS support.

You can see all available options with `./configure --help`

If you want to provide some specific flags to C compiler (for example, `'-O7 -mpentium'` to build highly optimized binary for Pentium™ processor if you use egcs/pgcc), you can do so using command

```
sh$ CFLAGS="-O7 -mpentium"
```

before running configure.

To compile DataparkSearch on FreeBSD with Solid in old aout format use

```
sh$ CFLAGS="-aout"
```

before running configure.

To compile DataparkSearch on FreeBSD with aout InterBase use

```
sh$ CFLAGS="-aout -static"
```

before running configure.

You may also specify `--enable-freebsd-pthreads` or `--enable-linux-pthreads` to compile multi-threaded indexer on FreeBSD and Linux machines.

To enable DMALLOC memory debugger (<http://sourceforge.net/projects/dmalloc/>) support use `--enable-dmalloc`.

The euc-kr, big5, gb2312, tscii, gujarati and shift-jis character sets are not supported by default. To build DataparkSearch with these charsets support use configure with `--with-extra-charsets` command line argument.

To build DataparkSearch with all additional charsets support use:

```
./configure --with-extra-charsets=all
```

To build DataparkSearch with only one specified charset support use:

```
./configure --with-extra-charsets=tscii
```

To build DataparkSearch with support for Chinese or Japanese charsets, use:

```
./configure --with-extra-charsets=japanese or ./configure  
--with-extra-charsets=chinese
```

To build DataparkSearch with support for several specified charsets, use a comma separated list of charsets you want:

```
./configure --with-extra-charset=japanese,tscii
```

If you run into problems with configure, please see .

Build and install the package.

```
sh$ make
```

```
sh$ make install
```

If you run into problems with configure, please see .

Create database search (for SQL database only).

You can use existing database, skip this step in this case.

MySQL:

```
sh$ mysqladmin create search
```

PostgreSQL:

```
sh$ createdb search
```

See database specific information if you use another database.

Create sql-tables

Edit `indexer.conf` config file and specify `DBAddr` command according SQL-database used and dbmode selected. Then run:

```
sh$ indexer -Ecreate
```

`indexer` will create all tables automatically.

Installing search scripts

Copy `search.cgi` to your web-server `cgi-bin` directory or make Apache alias to DataparkSearch bin directory.

2.5. Possible installation problems

.

Every time you run `configure`, you must run `make` again to recompile.

To prevent old configuration information or object files from being used, run these commands before re-running `configure`:

```
sh$ rm config.cache
```

```
sh$ make clean
```

•

If your compile fails with make errors, this can be because you are using the wrong version of make. The behavior of Solaris, FreeBSD, OpenBSD make is slightly different from GNU make (<http://www.gnu.org/software/make/make.html>). If you have make-related problems, you should use GNU make (<http://www.gnu.org/software/make/make.html>) instead, often installed as gmake.

GNU make version 3.77 is known to work.

•

If starting Apache with mod_dpsearch module, you're getting the following error: **Undefined symbol "pthread_join"** (or something similar related to pthreads), try to add the following command into httpd.conf file before loading mod_dpsearch.so:

```
LoadFile /usr/lib/libpthread.so
```

If above information doesn't help you, please feel free to contact DataparkSearch mailing list <dataparksearch@yahoogroups.com>.

2.6. Installation registration

If you use DataparkSearch to build search on public accessible web site, you may register this site on our users page (<http://www.dataparksearch.org/usersmap>).

Chapter 3. Indexing

3.1. Indexing in general

3.1.1. Configuration

First, you should configure DataparkSearch. Indexer configuration is covered mostly by `indexer.conf-dist` file. You can find it in `etc` directory of DataparkSearch distribution. You may take a look at other `*.conf` samples in `doc/samples` directory.

To set up `indexer.conf` file, change directory to DataparkSearch installation `/etc` directory, copy `indexer.conf-dist` to `indexer.conf` and edit it.

To configure search front-ends (`search.cgi` and/or `search.php3`, or other), you should copy `search.htm-dist` file in `/etc` directory of DataparkSearch installation to `search.htm` and edit it. See for detailed description.

3.1.2. Running indexer

Just run `indexer` once a week (a day, an hour ...) to find the latest modifications in your web sites. You may also insert `indexer` into your `crontab` job.

By default, `indexer` being called without any command line arguments reindex only expired documents. You can change expiration period with **Period** `indexer.conf` command. If you want to reindex all documents irrelevant if those are expired or not, use `-a` option. `indexer` will mark all documents as expired at startup.

Retrieving documents, `indexer` sends `If-Modified-Since` HTTP header for documents that are already stored in database. When `indexer` gets next document it calculates document's checksum. If checksum is the same with old checksum stored in database, it will not parse document again. `indexer -m` command line option prevents `indexer` from sending `If-Modified-Since` headers and make it parse document even if checksum is the same. It is useful for example when you have changed your **Allow/Disallow** rules in `indexer.conf` and it is required to add new pages that was disallowed earlier.

If DataparkSearch retrieves URL with redirect HTTP 301,302,303 status it will index URL given in `Location:` field of HTTP-header instead.

3.1.3. How to create SQL table structure

To create SQL tables required for DataparkSearch functionality, use `indexer -Ecreate`. Executed with this argument, `indexer` looks up a file containing SQL statements necessary for creating all SQL tables for the database type and storage mode given in **DBAddr** `indexer.conf` command. Files are looking up at `/share` directory of DataparkSearch installation, which is usually `/usr/local/dpsearch/share/`.

3.1.4. How to drop SQL table structure

To drop all SQL tables created by DataparkSearch, use `indexer -Edrop`. A file with SQL statements required to drop tables are looking up at `/share` directory of DataparkSearch installation.

3.1.5. Subsection control

`indexer` has `-t`, `-u`, `-s` options to limit action to only a part of the database. `-t` corresponds 'Tag' limitation, `-u` is a URL substring limitation (SQL LIKE wildcards). `-s` limits URLs with given HTTP status. All limit options in the same group are ORed and in the different groups are ANDed.

3.1.6. How to clear database

To clear the whole database, use '`indexer -C`'. You may also delete only the part of database by using `-t`, `-u`, `-s` subsection control options.

3.1.7. Database Statistics

If you run `indexer -S`, it will show database statistics, including count of total and expired documents of each status. `-t`, `-u`, `-s` filters are usable in this mode too.

The meaning of status is:

-
- 0 - new (not indexed yet) URL

If status is not 0, then it is HTTP response code, some of the HTTP codes are:

-
- 200 - "OK" (url is successfully indexed)
-
- 206 - "Partial OK" (a part of url is successfully indexed)
-
- 301 - "Moved Permanently" (redirect to another URL)
-
- 302 - "Moved Temporarily" (redirect to another URL)
-
- 303 - "See Other" (redirect to another URL)
-
- 304 - "Not modified" (url has not been modified since last indexing)
-

- 401 - "Authorization required" (use login/password for given URL)
-
- 403 - "Forbidden" (you have no access to this URL(s))
-
- 404 - "Not found" (there were references to URLs that do not exist)
-
- 500 - "Internal Server Error" (error in cgi, etc)
-
- 503 - "Service Unavailable" (host is down, connection timed out)
-
- 504 - "Gateway Timeout" (read timeout when retrieving document)

HTTP 401 means that this URL is password protected. You can use **AuthBasic** command in `indexer.conf` to set login:password for this URL(s).

HTTP 404 means that you have incorrect reference in one of your document (reference to resource that does not exist).

Take a look on HTTP specific documentation (<http://www.w3.org/Protocols/>) for further explanation of different HTTP status codes.

Status codes 2xxx are not in HTTP specification and they correspond to the documents marked as clones, where xxx - one of status codes described above.

3.1.8. Link validation

Being started with `-I` command line argument, `indexer` displays URL and it's referrer pairs. It is very useful to find bad links on your site. Don't use **HoldBadHrefs 0** command in `indexer.conf` for this mode. You may use subsection control options `-t,-u,-s` in this mode. For example, `indexer -I -s 404` will display all 'Not found' URLs with referrers where links to those bad documents are found. Setting relevant `indexer.conf` commands and command line options you may use `DataparkSearch` special for site validation purposes.

3.1.9. Parallel indexing

MySQL and PostgreSQL users may run several indexers simultaneously with the same `indexer.conf` file. We have successfully tested 30 simultaneous indexers with MySQL database. Indexer uses MySQL and PostgreSQL locking mechanism to avoid double indexing of the same URL by different indexer. Parallel indexing in the same database is not implemented for other back-ends yet. You may use multi-threaded version of `indexer` with any SQL back-end though which does support several simultaneous connections. Multi-threaded indexer version uses own locking mechanism.

It is not recommended to use the same database with different `indexer.conf` files! First process could add something but second could delete it, and it may never stop.

On the other hand, you may run several indexer processes with different databases with ANY supported SQL back-end.

3.2. Supported HTTP response codes

It is described here the way DataparkSearch processes different HTTP codes. Pseudo-language is used here for explanation.

- - 200 OK
 - If -m command line argument ("force reindex") specified, GOTO 4.
 - Comparing new checksum with old one stored in database
 - If checksum are the same, next_index_time=Now()+Period, GOTO 7
 - Parsing the document, creating word list and adding in "url" table all of the found HREFs
 - Comparing created word list with old one stored in "dict" table
 - Doing UPDATES, DELETES or INSERTs in table "dict" if something is different in word lists.
 - Done
- - 304 Not Modified
 - next_index_time=now()+Period
 - Done
- - 301 Moved Permanently
 - 302 Moved Temporarily
 - 303 See Other
 - Deleting all words in table "dict" for current URL
 - next_index_time=Now()+Period
 - Adding an URL given in Location: header
 - Done
- - 300 Multiple Choices
 - 305 Use Proxy (proxy redirect)
 - 400 Bad Request

401 Unauthorized
 402 Payment Required
 403 Forbidden
 404 Not found
 405 Method Not Allowed
 406 Not Acceptable
 407 Proxy Authentication Required
 408 Request Timeout
 409 Conflict
 410 Gone
 411 Length Required
 412 Precondition Failed
 413 Request Entity Too Large
 414 Request-URI Too Long
 415 Unsupported Media Type
 500 Internal Server Error
 501 Not Implemented
 502 Bad Gateway
 505 Protocol Version Not Supported

Deleting all words in table "dict" for current URL

`next_index_time=Now()+Period`

Done

•

503 Service Unavailable
 504 Gateway Timeout

`next_index_time=Now()+Period`

Done

3.3. Content-Encoding support

DataparkSearch engine supports HTTP compression (Content encoding). Compression can have a major impact on the performance of HTTP transactions. The only way to obtain higher performance is to reduce the number of bytes transmitted.

Using content encoding to receive a server's response you can reduce the traffic by twice or more.

The HTTP 1.1 (RFC 2616) specification (<ftp://ftp.isi.edu/in-notes/rfc2616.txt>) contains four content encoding methods: `gzip`, `deflate`, `compress`, and `identity`.

When Content-encoding is enabled, DataparkSearch's indexer sends to a server `Accept-Encoding: gzip, deflate, compress` string in HTTP headers.

If the server supports any of gzip, deflate or compress encoding, it sends gzipped, deflated or compressed response.

To compile DataparkSearch with HTTP Content encoding support, the `zlib` library is required.

To enable HTTP Content encoding support, configure DataparkSearch with the following option:

```
./configure --with-zlib
```

Use this option along with all the other necessary ones.

3.4. Stopwords

`Stopwords` -- are most frequently used words, i.e. words which appear in almost every document searched. Stopwords are filtered out prior to index construction, what is allow to reduce the total size of the index without any significant loss in quality of search.

3.4.1. StopwordFile command

Load stop words from the given text file. You may specify either absolute file name or a name relative to DataparkSearch `/etc` directory. You may use several **StopwordFile** commands.

```
StopwordFile stopwords/en.sl
```

3.5. Clones

`Clones` -- are documents having equal values of Hash32 on all document sections. Identical copies of the same document always have equal values of Hash32. This allow to eliminate duplicate documents in a collection. However, if only `title` section is defined in `sections.conf`, all documents with different bodies but with identical titles will be considered as clones.

3.5.1. DetectClones command

```
DetectClones yes/no
```

Allow/disallow clone detection and eliminating. If allowed, indexer will detect the same documents under different location, such as mirrors, and will index only one document from the group of such equal documents. "DetectClones yes" also allows to reduce space usage. Default value is "yes".

```
DetectClones no
```

3.6. Specifying WEB space to be indexed

When indexer tries to insert a new URL into database or is trying to index an existing one, it first of all checks whether this URL has corresponding **Server**, **Realm** or **Subnet** command given in `indexer.conf`. URLs without corresponding **Server**, **Realm** or **Subnet** command are not indexed. By default those URLs which are already in database and have no **Server/Realm/Subnet** commands will be deleted from database. It may happen for example after removing some **Server/Realm/Subnet** commands from `indexer.conf`.

These commands have following format:

```
[Server | Realm | Subnet] [method] [subsection] [CaseType] [MatchType] [CmpType] pattern [a
```

Mandatory parameter `pattern` specify an URL, or it part, or pattern to compare.

Optional parameter `method` specify an document action for this command. May take values: `Allow`, `Disallow`, `HrefOnly`, `CheckOnly`, `Skip`, `CheckMP3`, `CheckMP3Only`. By default, the value `Allow` is used.

Allow

Value `Allow` specify that all corresponding documents will be indexed and scanned for new links. Depends on `Content-Type` appropriate external parser is executed if need.

Disallow

Value `Disallow` specify that all corresponding documents will be ignored and deleted from database, if its was placed into before.

HrefOnly

Value `HrefOnly` specify that all corresponding documents will be only scanned for new links (not indexed). This is useful, for example, for mail archives indexing, when index pages is only scanned to detect new messages for indexing.

```
Server HrefOnly Page http://www.mail-archive.com/general%40mnogosearch.org/
Server Allow Path http://www.mail-archive.com/general%40mnogosearch.org/
```

CheckOnly

Value `CheckOnly` specify that all corresponding documents will be requested by HTTP HEAD request, not HTTP GET, i.e. only brief info about documents (size, last modified, content type) will be fetched. This allow, for example, check links on your site:

```
Server HrefOnly http://www.mnogosearch.org/
Realm CheckOnly *
```

These commands instruct **indexer** to scan all documents on `www.mnogosearch.org` site and collect all links. Brief info about every document found will be requested by HEAD method. After indexing done, **indexer -S** command will show status for all documents from this site.

Skip

Value `Skip` specify that all corresponding documents will be skipped while indexing. This is useful when need temporally disable reindexing several sites, but able search on. These documents will marked as expired.

CheckMP3

Value `CheckMP3` specify that corresponding documents will be checked for MP3 tags along if its `Content-Type` is equal to `audio/mpeg`. This is useful, for example, if remote server supply `application/octet-stream` as `Content-Type` for MP3 files. If this tag is present, these files will indexed as MP3 file, otherwise its will be processed according to `Content-Type`.

CheckMP3Only

This value is equal to `CheckMP3`, but if MP3 tag is not present, processing on `Content-Type` will not be taken.

Use optional `subsection` parameter to specify server's checking behavior. Subsection value must be one of the following: `nofollow`, `page`, `path`, `site`, `world` and has "path" value by default.

path subsection

When indexer seeks for a "Server" command corresponding to an URL it checks that the discovered URL starts with URL given in Server command argument but without trailing file name. For example, if `Server path http://localhost/path/to/index.html` is given, all URLs which have `http://localhost/path/to/` at the beginning correspond to this Server command.

The following commands have the same effect except that they insert different URLs into database:

```
Server path http://localhost/path/to/index.html
Server path http://localhost/path/to/index
Server path http://localhost/path/to/index.cgi?q=bla
Server path http://localhost/path/to/index?q=bla
```

site subsection

indexer checks that the discovered URL have the same hostname with URL given in Server command. For example, `Server site http://localhost/path/to/a.html` will allow to index whole `http://localhost/` server.

world subsection

If world subsection is specified in Server command, it has the same effect that URL is considered to match this Server command. See explanation below.

page subsection

This subsection describes the only one URL given in Server argument.

nofollow subsection

Skip links following for URL that match the pattern.

subsection in news:// schema

Subsection is always considered as "site" for `news://` URL schema. This is because `news://` schema has no nested paths like `ftp://` or `http://` Use `Server news://news.server.com/` to index whole news server or for example `Server news://news.server.com/udm` to index all messages from "udm" hierarchy.

Optional parameter `CaseType` is specify the case sensivity for string comparison, it can take one of follow value: `case` - case insensitive comparison, or `nocase` - case sensitive comparison.

Optional parameter `CmpType` is specify the type of comparison and can take two value: `Regex` and `String`. `String` wildcards is default match type. You can use `?` and `*` signs in `URLMask` parameters, they means "one character" and "any number of characters" respectively. For example, if you want to index all HTTP sites in `.ru` domain, use this command:

```
Realm http://*.ru/*
```

`Regex` comparison type takes a regular expression as it's argument. Activate `regex` comparison type using `Regex` keyword. For example, you can describe everything in `.ru` domain using `regex` comparison type:

```
Realm Regex ^http://.*\.ru/
```

Optional parameter `MatchType` means match type. There are `Match` and `NoMatch` possible values with `Match` as default. `Realm NoMatch` has reverse effect. It means that URL that does not match given pattern will correspond to this **Realm** command. For example, use this command to index everything without `.com` domain:

```
Realm NoMatch http://*.com/*
```

Optional `alias` argument allows providing very complicated URL rewrite more powerful than other aliasing mechanism. Take a look for `alias` argument usage explanation. `Alias` works only with `Regex` comparison type and has no effect with `String` type.

3.6.1. Server command

This is the main command of the `indexer.conf` file. It is used to add servers or their parts to be indexed. This command also says `indexer` to insert given URL into database at startup.

E.g. command `Server http://localhost/` allows to index whole `http://localhost/` server. It also makes `indexer` insert given URL into database at startup. You can also specify some path to index server subsection: `Server http://localhost/subsection/`. It also says `indexer` to insert given URL at startup.

Note: You can suppress `indexer` behavior to add URL given in `Server` command by using `-q` `indexer` command line argument. It is useful when you have hundreds or thousands `Server` commands and their URLs are already in database. This allows having more quick `indexer` startup.

3.6.2. Realm command

Realm command is a more powerful means of describing web area to be indexed. It works almost like **Server** command but takes a regular expression or string wildcards as it's pattern parameter and do not insert any URL into database for indexing.

3.6.3. Subnet command

Subnet command is another way to describe web area to be indexed. It works almost like **Server** command but takes a string wildcards or network specified in CIDR presentation format as it's **pattern** argument which is compared against IP address instead of URL. In case of string wildcards format, argument may have * and ? signs, they means "one character" and "any number of characters" respectively. For example, if you want to index all HTTP sites in your local subnet, use this command:

```
Subnet 192.168.*.*
```

In case of network specified in CIDR presentation format, you may specify subnet in forms: a.b.c.d/m, a.b.c, a.b, a

```
Subnet 1291.168.10.0/24
```

You may use "NoMatch" optional argument. For example, if you want to index everything without 195.x.x.x subnet, use:

```
Subnet NoMatch 195.*.*.*
```

3.6.4. Using different parameter for server and it's subsections

Indexer seeks for "Server" and "Realm" commands in order of their appearance. Thus if you want to give different parameters to e.g. whole server and its subsection you should add subsection line before whole server's. Imagine that you have server subdirectory which contains news articles. Surely those articles are to be reindexed more often than the rest of the server. The following combination may be useful in such cases:

```
# Add subsection
Period 200000
Server http://servername/news/

# Add server
Period 600000
Server http://servername/
```

These commands give different reindexing period for `/news/` subdirectory comparing with the period of server as a whole. indexer will choose the first "Server" record for the `http://servername/news/page1.html` as far as it matches and was given first.

3.6.5. Default indexer behavior

The default behavior of indexer is to follow through links having correspondent Server/Realm command in the `indexer.conf` file. It also jumps between servers if both of them are present in `indexer.conf` either directly in Server command or indirectly in Realm command. For example, there are two Server commands:

```
Server http://www/
Server http://web/
```

When indexing `http://www/page1.html` indexer WILL follow the link `http://web/page2.html` if the last one has been found. Note that these pages are on different servers, but BOTH of them have correspondent Server record.

If one of the Server command is deleted, indexer will remove all expired URLs from this server during next reindexing.

3.6.6. Using `indexer -f <filename>`

The third scheme is very useful for `indexer -i -f url.txt` running. You may maintain required servers in the `url.txt`. When new URL is added into `url.txt` indexer will index the server of this URL during next startup.

3.6.7. ServerDB, RealmDB, SubnetDB and URLDB commands

```
URLDB pgsql://foo:bar@localhost/portal/links?field=url
```

These commands are equal to **Server**, **Realm**, **Subnet** and **URL** commands respectively, but takes arguments from field of SQL-table specified. In example above, URLs are takes from database `portal`, SQL-table `links` and filed `url`.

3.6.8. URL command

```
URL http://localhost/path/to/page.html
```

This command inserts given URL into database. This is usefull to add several entry points to one server. Has no effect if an URL is already in the database.

3.7. Aliases

DataparkSearch has an alias support making it possible to index sites taking information from another location. For example, if you index local web server, it is possible to take pages directly from disk

without involving your web server in indexing process. Another example is building of search engine for primary site and using its mirror while indexing. There are several ways of using aliases.

3.7.1. Alias `indexer.conf` command

Format of "Alias" `indexer.conf` command:

```
Alias <masterURL> <mirrorURL>
```

E.g. you wish to index `http://search.mnogo.ru/` using nearest German mirror `http://www.gstammw.de/mirrors/mnoGoSearch/`. Add these lines in your `indexer.conf`:

```
Server http://search.mnogo.ru/
Alias http://search.mnogo.ru/ http://www.gstammw.de/mirrors/mnoGoSearch/
```

`search.cgi` will display URLs from master site `http://search.mnogo.ru/` but indexer will take corresponding page from mirror site `http://www.gstammw.de/mirrors/mnoGoSearch/`.

Another example. If you want to index everything in `udm.net` domain and one of servers, for example `http://home.udm.net/` is stored on local machine in `/home/httpd/htdocs/` directory. These commands will be useful:

```
Realm http://*.udm.net/
Alias http://home.udm.net/ file:/home/httpd/htdocs/
```

Indexer will take `home.udm.net` from local disk and index other sites using HTTP.

3.7.2. Different aliases for server parts

Aliases are searched in the order of their appearance in `indexer.conf`. So, you can create different aliases for server and its parts:

```
# First, create alias for example for /stat/ directory which
# is not under common location:
Alias http://home.udm.net/stat/ file:/usr/local/stat/htdocs/

# Then create alias for the rest of the server:
Alias http://home.udm.net/ file:/usr/local/apache/htdocs/
```

Note: if you change the order of these commands, alias for `/stat/` directory will never be found.

3.7.3. Using aliases in Server commands

You may specify location used by indexer as an optional argument for Server command:

```
Server http://home.udm.net/ file:/home/httpd/htdocs/
```

3.7.4. Using aliases in Realm commands

Aliases in Realm command is a very powerful feature based on regular expressions. The idea of aliases in Realm command implementation is similar to how PHP `preg_replace()` function works. Aliases in Realm command work only if "regex" match type is used and does not work with "string" match type.

Use this syntax for Realm aliases:

```
Realm regex <URL_pattern> <alias_pattern>
```

Indexer searches URL for matches to `URL_pattern` and builds an URL alias using `alias_pattern`. `alias_pattern` may contain references of the form `$n`. Where `n` is a number in the range of 0-9. Every such reference will be replaced by text captured by the `n`'th parenthesized pattern. `$0` refers to text matched by the whole pattern. Opening parentheses are counted from left to right (starting from 1) to obtain the number of the capturing subpattern.

Example: your company hosts several hundreds users with their domains in the form of `www.username.yourname.com`. Every user's site is stored on disk in "htdocs" under user's home directory: `/home/username/htdocs/`.

You may write this command into `indexer.conf` (note that dot `.` character has a special meaning in regular expressions and must be escaped with `\` sign when dot is used in usual meaning):

```
Realm regex (http://www\.) (.*) (\.yourname\.com/) (.*) file:/home/$2/htdocs/$4
```

Imagine indexer process `http://www.john.yourname.com/news/index.html` page. It will build patterns from `$0` to `$4`:

```
$0 = 'http://www.john.yourname.com/news/index.htm' (whole patter match)
$1 = 'http://www.' subpattern matches '(http://www\.)'
$2 = 'john' subpattern matches '(.*)'
$3 = '.yourname.com/' subpattern matches '(\.yourname\.com/)'
$4 = '/news/index.html' subpattern matches '(.*)'
```

Then indexer will compose alias using `$2` and `$4` patterns:

```
file:/home/john/htdocs/news/index.html
```

and will use the result as document location to fetch it.

3.7.5. AliasProg command

You may also specify "AliasProg" command for aliasing purposes. AliasProg is useful for major web hosting companies which want to index their web space taking documents directly from a disk without having to involve web server in indexing process. Documents layout may be very complex to describe it using alias in Realm command. AliasProg is an external program that can be called, that takes a URL and returns one string with the appropriate alias to stdout. Use \$1 to pass URL to command line.

For example this AliasProg command uses 'replace' command from MySQL distribution and replaces URL substring `http://www.apache.org/` to `file:/usr/local/apache/htdocs/`:

```
AliasProg "echo $1 | /usr/local/mysql/bin/mysql/replace http://www.apache.org/ file:/usr/l
```

You may also write your own very complex program to process URLs.

3.7.6. ReverseAlias command

The ReverseAlias `indexer.conf` command allows URL mapping before URL is inserted into database. Unlike Alias command, that triggers mapping right before a document is downloaded, ReverseAlias command triggers mapping after the link is found.

```
ReverseAlias http://name2/    http://name2.yourname.com/
Server          http://name2.yourname.com/
```

All links with short server name will be mapped to links with full server name before they are inserted into database.

One of the possible use is cutting various unnecessary strings like `PHPSESSION=XXXX`

E.g. cutting from URL like `http://www/a.php?PHPSESSION=XXX`, when `PHPSESSION` is the only parameter. The question sign is deleted as well:

```
ReverseAlias regex (http://[^\?]*)[\?]{0,1}PHPSESSION=[^\&]*$          $1$2
```

Cutting from URL like `w/a.php?PHPSESSION=xxx&...`, i.e. when `PHPSESSION` is the first parameter, but there are other parameters following it. The '&' sign after `PHPSESSION` is deleted as well. Question mark is not deleted:

```
ReverseAlias regex (http://[^\?]*[\?]{0,1})PHPSESSION=[^\&]*&(.*)      $1$2
```

Cutting from URL like `http://www/a.php?a=b&PHPSESSION=xxx` or `http://www/a.php?a=b&PHPSESSION=xxx&c=d`, where `PHPSESSION` is not the first parameter. The '&' sign before `PHPSESSION` is deleted:

```
ReverseAlias regex (http://.*)&PHPSESSION=[^\&]*(.*)                  $1$2
```

3.7.7. Alias command in search.htm search template

It is also possible to define aliases in search template (`search.htm`). The Alias command in `search.htm` is identical to the one in `indexer.conf`, however it is active during searching, not indexing.

The syntax of the `search.htm` Alias command is the same as in `indexer.conf`:

```
Alias <find-prefix> <replace-prefix>
```

For example, there is the following command in `search.htm`:

```
Alias http://localhost/ http://www.mnogo.ru/
```

Search returned a page with the following URL:

```
http://localhost/news/article10.html
```

As a result, the `$(DU)` variable will be replace NOT with this URL:

```
http://localhost/news/article10.html
```

but with the following URL (that results in processing with Alias):

```
http://www.mnogo.ru/news/article10.html
```

3.8. Servers Table

DataparkSearch has **ServerTable** `indexer.conf` command. It allow load servers and filters configuration from SQL table.

3.8.1. Loading servers table

When **ServerTable** `mysql://user:pass@host/dbname/tablename[?srvinfo=infotablename]` is specified, indexer will load servers information from given `tablename` SQL table, and will load servers parameters from given `infotablename` SQL table. If `srvinfo` parameter is not specified, parameters will be loaded from `srvinfo` table. Check the structure for `server` and `srvinfo` tables in

`create/mysql/create.txt` file. If there is no structure example for your database, take it as an example.

You may use several **ServerTable** command to load servers information from different tables.

3.8.2. Servers table structure

Servers table consists of all necessary fields which describe servers parameters. Field names have correspondent `indexer.conf` commands. For example, "period" field corresponds "Period" `indexer.conf` command. Default field values are the same with default `indexer.conf` parameters.

"gindex" field corresponds "Index" command. Name is slightly changed to avoid SQL reserved word usage.

Description for several fields see in .

Note: Only those servers are read from the table where "active" field has 1 value and "parent" field has 0 value. This is useful to allow users to submit new URLs into servers table and give administrator a possibility to approve added URLs.

3.8.3. Flushing Servers Table

Flush `server.enabled` to inactive for all server table records. Use this command to deactivate all command in `servertable` before load new from `indexer.conf` or from other `servertable`.

3.9. External parsers

DataparkSearch indexer can use external parsers to index various file types (mime types).

Parser is an executable program which converts one of the mime types to `text/plain` or `text/html`. For example, if you have postscript files, you can use `ps2ascii` parser (filter), which reads postscript file from `stdin` and produces `ascii` to `stdout`.

3.9.1. Supported parser types

Indexer supports four types of parsers that can:

- read data from `stdin` and send result to `stdout`
- read data from file and send result to `stdout`
-

read data from file and send result to file

•

read data from stdin and send result to file

3.9.2. Setting up parsers

Configure mime types

Configure your web server to send appropriate "Content-Type" header. For apache, have a look at `mime.types` file, most mime types are already defined there.

If you want to index local files or via ftp use "AddType" command in `indexer.conf` to associate file name extensions with their mime types. For example:

```
AddType text/html *.html
```

Add parsers

Add lines with parsers definitions. Lines have the following format with three arguments:

```
Mime <from_mime> <to_mime> <command line>
```

For example, the following line defines parser for man pages:

```
# Use deroff for parsing man pages ( *.man )
Mime application/x-troff-man text/plain deroff
```

This parser will take data from stdin and output result to stdout.

Many parsers can not operate on stdin and require a file to read from. In this case `indexer` creates a temporary file in `/tmp` and will remove it when parser exists. Use `$1` macro in parser command line to substitute file name. For example, Mime command for "catdoc" MS Word to ASCII converters may look like this:

```
Mime application/msword text/plain "/usr/bin/catdoc -a $1"
```

If your parser writes result into output file, use `$2` macro. `indexer` will replace `$2` by temporary file name, start parser, read result from this temporary file then remove it. For example:

```
Mime application/msword text/plain "/usr/bin/catdoc -a $1 >$2"
```

The parser above will read data from first temporary file and write result to second one. Both temporary files will be removed when parser exists. Note that result of usage of this parser will be absolutely the same with the previous one, but they use different execution mode: `file->stdout` and `file->file` correspondingly.

3.9.3. Avoid indexer hang on parser execution

To avoid a `indexer` hang on parser execution, you may specify the amount of time in seconds for parser execution in your `indexer.conf` by `ParserTimeOut` command. For example:

```
ParserTimeOut 600
```

Default value is 300 seconds, i.e. 5 minutes.

3.9.4. Pipes in parser's command line

You can use pipes in parser's command line. For example, these lines will be useful to index gzipped man pages from local disk:

```
AddType  application/x-gzipped-man *.1.gz *.2.gz *.3.gz *.4.gz
Mime      application/x-gzipped-man text/plain "zcat | deroff"
```

3.9.5. Charsets and parsers

Some parsers can produce output in other charset than given in LocalCharset command. Specify charset to make indexer convert parser's output to proper one. For example, if your catdoc is configured to produce output in windows-1251 charset but LocalCharset is koi8-r, use this command for parsing MS Word documents:

```
Mime  application/msword "text/plain; charset=windows-1251" "catdoc -a $1"
```

3.9.6. DPS_URL environment variable

When executing a parser **indexer** creates DPS_URL environment variable with an URL being processed as a value. You can use this variable in parser scripts.

3.9.7. Some third-party parsers

.

RPM parser by Mario Lang <lang@zid.tu-graz.ac.at>

/usr/local/bin/rpminfo:

```
#!/bin/bash
/usr/bin/rpm -q --queryformat="<html><head><title>RPM: %{NAME} %{VERSION}-%{RELEASE}
{%{GROUP}}</title><meta name=\"description\" content=\"{%{SUMMARY}}\"></head><body>
{%{DESCRIPTION}}\n</body></html>" -p $1
```

indexer.conf:

```
Mime application/x-rpm text/html "/usr/local/bin/rpminfo $1"
```

It renders to such nice RPM information:

```
3. RPM: mysql 3.20.32a-3 (Applications/Databases) [4]
    Mysql is a SQL (Structured Query Language) database server.
```

Mysql was written by Michael (monty) Widenius. See the CREDITS file in the distribution for more credits for mysql and related things....

(application/x-rpm) 2088855 bytes

•

catdoc MS Word to text converter

Home page (http://freshmeat.net/redirect/catdoc/1055/url_homepage/), also listed on Freshmeat (<http://freshmeat.net/>).

indexer.conf:

```
Mime application/msword      text/plain      "catdoc $1"
```

•

xls2csv MS Excel to text converter

It is supplied with catdoc.

indexer.conf:

```
Mime application/vnd.ms-excel text/plain      "xls2csv $1"
```

•

pdftotext Adobe PDF converter

Supplied with xpdf project.

Homepage (http://freshmeat.net/redirect/xpdf/12080/url_homepage/), also listed on Freshmeat (<http://freshmeat.net/>).

indexer.conf:

```
Mime application/pdf        text/plain      "pdftotext $1 -"
```

•

unrtf RTF to html converter

Homepage (<ftp://ftp.gnu.org/pub/gnu/unrtf/>)

indexer.conf:

```
Mime text/rtf*              text/html      "/usr/local/dpsearch/sbin/unrtf --html $1"
Mime application/rtf        text/html      "/usr/local/dpsearch/sbin/unrtf --html $1"
```

•

xlhtml XLS to html converter

Homepage (<http://chicago.sourceforge.net/xlhtml/>)

indexer.conf:

```
Mime application/vnd.ms-excel text/html      "/usr/local/dpsearch/sbin/xlhtml $1"
```

•

ppthtml PowerPoint (PPT) to html converter. Part of xlhtml 0.5.

Homepage (<http://chicago.sourceforge.net/xlhtml/>)

`indexer.conf:`

```
Mime application/vnd.ms-powerpoint text/html "/usr/local/dpsearch/sbin/ppthtml $1"
```

•

Using **vwHtml** (<http://wvWare.sourceforge.net/>) (DOC to html).

`/usr/local/dpsearch/sbin/0vwHtml.pl:`

```
#!/usr/bin/perl -w
```

```
$p = $ARGV[1];
```

```
$f = $ARGV[1];
```

```
$p =~ s/(.*)\[([^\[]*)/$1\//;
```

```
$f =~ s/(.*)\[([^\[]*)/$2/;
```

```
system("/usr/local/bin/vwHtml --targetdir=$p $ARGV[0] $f");
```

`indexer.conf:`

```
Mime application/msword text/html "/usr/local/dpsearch/sbin/0vwHtml.pl $1 $2"
```

```
Mime application/vnd.ms-word text/html "/usr/local/dpsearch/sbin/0vwHtml.pl $1 $2"
```

•

swf2html from Flash Search Engine SDK

(http://www.macromedia.com/software/flash/download/search_engine/)

`indexer.conf:`

```
Mime application/x-shockwave-flash text/html "/usr/local/dpsearch/sbin/swf2html $1"
```

•

djvutxt from djvuLibre (<http://djvu.sourceforge.net/>)

`indexer.conf:`

```
Mime image/djvu text/plain "/usr/local/bin/djvutxt $1 $2"
```

```
Mime image/x.djvu text/plain "/usr/local/bin/djvutxt $1 $2"
```

```
Mime image/x-djvu text/plain "/usr/local/bin/djvutxt $1 $2"
```

```
Mime image/vnd.djvu text/plain "/usr/local/bin/djvutxt $1 $2"
```

3.10. Other commands are used in `indexer.conf`

3.10.1. Include command

You may include another configuration file in any place of the `indexer.conf` using **Include** `<filename>` command. Absolute path if `<filename>` starts with `"/"`:

```
Include /usr/local/dpsearch/etc/incl.conf
```

Relative path else:

```
Include incl.conf
```

3.10.2. DBAddr command

DBAddr command is URL-style database description. It specify options (type, host, database name, port, user and password) to connect to SQL database. Should be used before any other commands. You may specify several **DBAddr** commands. In this case DataparkSearch will merge result from every database specified. Command have global effect for whole config file. Format:

```
DBAddr <Type>:[//[User[:Pass]@]Host[:Port]]/DBName/[?[dbmode=mode] {&<parameter name>=<param
```

Note: ODBC related. Use `DBName` to specify ODBC data source name (DSN) `Host` does not matter, use "localhost".

Note: Solid related. Use `Host` to specify Solid server `DBName` does not matter for Solid.

You may use CGI-like encoding for `User` and `Pass` if you need use special characters in user name or password. For example, if you have `ABC@DEF` as password, you should write it as `ABC%40DEF`.

Currently supported `Type` values are `mysql`, `pgsql`, `mssql`, `solid`, `mssql`, `oracle`, `ibase`, `sqlite`. Actually, it does not matter for native libraries support. But ODBC users should specify one of supported values. If your database type is not supported, you may use "unknown" instead.

MySQL and PostgreSQL users can specify path to Unix socket when connecting to localhost:

```
mysql://foo:bar@localhost/dpsearch/?socket=/tmp/mysql.sock
```

If you are using PostgreSQL and do not specify hostname, e.g. `pgsql://user:password@/dbname/` then PostgreSQL will not work via TCP, but will use default Unix socket.

`dbmode` parameter. You may also select database mode of words storage. When "single" is specified, all words are stored in the same table (file). If "multi" is selected, words will be located in

different tables (files) depending of their lengths. "multi" mode is usually faster but requires more tables (files). If "crc" mode is selected, DataparkSearch will store 32 bit integer word IDs calculated by HASH32 algorithm instead of words. This mode requires less disk space and it is faster comparing with "single" and "multi" modes, however it doesn't support substring searches. "crc-multi" uses the same storage structure with the "crc" mode, but also stores words in different tables (files) depending on words lengths like "multi" mode. Default mode is "single".

stored parameter. Format: **stored=StoredHost[:StoredPort]**. This parameter is used to specify host and port, if specified, where **stored** daemon is running, if you plan to use document excerpts and cached copies.

cached parameter. Format: **cached=CachedHost[:CachedPort]**. Use **cached** at given host and port if specified. It is required for cache storage mode only (see). Each **indexer** will connect to **cached** on given address at startup.

charset parameter. Format: **charset=DBCharacterSet**. This parameter can be used to specify database connection charset. The charset specified by DBCharacterSet should be equal to charset specified by **LocalCharset** command.

label parameter. Format: **label=DBAlabel**. This parameter may be used to assign a label to **DBAddr** command. So, if you pass **label** CGI-variable to the DataparkSearch, then only **DBAddr** marked by label value will be used to performing search. Thus, you can use one **searchd** daemon to answer queries for several search databases selectable by **label** variable.

Note: If no **label** is passed as CGI-parameter, then only **DBAddr** without a **label** will be used to perform search query.

Example:

```
DBAddr mysql://foo:bar@localhost/dpsearch/?dbmode=single
```

3.10.3. VarDir command

You may choose alternative working directory for cache mode:

```
VarDir /usr/local/dpsearch/var
```

3.10.4. NewsExtensions command

Whether to enable news extensions. Default value is no.

```
NewsExtensions yes
```

3.10.5. SyslogFacility command

This is used if DataparkSearch was compiled with syslog support and if you don't like the default value. Argument is the same as used in `syslog.conf` file. For list of possible facilities see `syslog.conf(5)`

```
SyslogFacility local7
```

3.10.6. Word length commands

Word lengths. You may change default length range of words stored in database. By default, words with the length in the range from 1 to 32 are stored.

```
MinWordLength 1
MaxWordLength 32
```

3.10.7. MaxDocSize command

This command is used for specify maximal document size. Default value 1048576 (1 Megabyte). Takes global effect for whole config file.

```
MaxDocSize 1048576
```

3.10.8. MinDocSize command

This command is used to checkonly urls with content size less than value specified. Default value 0. Takes global effect for whole config file.

```
MinDocSize 1024
```

3.10.9. IndexDocSizeLimit command

Use this command to specify the maximal amount of data stored in index per document. Default value 0. This mean no limit. Takes effect till next **IndexDocSizeLimit** command.

```
IndexDocSizeLimit 65536
```

3.10.10. URLSelectCacheSize command

Select number of targets to index at once. Default value is 1024.

```
URLSelectCacheSize 10240
```

3.10.11. URLDumpCacheSize command

Select at once this number of urls to write cache mode indexes, to preload url data or to calculate the Popularity Rank. Default value is 100000.

```
URLDumpCacheSize 10240
```

3.10.12. UseCRC32URLId command

Switch on or off the ID generation for URL using HASH32. Default value is "no".

```
UseCRC32URLId yes
```

Switching it on allow speed up indexing a bit, but some small number of collisions is possible.

3.10.13. HTTPHeader command

You may add desired headers to indexer's HTTP request. You should not use "If-Modified-Since", "Accept-Charset" headers, these headers are composed by indexer itself. "User-Agent: DataparkSearch/version" header is sent too, but you may override it. Command has global effect for all configuration file.

```
HTTPHeader "User-Agent: My_Own_Agent"
HTTPHeader "Accept-Language: ru, en"
HTTPHeader "From: webmaster@mysite.com"
```

3.10.14. Allow command

```
Allow [Match|NoMatch] [NoCase|Case] [String|Regex] <arg> [<arg> ... ]
```

Use this command to allow URLs that match (doesn't match) given argument. First three optional parameters describe the type of comparison. Default values are Match, NoCase, String. Use NoCase or Case values to choose case sensitive or case insensitive comparison. Use Regex to choose regular

expression comparison. Use `String` to choose string with wildcards comparison. Wildcards are `'*'` for any number of characters and `'?'` for one character. Note that `'?'` and `'*'` have special meaning in `String` match type. Please use `Regex` to describe documents with `'?'` and `'*'` signs in URL. `String` match is much faster than `Regex`. Use `String` where it is possible. You may use several arguments for one **Allow** command. You may use this command any times. Takes global effect for config file. Note that `DataparkSearch` automatically adds one `"Allow regex .*"` command after reading config file. It means that allowed everything that is not disallowed.

Examples

```
# Allow everything:
Allow *
# Allow everything but .php .cgi .pl extensions case insensitively using regex:
Allow NoMatch Regex /\.php$/\.cgi$/\.pl$/
# Allow .HTM extension case sensitively:
Allow NoCase *.HTM
```

3.10.15. Disallow command

```
Disallow [Match|NoMatch] [NoCase|Case] [String|Regex] <arg> [<arg> ... ]
```

Use this command to disallow URLs that match (doesn't match) given argument. The meaning of first three optional parameters is exactly the same with **Allow** command. You can use several arguments for one **Disallow** command. Takes global effect for config file. Examples:

```
# Disallow URLs that are not in udm.net domains using "string" match:
Disallow NoMatch *.udm.net/*
# Disallow any except known extensions and directory index using "regex" match:
Disallow NoMatch Regex \/$|\.htm$|\.html$|\.shhtml$|\.phtml$|\.php$|\.txt$
# Exclude cgi-bin and non-parsed-headers using "string" match:
Disallow */cgi-bin/* *.cgi */nph-*
# Exclude anything with '?' sign in URL. Note that '?' sign has a
# special meaning in "string" match, so we have to use "regex" match here:
Disallow Regex \?
```

3.10.16. CheckOnly command

```
CheckOnly [Match|NoMatch] [NoCase|Case] [String|Regex] <arg> [<arg> ... ]
```

The meaning of first three optional parameters is exactly the same with **Allow** command. Indexer will use HEAD instead of GET HTTP method for URLs that match/do not match given regular expressions. It means that the file will be checked only for being existing and will not be downloaded. Useful for zip,exe,arj and other binary files. Note that you can disallow those files with commands given below. You may use several arguments for one **CheckOnly** commands. Useful for example for searching through the URL names rather than the contents (a la FTP-search). Takes global effect for config file. Examples:

```
# Check some known non-text extensions using "string" match:
CheckOnly *.b *.sh *.md5
# or check ANY except known text extensions using "regex" match:
CheckOnly NoMatch Regex \/$|\.html$|\.shtml$|\.phtml$|\.php$|\.txt$
```

3.10.17. HrefOnly command

```
HrefOnly [Match|NoMatch] [NoCase|Case] [String|Regex] <arg> [<arg> ... ]
```

The meaning of first three optional parameters is exactly the same with **Allow** command. Use this to scan a HTML page for "href" attribute of tags but not to index the contents of the page with an URLs that match (doesn't match) given argument. Commands have global effect for all configuration file. When indexing large mail list archives for example, the index and thread index pages (like mail.10.html, thread.21.html, etc.) should be scanned for links but shouldn't be indexed:

```
HrefOnly */mail*.html */thread*.html
```

3.10.18. CheckMp3 command

```
CheckMp3 [Match|NoMatch] [NoCase|Case] [String|Regex] <arg> [<arg> ...]
```

The meaning of first three optional parameters is exactly the same with **Allow** command. If an URL matches given rules, indexer will download only a little part of the document and try to find MP3 tags in it. On success, indexer will parse MP3 tags, else it will download whole document then parse it as usual. Notes: This works only with those servers which support HTTP/1.1 protocol. It is used "Range: bytes" header to download mp3 tag.

```
CheckMp3 *.bin *.mp3
```

3.10.19. CheckMp3Only command

```
CheckMP3Only [Match|NoMatch] [NoCase|Case] [String|Regex] <arg> [<arg> ...]
```

The meaning of first three optional parameters is exactly the same with **Allow** command. If an URL matches given rules, indexer, like in the case **CheckMP3** command, will download only a little part of the document and try to find MP3 tags. On success, indexer will parse MP3 tags, else it will NOT download whole document.

```
CheckMP3Only *.bin *.mp3
```

3.10.20. IndexIf command

```
IndexIf [Match|NoMatch] [NoCase|Case] [String|Regex] <section> <arg> [<arg> ... ]
```

Use this command to allow indexing, if the value of `section` match the `arg` pattern given. The meaning of first three optional parameters is exactly the same as for the **Allow** command (see).

Example

```
IndexIf regex Title Manual
IndexIf body "*important detail*"
```

3.10.21. NoIndexIf command

```
NoIndexIf [Match|NoMatch] [NoCase|Case] [String|Regex] <section> <arg> [<arg> ... ]
```

Use this command to disallow indexing, if the value of `section` match the `arg` pattern given. The meaning of first three optional parameters is exactly the same as for the **Allow** command (see).

Example

```
NoIndexIf regex Title Sex
IndexIf body *xxx*
```

3.10.22. HoldBadHrefs command

```
HoldBadHrefs <time>
```

How much time to hold URLs with erroneous status before deleting them from the database. For example, if host is down, indexer will not delete pages from this site immediately and search will use previous content of these pages. However if site doesn't respond for a month, probably it's time to remove these pages from the database. For `<time>` format see description of **Period** command in .

```
HoldBadHrefs 30d
```

3.10.23. DeleteOlder command

```
DeleteOlder <time>
```

How much time to hold URLs before deleting them from the database. For example, for news sites indexing, you may delete automatically old news articles after specified period. For `<time>` format see description of **Period** command in . Default value is 0. "0" value mean "do not check". You may specify several **DeleteOlder** commands, for example, by one for every **Server** command.

```
DeleteOlder 7d
```

3.10.24. UseRemoteContentType command

```
UseRemoteContentType yes/no
```

This command specifies if the indexer should get content type from http server headers (yes) or from it's AddType settings (no). If set to 'no' and the indexer could not determine content-type by using its AddType settings, then it will use http header. Default: yes

```
UseRemoteContentType yes
```

3.10.25. AddType command

```
AddType [String|Regex] [Case|NoCase] <mime type> <arg> [<arg>...]
```

This command associates filename extensions (for services that don't automatically include them) with their mime types. Currently "file:" protocol uses these commands. Use optional first two parameter to choose comparison type. Default type is "String" "Case" (case insensitive string match with '?' and '*' wildcards for one and several characters correspondently).

```
AddType image/x-xpixmap *.xpm
```

3.10.26. Period command

```
Period <time>
```

Set reindex period. <time> is in the form 'xxxA[yyyB[zzzC]]' (Spaces are allowed between xxx and A and yyy and so on) there xxx, yyy, zzz are numbers (can be negative!) A, B, C can be one of the following: s - second M - minute h - hour d - day m - month y - year (these letters are the same as in strptime/strftime functions). Examples:

```
15s - 15 seconds
4h30M - 4 hours and 30 minutes
1y6m-15d - 1 year and six month minus 15 days
1h-10M+1s - 1 hour minus 10 minutes plus 1 second
```

If you specify only number without any character, it is assumed that time is given in seconds. Can be set many times before **Server** command and takes effect till the end of config file or till next **Period** command.

Period 7d

3.10.27. PeriodByHops command

PeriodByHops <hops> [<time>]

Set reindex period per <hops> basis. The format for <time> is the same as for **Period**.

Can be set many times before **Server** command and takes effect till the end of config file or till next **PeriodByHops** command with same <hops> value. If <time> parameter is omitted, this undefine the previous defined value.

If for given <hops> value the appropriate **PeriodByHops** command is not specified, in this case the value defined in **Period** command is used.

3.10.28. ExpireAt command

ExpireAt [A [B [C [D [E]]]]]

This command allow specify the exactly expiration time for documents. May be specified per Server/Realm basis and takes effect till the end of config file or till next **ExpireAt** command. **ExpireAt** specified without any arguments disable previously specified value. A - stand for minute, may be * or 0-59; B - stand for hour, may be * or 0-23; C - stand for day of month, may be * or 1-31; D - stand for month, may be * or 1-12; E - stand for day of week, may be * or 0-6, 0 - is Sunday. **ExpireAt** command have higher priority over **Period** or **PeriodByHops** command.

3.10.29. UseDateHeader command

UseDateHeader yes|no

Use Date header if no Last-Modified header is sent by remote web-server. Default value: no.

3.10.30. MaxHops command

MaxHops <number>

Maximum way in "mouse clicks" from start url. Default value is 256. Can be set multiple times before "Server" command and takes effect till the end of config file or till next MaxHops command.

MaxHops 256

3.10.31. TrackHops command

```
TrackHops yes|no
```

This command enable or disable hops tracking in reindexing. Default value is no. If enabled, the value of hops for url is recalculated when reindexing. Otherwise the value of hops is calculated only once at insertion of url into database.

```
TrackHops yes
```

3.10.32. MaxDepth command

```
MaxDepth <number>
```

Maximum directory depth of url. Default value is 16. Can be set multiple times before "Server" command and takes effect till the end of config file or till next MaxDepth command.

```
MaxDepth 2
```

3.10.33. MaxDocsPerServer command

```
MaxDocsPerServer <number>
```

Limits the number of documents retrieved from Server. Default value is -1, that means no limits. If set to positive value, no more than given number of pages will be indexed from one server during this run of index. Can be set multiple times before **Server** command and takes effect till the end of config file or till next **MaxDocsPerServer** command.

```
MaxDocsPerServer 100
```

3.10.34. MaxNetErrors command

```
MaxNetErrors <number>
```

Maximum network errors for each server. Default value is 16. Use 0 for unlimited errors number. If there too many network errors on some server (server is down, host unreachable, etc) **indexer** will try to do not more then 'number' attempts to connect to this server. Takes effect till the end of config file or till next **MaxNetErrors** command.

```
MaxNetErrors 16
```

3.10.35. ReadTimeOut command

```
ReadTimeOut <time>
```

Connect timeout and stalled connections timeout. For <time> format see . Default value is 30 seconds. Can be set any times before **Server** command and takes effect till the end of config file or till next **ReadTimeOut** command.

```
ReadTimeOut 30s
```

3.10.36. DocTimeOut command

```
DocTimeOut <time>
```

Maximum amount of time indexer spends for one document downloading. For <time> format see . Default value is 90 seconds. Can be set any times before **Server** command and takes effect till the end of config file or till next **DocTimeOut** command.

```
DocTimeOut 1m30s
```

3.10.37. NetErrorDelayTime command

```
NetErrorDelayTime <time>
```

Specify document processing delay time if network error has occurred. For <time> format see . Default value is one day

```
NetErrorDelayTime 1d
```

3.10.38. Cookies command

```
Cookies yes/no
```

Enables/Disables the support for HTTP cookies. Command may be used several times before **Server** command and takes effect till the end of config file or till next **Cookies** command. Default value is "no".

```
Cookies yes
```

3.10.39. Robots command

```
Robots yes/no
```

Allows/disallows using `robots.txt` and `<META NAME="robots" ...>` exclusions. Use `no`, for example for link validation of your server(s). Command may be used several times before **Server** command and takes effect till the end of config file or till next **Robots** command. Default value is "yes".

```
Robots yes
```

3.10.40. RobotsPeriod command

By defaults, `robots.txt` data holds in SQL-database for one week. You may change this period using **RobotsPeriod** command:

```
RobotsPeriod <time>
```

For `<time>` format see description of **Period** command in .

```
RobotsPeriod 30d
```

3.10.41. CrawlDelay command

Use this command to specify default pause in seconds between consecutive fetches from same server. This is similar to `crawl-delay` command in `robots.txt` file, but can specified in `indexer.conf` file on per server basis. If no `crawl-delay` value is specified in `robots.txt`, the value of **CrawlDelay** is used. If `crawl-delay` is specified in `robots.txt`, then the maximum of **CrawlDelay** and `crawl-delay` is used as interval between consecutive fetches.

3.10.42. Section command

```
Section <string> <number> <maxlen> [strict] [ <pattern> <replacement> ]
```

where `<string>` is a section name and `<number>` is section ID between 0 and 255. Use 0 if you don't want to index some of these sections. It is better to use different IDs for different sections. In this case during search time you'll be able to give different weight to each section or even disallow some sections at a search time. `<maxlen>` argument contains a maximum length of section which will be stored in database. Use 0 for `<maxlen>`, if you don't want to store this section. `<pattern>` and `<replacement>` are a regex-like pattern and replacement to extract section value from document content.

You can specify `strict` option to set strict string tokenization for a section, which mean word break at any non-character symbol despite the context. It's useful, for example, in indexing of URL, where hyphen, the character, uses as delimiter between words.

```
# Standard HTML sections: body, title
Section body    1 256
Section title   2 128
# strict tokenization for URL
Section url          3          0 strict
# regex-pattern for a section
Section GoodName     4          128 "<h1>([^<]*)</h1>" "<b>GoodName:</b> $1"
```

3.10.43. HrefSection command

```
HrefSection <string> [ <pattern> <replacement> ]
```

where <string> is a section name, <pattern> and <replacement> are a regex-like pattern and replacement to extract section value from document content. Use this command to extract links from document content.

```
# Standard HTML sections: body, title
HrefSection link
HrefSection      NewLink "<newlink>([^<]*)</newlink>" "$1"
```

3.10.44. FastHrefCheck command

The "FastHrefCheck yes" command is useful to speed-up the indexing when you have a huge list of **Server/Realm/Subnet** commands as it disables the href checking against server list during parsing.

3.10.45. Index command

```
Index yes/no
```

Prevent indexer from storing words into database. Useful for example for link validation. Can be set multiple times before **Server** command and takes effect till the end of config file or till next **Index** command. Default value is "yes".

```
Index no
```

3.10.46. ProxyAuthBasic command

```
ProxyAuthBasic login:passwd
```

Use http proxy basic authorization. Can be used before every **Server** command and takes effect only for next one **Server** command! It should be also before **Proxy** command. Examples:

```
ProxyAuthBasic somebody:something
```

3.10.47. Proxy command

```
Proxy your.proxy.host[:port]
```

Use proxy rather then connect directly. One can index ftp servers when using proxy Default port value if not specified is 3128 (Squid) If proxy host is not specified direct connect will be used. Can be set before every **Server** command and takes effect till the end of config file or till next **Proxy** command. If no one **Proxy** command specified indexer will use direct connect. Examples:

```
#           Proxy on atoll.anywhere.com, port 3128:
Proxy atoll.anywhere.com
#           Proxy on lota.anywhere.com, port 8090:
Proxy lota.anywhere.com:8090
#           Disable proxy (direct connect):
Proxy
```

3.10.48. AuthBasic command

```
AuthBasic login:passwd
```

Use basic http authorization. Can be set before every **Server** command and takes effect only for next one **Server** command! Examples:

```
AuthBasic somebody:something
```

```
# If you have password protected directory(-ies), but whole server is open,use:
AuthBasic login1:passwd1
Server http://my.server.com/my/secure/directory1/
AuthBasic login2:passwd2
Server http://my.server.com/my/secure/directory2/
Server http://my.server.com/
```

3.10.49. ServerWeight command

```
ServerWeight <number>
```

Server weight for Popularity Rank calculation (see). Default value is 1.

`ServerWeight 1`

3.10.50. OptimizeAtUpdate command

`OptimizeAtUpdate yes`

Specify word index optimize strategy. Default value: no If enabled, this save disk space, but slow down indexing. May be placed in `indexer.conf` and `cached.conf`.

3.10.51. SkipUnreferred command

`SkipUnreferred yes|no|del`

Default value: no. Use this command to skip reindexing or delete unreferred documents. An unreferred document is a document with no links to it. This command require the links collection to be enabled (see).

3.10.52. Bind command

`Bind 127.0.0.1`

You may use this command to specify local ip address, if your system have several network interfaces.

3.10.53. ProvideReferer command

`ProvideReferer yes`

Use this command to provide `Referer:` request header for HTTP and HTTPS connections.

3.10.54. LongestTextItems command

`LongestTextItems 4`

Use this command to specify the number of longest text items to index.

3.11. Extended indexing features

3.11.1. Indexing SQL database tables (htdb: virtual URL scheme)

DataparkSearch can index SQL database text fields - the so called htdb: virtual URL scheme.

Using htdb:/ virtual scheme you can build full text index of your SQL tables as well as index your database driven WWW server.

Note: You must have PRIMARY key on the table you want to index.

3.11.1.1. HTDB indexer.conf commands

Five `indexer.conf` commands provide HTDB. They are **HTDBAddr**, **HTDBList**, **HTDBLimit**, **HTDBDoc** and **HTDBText**.

HTDBAddr is used to specify database connection. It's syntax identical to **DBAddr** command.

HTDBList is SQL query to generate list of all URLs which correspond to records in the table using PRIMARY key field. You may use either absolute or relative URLs in HTDBList command:

For example:

```
HTDBList "SELECT concat('htdb://',id) FROM messages"
or
HTDBList "SELECT id FROM messages"
```

HTDBLimit command may be used to specify maximal number of records in one SELECT operation. It allow reduce memory usage for big data tables indexing. For example:

```
HTDBLimit 512
```

HTDBDoc is a query to get only certain record from database using PRIMARY key value.

HTDBList SQL query is used for all URLs which end with '/' sign. For other URLs SQL query given in **HTDBDoc** is used.

Note: HTDBDoc query must return FULL HTTP response with headers. So, you can build very flexible indexing system giving different HTTP status in query. Take a look at HTTP response codes section of documentation to understand indexer behavior when it gets different HTTP status.

If there is no result of HTDBDoc or query does return several records, HTDB retrieval system generates "HTTP 404 Not Found". This may happen at reindex time if record was deleted from your table since last reindexing. You may use **HoldBadHrefs 0** to delete such records from DataparkSearch tables as well.

You may use several HTDBDoc/List commands in one `indexer.conf` with corresponding Server commands.

HTDBText <section> is a query to get raw text data from database using PRIMARY key value collected via **HTDBList** command. The <section> parameter is specify the section name useing for storing this data. This query may return as many rows as required. You may specify several **HTDBText** commands per **Server** or **Realm** command.

```
DEAddr mysql://foo:bar@localhost/database/?dbmode=single
```

```
HTDBAddr mysql://foofoo:barbar@localhost/database/
```

```
HTDBList "SELECT DISTINCT topic_id FROM messages"
```

```
HTDBText body "SELECT raw_text\
FROM messages WHERE topic_id='$1'"
```

```
Server htdb:/
```

It's possible to specify both **HTDBDoc** and **HTDBText** commands per one **Server** or **Realm** command. **HTDBText** commands are processing first.

3.11.1.2. HTDB variables

You may use PATH parts of URL as parameters of both HTDBList and HTDBDoc SQL queries. All parts are to be used as \$1, \$2, ... \$n, where number is the number of PATH part:

```
htdb:/part1/part2/part3/part4/part5
      $1      $2      $3      $4      $5
```

For example, you have this `indexer.conf` command:

```
HTDBList "SELECT id FROM catalog WHERE category='$1'"
```

When `htdb:/cars/` URL is indexed, \$1 will be replaced with 'cars':

```
SELECT id FROM catalog WHERE category='cars'
```

You may use long URLs to provide several parameters to both HTDBList and HTDBDoc queries. For example, `htdb:/path1/path2/path3/path4/id` with query:

```
HTDBList "SELECT id FROM table WHERE field1='$1' AND field2='$2' and field3='$3'"
```

This query will generate the following URLs:

```
htdb:/path1/path2/path3/path4/id1
...
htdb:/path1/path2/path3/path4/idN
```

for all values of the field "id" which are in HTDBList output.

3.11.1.3. Creating full text index

Using htdb:/ scheme you can create full text index and use it further in your application. Lets imagine you have a big SQL table which stores for example web board messages in plain text format. You also want to build an application with messages search facility. Lets say messages are stored in "messages" table with two fields "id" and "msg". "id" is an integer primary key and "msg" big text field contains messages themselves. Using usual SQL LIKE search may take long time to answer:

```
SELECT id, message FROM message WHERE message LIKE '%someword%'
```

Using DataparkSearch htdb: scheme you have a possibility to create full text index on "message" table. Install DataparkSearch in usual order. Then edit your `indexer.conf`:

```
DEAddr mysql://foo:bar@localhost/search/?dbmode=single
```

```
HTDBAddr mysql://foofoo:barbar@localhost/database/
```

```
HTDBList "SELECT id FROM messages"
```

```
HTDBDoc "SELECT concat(\
'HTTP/1.0 200 OK\\r\\n',\
'Content-type: text/plain\\r\\n',\
'\\r\\n',\
msg) \
FROM messages WHERE id='$1'"
```

```
Server htdb:/
```

After start indexer will insert 'htdb:/' URL into database and will run an SQL query given in HTDBList. It will produce 1,2,3, ..., N values in result. Those values will be considered as links relative to 'htdb:/' URL. A list of new URLs in the form htdb:/1, htdb:/2, ... , htdb:/N will be added into database. Then HTDBDoc SQL query will be executed for each new URL. HTDBDoc will produce HTTP document for each document in the form:

```
HTTP/1.0 200 OK
Content-Type: text/plain

<some text from 'message' field here>
```

This document will be used to create full text index using words from 'message' fields. Words will be stored in 'dict' table assuming that we are using 'single' storage mode.

After indexing you can use DataparkSearch tables to perform search:

```
SELECT url.url
FROM url,dict
WHERE dict.url_id=url.rec_id
AND dict.word='someword';
```

As far as DataparkSearch 'dict' table has an index on 'word' field this query will be executed much faster than queries which use SQL LIKE search on 'messages' table.

You can also use several words in search:

```
SELECT url.url, count(*) as c
FROM url,dict
WHERE dict.url_id=url.rec_id
AND dict.word IN ('some','word')
GROUP BY url.url
ORDER BY c DESC;
```

Both queries will return 'htdb:/XXX' values in url.url field. Then your application has to cat leading 'htdb:/' from those values to get PRIMARY key values of your 'messages' table.

3.11.1.4. Indexing SQL database driven web server

You can also use htdb:/ scheme to index your database driven WWW server. It allows to create indexes without having to invoke your web server while indexing. So, it is much faster and requires less CPU resources when direct indexing from WWW server.

The main idea of indexing database driven web server is to build full text index in usual order. The only thing is that search must produce real URLs instead of URLs in 'htdb:/...' form. This can be achieved using DataparkSearch aliasing tools.

HTDBList command generates URLs in the form:

```
http://search.mnogo.ru/board/message.php?id=XXX
```

where XXX is a "messages" table primary key values.

For each primary key value HTDBDoc command generates text/html document with HTTP headers and content like this:

```
<HTML>
<HEAD>
<TITLE> ... subject field here .... </TITLE>
<META NAME="Description" Content=" ... author here ...">
</HEAD>
```

```
<BODY> ... message text here ... </BODY>
```

At the end of `doc/samples/htdb.conf` we wrote three commands:

```
Server htdb:/
Realm http://search.mnogo.ru/board/message.php?id=*
Alias http://search.mnogo.ru/board/message.php?id= htdb:/
```

First command says `indexer` to execute `HTDBList` query which will generate a list of messages in the form:

```
http://search.mnogo.ru/board/message.php?id=XXX
```

Second command allow `indexer` to accept such message URLs using string match with `'*'` wildcard at the end.

Third command replaces `"http://search.mnogo.ru/board/message.php?id="` substring in URL with `"htdb:/"` when `indexer` retrieve documents with messages. It means that `"http://mysearch.udm.net/board/message.php?id=xxx"` URLs will be shown in search result, but `"htdb:/xxx"` URL will be indexed instead, where `xxx` is the PRIMARY key value, the ID of record in "messages" table.

3.11.2. Indexing binaries output (exec: and cgi: virtual URL schemes)

`DataparkSearch` supports `exec:` and `cgi:` virtual URL schemes. They allows running an external program. This program must return a result to it's stdout. Result must be in HTTP standard, i.e. HTTP response header followed by document's content.

For example, when indexing both `cgi:/usr/local/bin/myprog` and `exec:/usr/local/bin/myprog`, `indexer` will execute the `/usr/local/bin/myprog` program.

3.11.2.1. Passing parameters to cgi: virtual scheme

When executing a program given in `cgi:` virtual scheme, `indexer` emulates that program is running under HTTP server. It creates `REQUEST_METHOD` environment variable with `"GET"` value and `QUERY_STRING` variable according to HTTP standards. For example, if `cgi:/usr/local/apache/cgi-bin/test-cgi?a=b&d=e` is being indexed, `indexer` creates `QUERY_STRING` with `a=b&d=e` value. `cgi:` virtual URL scheme allows indexing your site without having to invoke web servers even if you want to index CGI scripts. For example, you have a web site with static documents under `/usr/local/apache/htdocs/` and with CGI scripts under `/usr/local/apache/cgi-bin/`. Use the following configuration:

```
Server http://localhost/
```

```
Alias http://localhost/cgi-bin/ cgi:/usr/local/apache/cgi-bin/
Alias http://localhost/ file:/usr/local/apache/htdocs/
```

3.11.2.2. Passing parameters to exec: virtual scheme

indexer does not create QUERY_STRING variable like in cgi: scheme. It creates a command line with argument given in URL after ? sign. For example, when indexing
 exec:/usr/local/bin/myprog?a=b&d=e, this command will be executed:

```
/usr/local/bin/myprog "a=b&d=e"
```

3.11.2.3. Using exec: virtual scheme as an external retrieval system

exec: virtual scheme allow using it as an external retrieval system. It allows using protocols which are not supported natively by DataparkSearch. For example, you can use curl program which is available from <http://curl.haxx.se/> to index HTTPS sites.

Put this short script to /usr/local/dpsearch/bin/ under curl.sh name.

```
#!/bin/sh
/usr/local/bin/curl -i $1 2>/dev/null
```

This script takes an URL given in command line argument and executes curl program to download it. -i argument says curl to output result together with HTTP headers.

Now use these commands in your indexer.conf:

```
Server https://some.https.site/
Alias https:// exec:/usr/local/dpsearch/etc/curl.sh?https://
```

When indexing https://some.https.site/path/to/page.html, indexer will translate this URL to

```
exec:/usr/local/dpsearch/etc/curl.sh?https://some.https.site/path/to/page.html
```

execute the curl.sh script:

```
/usr/local/dpsearch/etc/curl.sh "https://some.https.site/path/to/page.html"
```

and take it's output.

3.11.3. Mirroring

You may specify a path to root dir to enable sites mirroring

```
MirrorRoot /path/to/mirror
```

You may specify as well root directory of mirrored document's headers indexer will store HTTP headers to local disk too.

```
MirrorHeadersRoot /path/to/headers
```

You may specify period during which earlier mirrored files will be used while indexing instead of real downloading.

```
MirrorPeriod <time>
```

It is very useful when you do some experiments with DataparkSearch indexing the same hosts and do not want much traffic from/to Internet. If MirrorHeadersRoot is not specified and headers are not stored to local disk then default Content-Type's given in AddType commands will be used. Default value of the MirrorPeriod is -1, which means do not use mirrored files.

<time> is in the form xxxA[yyyB[zzzC]] (Spaces are allowed between xxx and A and yyy and so on) where xxx, yyy, zzz are numbers (can be negative!). A, B, C can be one of the following:

```
s - second
M - minute
h - hour
d - day
m - month
y - year
```

(these letters are the same as in strftime/strptime functions)

Examples:

```
15s - 15 seconds
4h30M - 4 hours and 30 minutes
1y6m-15d - 1 year and six month minus 15 days
1h-10M+1s - 1 hour minus 10 minutes plus 1 second
```

If you specify only number without any character, it is assumed that time is given in seconds (this behavior is for compatibility with versions prior to 3.1.7).

The following command will force using local copies for one day:

```
MirrorPeriod 1d
```

If your pages are already indexed, when you re-index with `-a` indexer will check the headers and only download files that have been modified since the last indexing. Thus, all pages that are not modified will not be downloaded and therefore not mirrored either. To create the mirror you need to either (a) start again with a clean database or (b) use the `-m` switch.

You can actually use the created files as a full featured mirror to you site. However be careful: indexer will not download a document that is larger than `MaxDocSize`. If a document is larger it will be only partially downloaded. If you site has no large documents, everything will be fine.

3.11.4. Data acquisition

With **ActionSQL** command you can execute SQL-queries with document related data while indexing. The syntax of **ActionSQL** command is as follow:

```
ActionSQL <section> <pattern> <sql-template> [<dbaddr>]
```

where `<section>` is the name of document section to check for regex pattern `>pattern>` match. If a match is found then the `<sql-template>` is filled with regex meta-variables `$1-$9` as well with search template meta-variables (as for example, `$(Title)`, `$(Last-Modified)`, etc.) to form a sql-query, which is executed in the first `DBAddr` defined in `indexer.conf` file. If the optional `<dbaddr>` paramater of **ActionSQL** command is set, a new connection is set according this `DBAddr` and sql-query is executed in this connection.

Thus you can use **ActionSQL** commands to mind and collect the data on pages while indexing. For example, the following command collect phone numbers (in Russian local notation) along with titles of pages where these phone numbers have been discovered:

```
ActionSQL body "\(([0-9]{3})\) [ ]*([0-9]{3})[- \.]*([0-9]{2})[- \.]*([0-9]{2})" "INSERT INTO
```

3.12. Using syslog

DataparkSearch indexer uses syslog to log its messages. Different verbose levels could be specified with `-v` option or by **LogLevel** command in config files:

Table 3-1. Verbose levels

0	SILENT	suppress all log. Bad idea, better use <code>-l</code> option.
1	ERROR	log only critical errors
2	WARNING	log warnings as well
3	INFO	add info messages to log. Default.

4	EXTRA	extra logging
5	DEBUG	for debug purposes

You may use `-l` option to suppress log to stdout/stderr when running indexer via crontab. Without `-l` option log is sent both to stdout/stderr and to log files. If you do not like such behavior, run configure with `--disable-syslog` flag and recompile indexer. Compiled without syslog support, indexer uses only stdout/stderr.

Syslog uses different facilities to separate log messages. The indexer's default facility is LOCAL7. Facility could be changed during configure with `--enable-syslog=LOG_FACILITY` option. LOG_FACILITY should be one of the standard facilities, usually listed in `/usr/include/sys/syslog.h` header file.

Facility helps to separate DataparkSearch messages from others. You can modify `/etc/syslog.conf` to tell syslog how to treat DataparkSearch messages. For example:

```
# Log all messages from DataparkSearch to separate file
local7.*          -/var/log/DataparkSearch.log
```

Other example:

```
# Send all DataparkSearch messages to host named central
# Syslog on central should be configured to allow this
local7.*          @central
```

By default all messages are logged to `/var/log/messages` as well. DataparkSearch could populate this file with a number of messages. To avoid this, add `local7.none` or `local7.!*` (ignore any messages from local7 facility) to your 'catch-all' log files.

For example:

```
#
# Some 'catch-all' logfiles.
#
*.=info;*.=notice;*.=warn;\
    auth,authpriv.none;\
    cron,daemon.none;\
    mail,news.none;\
    local7.!*          -/var/log/messages
```

Please take a look at `syslogd(8)` and `syslog.conf(5)` man pages for more detailed information about syslog and its configuration notes.

3.13. Storing compressed document copies

In DataparkSearch it is possible to store compressed copies of indexed documents. Copies are stored and retrieved by the new daemon - **stored**, that is installed into `sbin` directory of DataparkSearch installation (default: `/usr/local/dpsearch/sbin`).

stored is also used to make documents excerpts for search results. You may use **ExcerptSize** command in `search.htm` template to specify average excerpt size in bytes. Value by default: 256. You may switch off document excerpts (but retain ability to show stored copies) using **DoExcerpt no** command in your search template.

To enable documents copies archiving without **stored** usage, place **DoStore yes** command in your `indexer.conf` file instead of **stored** daemon configuration.

Stored document copies are retrieved by means of `storedoc.cgi` CGI script. It requests a saved copy of a documents from **stored**, then a copy is displayed with user's web browser with search keywords highlighted.

To enable **stored** support, compile DataparkSearch with `zlib` support:

```
./configure --with-zlib <other arguments>
```

You may use the **Store** and **NoStore** commands to allow or disallow storing several files by pattern. For arguments of those commands are exactly the same as for the **Allow** command (see). All documents are stores by defaults, if support for **stored** is enabled.

3.13.1. Configure stored

To start using **stored**, please do the following:

-

```
Copy /usr/local/dpsearch/etc/stored.conf-dist to
/usr/local/dpsearch/etc/stored.conf.
```

```
Edit /usr/local/dpsearch/etc/stored.conf
```

There are several commands available for using with **stored**:

-

Listen tells **stored** which address and/or port to bind to. By default **stored** listens to port 7004 and any address. It is possible to specify port only:

```
Listen 7004
```

Or address only:

```
Listen 127.0.0.2
```

Or both address and port:

```
Listen 127.0.0.2:7004
```

-

VarDir command specifies an alternate `var/` working directory, e.g.

```
VarDir /mnt/d/dpsearch/var/
```

.

StoredFiles command specifies number of stored data files created in `var/stored/` directory, e.g.

```
StoredFiles 256
```

.

OptimizeInterval command specifies interval in seconds between attempts to optimize a stored datafile, e.g.

```
OptimizeInterval 300
```

.

OptimizeRatio command specifies defragmentation threshold for a stored datafile optimization.

```
OptimizeRatio 3
```

.

Run stored:

```
/usr/local/dpsearch/sbin/stored &
```

.

Configure `indexer.conf` and `search.htm` (or `searchd.conf`, if `searchd` is used). Specify address and port that `indexer` will use to connect to `stored`. Use `stored` parameter for **DBAddr** command, e.g.:

```
DBAddr mysql://localhost/search/?dbmode=cache&stored=localhost:7004
```

3.13.2. How stored works

After you have successfully configured `stored`, the `indexer` pass downloaded documents to `stored` daemon. After that, `stored` will compress the received documents and save them.

3.13.3. Using stored during search

To enable displaying stored documents during search, do the following:

.

Configure `storedoc.htm` (`storedoc.cgi` template) if necessary.

.

Add a `storedoc.cgi` link to `search.htm` `<!--res-->` section; e.g. `<A HREF="$ (stored_href) ">Cached copy</A>`

.

Specify `storedoc.cgi` CGI script URL in `search.htm` (by default `$(stored_href)` will return `/cgi-bin/storedoc.cgi`). In case you have to specify other URL, add to `search.htm` variables section the following line:

StoredocURL `/path/to/storedoc.cgi`

Or an absolute path:

StoredocURL `http://servername/path/to/storedoc.cgi`

This is how **stored** works during search, if everything configured correctly:

`search.htm` displays a link to `storedoc.cgi`;

When user clicks the link, `storedoc.cgi` sends a query to **stored** daemon to the address, specified in `storedoc.htm` with the **Stored** parameter of **DBAddr** command;

After the query, **stored** will decompress the requested saved copy and send it to `storedoc.cgi`;

`storedoc.cgi` parses the received document and apply highlighting of search keywords. Highlighting method is specified with `storedoc.htm` **HlBeg** and **HlEnd** commands;

Chapter 4. DataparkSearch HTML parser

4.1. Tag parser

Tag parser understands the following tag notation:

- `< ... parameter=value ... >`
- `< ... parameter="value" ... >`
- `< ... parameter='value' ... >`

4.2. Special characters

indexer understands the following special HTML characters:

- `<`, `>`, `&`, ` `, `"`;
- All HTML-4 character entities: `ä`, `ü`, and other.
- Characters in their Unicode code notation: `ê`;

4.3. META tags

Indexer's HTML parser currently understands the following META tags. Note that "HTTP-EQUIV" may be used instead of "NAME" in all entries.

- `<META NAME="Content-Type" Content="text/html; charset=xxxx">` This is used to detect document character set if it is not specified in Content-type HTTP header.
- `<META NAME="REFRESH" Content="5; URL=http://www.somewhere.com">` URL value will be inserted in database.
-

- `<META NAME="Keywords" Content="xxx">`
- `<META NAME="Description" Content="xxx">`
- `<META NAME="Robots" Content="xxx">` with content value ALL, NONE, INDEX, NOINDEX, FOLLOW, NOFOLLOW, NOARCHIVE.
- `<META NAME="DP.PopRank" Content="xxx">` with Content value as a real number. This is used to assign the initial value of PopularityRank for the page.

4.4. Links

HTML parser understand the following links:

- `<A HREF="xxx">`
- `<IMG SRC="xxx">`
- `<LINK HREF="xxx">`
- `<FRAME SRC="xxx">`
- `<AREA HREF="xxx">`
- `<BASE HREF="xxx">`

Note: If BASE HREF value has incorrectly formed URL, current one will be used instead to compose relative links.

4.5. Comments

-

Text inside the `<!-- .... -->` tag is recognized as HTML comment.

- You may use special `<!--UdmComment--> .... <!--/UdmComment-->` comment tags to exclude the text between from indexing. This may be useful to hide such things like menus and others from indexing.
- You may also use `<NOINDEX> ... </NOINDEX>` as a synonyms to `<!--UdmComment-->` and `<!--/UdmComment-->`
- For compatibility with ASPSeek, `<!--noindex--> ... <!--/noindex-->` are also equivalent to `<!--UdmComment-->` and `<!--/UdmComment-->`
- Google's special comments `<!-- google_ad_section_start -->`, `<!-- google_ad_section_start(weight=ignore) -->` and `<!-- google_ad_section_end -->` consider as tags to include/exclude content for indexing.

4.6. Body patterns

If you need index not whole page, for example, to exclude navigation, ads, etc., you may use **BodyPattern** command to specify a pattern to extract content of a page for indexing. For example:

```
BodyPattern "<!--content--> (.*) <!--/content-->" "$1"
```

this pattern will extract content between special comments and only that content will be indexed for this page.

You may specify several **BodyPattern** commands, but only the first match will be applied to a page. These patterns are trying to apply to all pages indexed. Beware, huge number of such body patterns may hurt indexing speed.

4.7. Sub-documents

The sub-documents are: frames, iframes and embedded objects (flash tubes in general); temporary redirects (often used to place cookies or redirect to a page, depending on the language preferences of the user); versions of the same page in different languages obtained with Content negotiation.

The indexing of sub-documents is controlled by two commands: the **SubDocLevel** command sets the maximum nesting level of a sub-document to be indexed. The default value is 0, which prohibits the sub-document indexing. The **SubDocCnt** command sets the maximum number of sub-documents to be indexed at all nesting levels (this command is mainly to prevent endless cycles of pages nested into each others). The default value is 5.

Chapter 5. Storing data

5.1. SQL storage types

5.1.1. General storage information

DataparkSearch stores every words found in any defined section of document. The count of word appearance in the document does not affect it's weight. But the fact whether the word appears in more important parts of the document (title, description, etc.) is taken in account however.

5.1.2. Various modes of words storage

There are different modes of word storage which are currently supported by DataparkSearch: "single", "multi", "crc", "crc-multi", "cache". Default mode is "cache". Mode is to be selected by `dbmode` parameter of `DBAddr` command in both `indexer.conf` and `search.htm` files.

Examples:

```
DBAddr mysql://localhost/search/?dbmode=single
DBAddr mysql://localhost/search/?dbmode=multi
DBAddr mysql://localhost/search/?dbmode=crc
DBAddr mysql://localhost/search/?dbmode=crc-multi
```

5.1.3. Storage mode - single

When "single" is specified, all words are stored in one table with structure (`url_id`,`word`,`weight`), where `url_id` is the ID of the document which is referenced by `rec_id` field in "url" table. Word has variable char (32) SQL type.

5.1.4. Storage mode - multi

If "multi" is selected, words will be located in different 13 tables depending of their lengths. Structures of these tables are the same with "single" mode, but fixed length char type is used, which is usually faster in most databases. This fact makes "multi" mode usually faster comparing with "single" mode.

5.1.5. Storage mode - crc

If "crc" mode is selected, DataparkSearch will store 32 bit integer word IDs calculated by HASH32 algorithm instead of words. This mode requires less disc space and is faster than "single" and "multi" modes. DataparkSearch uses the fact that HASH32 calculates quite unique check sums for different words. According to our tests there are only 250 pairs of words have the same HASH32 value in the list of about 1.600.000 unique words. Most of these pairs (>90%) have at least one misspelled word. Words

information is stored in the structure (url_id,word_id,weight), where word_id is 32 bit integer ID calculated by HASH32 algorithm. This mode is recommended for big search engines.

5.1.6. Storage mode - crc-multi

When "crc-multi" mode is selected, DataparkSearch stores HASH32 word IDs in several tables with the same to "crc" structures depending on word lengths like in "multi" mode. This mode usually is the most fast and recommended for big search engines.

5.1.7. SQL structure notes

Please note that we develop DataparkSearch with PostgreSQL as back-end and often have no possibility to test each version with all of other supported databases. So, if there is no table definition in create/you_database directory, you may found PostgreSQL definition for the same table and just adopt it for your back-end. PostgreSQL table definitions are always up-to-date.

5.1.8. Additional features of non-CRC storage modes

"single" and "multi" modes support substring search. As far as "crc" and "crc-multi" do not store words themselves and use integer values generated by HASH32 algorithm instead, there is no possibility of substring search in these modes.

5.2. Cache mode storage

5.2.1. Introduction

cache words storage mode is able to index and search quickly through several millions of documents.

5.2.2. Cache mode word indexes structure

The main idea of cache storage mode is that word index and URLs sorting information is stored on disk rather than in SQL database. Full URL information however is kept in SQL database (tables url and urlinfo). Word index is divided into number of files specified by **WrdFiles** command (default value is 0x300). URLs sorting information is divided into number of files specified by **URLDataFiles** command (default value is 0x300).

Note: Beware: you should have identical values for **WrdFiles** and **URLDataFiles** commands in all your configs.

Word index is located in files under `/var/tree` directory of DataparkSearch installation. URLs sorting information is located in files under `/var/url` directory of DataparkSearch installation.

5.2.3. Cache mode tools

There are two additional programs `cached` and `splitter` used in `cache` mode indexing.

`cached` is a TCP daemon which collects word information from indexers and stores it on your hard disk. It can operate in two modes, as old `cachelogd` daemon to logs data only, and in new mode, when `cachelogd` and `splitter` functionality are combined.

`splitter` is a program to create fast word indexes using data collected by `cached`. Those indexes are used later in search process.

5.2.4. Starting cache mode

To start "cache mode" follow these steps:

Start `cached` server:

```
cd /usr/local/dpsearch/sbin
./cached 2>cached.out &
```

It will write some debug information into `cached.out` file. `cached` also creates a `cached.pid` file in `/var` directory of base DataparkSearch installation.

`cached` listens to TCP connections and can accept several indexers from different machines. Theoretical number of indexers connections is equal to 128. In old mode `cached` stores information sent by indexers in `/var/splitter/` directory of DataparkSearch installation. In new mode it stores in `/var/tree/` directory.

By default, `cached` starts in new mode. To run it in old mode, i.e. logs only mode, run it with `-l` switch:

```
cached -l
```

Or by specify **LogsOnly** yes command in your `cached.conf`.

You can specify port for `cached` to use without recompiling. In order to do that, please run

```
./cached -p8000
```

where 8000 is the port number you choose.

You can as well specify a directory to store data (it is `/var` directory by default) with this command:

```
./cached -w /path/to/var/dir
```

Configure your `indexer.conf` as usual and for **DBAddr** command add `cache` as value of `dbmode` parameter and `localhost:7000` as value of `cached` parameter (see).

Run indexers. Several indexers can be executed simultaneously. Note that you may install indexers on different machines and then execute them with the same `cached` server. This distributed system allows making indexing faster.

Flushing cached buffers and url data, and creating cache mode limits. To flush cached buffers and url data and to create cache mode limits after indexing is done, send -HUP signal to `cached`. You can use `cached.pid` file to do this:

```
kill -HUP `cat /usr/local/dpsearch/var/cached.pid`
```

N.B.: you needs wait till all buffers will be flushed before going to next step.

Creating word index. This stage is no needs, if `cached` runs in new, i.e. combined, mode. When some information is gathered by indexers and collected in `/var/splitter/` directory by `cached` it is possible to create fast word indexes. `splitter` program is responsible for this. It is installed in `/sbin` directory. Note that indexes can be created anytime without interrupting current indexing process.

Run `splitter` without any arguments:

```
/usr/local/dpsearch/sbin/splitter
```

It will take sequentially all prepared files in `/var/splitter/` directory and use them to build fast word index. Processed logs in `/var/splitter/` directory are truncated after this operation.

5.2.5. Optional usage of several splitters

`splitter` has two command line arguments: `-f [first file]` `-t [second file]` which allows limiting used files range. If no parameters are specified `splitter` distributes all prepared files. You can limit files range using `-f` and `-t` keys specifying parameters in HEX notation. For example, `splitter -f 000 -t A00` will create word indexes using files in the range from 000 to A00. These keys allow using several splitters at the same time. It usually gives more quick indexes building. For example, this shell script starts four splitters in background:

```
#!/bin/sh
splitter -f 000 -t 3f0 &
splitter -f 400 -t 7f0 &
splitter -f 800 -t bf0 &
splitter -f c00 -t ff0 &
```

5.2.6. Using run-splitter script

There is a `run-splitter` script in `/sbin` directory of DataparkSearch installation. It helps to execute subsequently all three indexes building steps.

"`run-splitter`" has these two command line parameters:

```
run-splitter --hup --split
```

or a short version:

```
run-splitter -k -s
```

Each parameter activates corresponding indexes building step. `run-splitter` executes all three steps of index building in proper order:

Sending -HUP signal to `cached`. `--hup` (or `-k`) `run-splitter` arguments are responsible for this.

Running splitter. Keys `--split` (or `-s`).

In most cases just run **run-splitter** script with all **-k -s** arguments. Separate usage of those three flags which correspond to three steps of indexes building is rarely required.

run-splitter have optional parameters: **-p=n** and **-v=m** to specify pause in seconds after each log buffer update and verbose level respectively. **n** is seconds number (default value: 0), **m** is verbosity level (default value: 4).

5.2.7. Doing search

To start using **search.cgi** in the "cache mode", edit as usually your **search.htm** template and add the "cache" as value of **dbmode** parameter of **DBAddr** command.

5.2.8. Using search limits

To use search limits in cache mode, you should add appropriate **Limit** command(s) to your **indexer.conf** (or **cached.conf**, if **cached** is used) and to **search.htm** or **searchd.conf** (if **searchd** is used).

To use, for example, search limit by tag, by category and by site, add follow lines to **search.htm** or to **indexer.conf** (**searchd.conf**, if **searchd** is used).

```
Limit t:tag
Limit c:category
Limit site:siteid
```

where **t** - name of CGI parameter (&t=) for this constraint, **tag** - type of constraint.

Instead of tag/category/siteid in example above you can use any of values from table below:

Table 5-1. Cache limit types

category	Category limit.
tag	Tag limit.
time	Time limit (a hour precision).
language	Language limit.
content	Content-Type limit.
siteid	url.site_id limit.
link	Limit by pages what links to url.rec_id specified.
hostname (obsolete)	Hostname (url) limit. This limit is obsolete and should be replaced by site_id limit

5.3. DataparkSearch performance issues

The cache mode is the fastest DataparkSearch's storage mode. Use it if you need maximal search speed.

If your `/var` directory isn't changing since indexing is finished, you may disable file locking using **"ColdVar yes"** command placed in `search.htm` (or in `searchd.conf`, if `searchd` is used). This allow you to save some time on file locking.

Using **UseCRC32URLId yes** command (see) allow to speed up indexing, but small number of collisions is possible, especially on large database.

5.3.1. searchd usage recommendation

If you plan use ispell data, synonym or stopword lists, it's recommended setup the `searchd` daemon for speed-up searches (See). `searchd` daemon preload all these data and lists and holds them in memory. This reduce average search query execution time.

Also, `searchd` can preload url info data (20 bytes per URL indexed) and cache mode limits (4 or 8 bytes per URL depend on limit type). This allow reduce average search time.

5.3.2. Memory based filesystem (mfs) usage recommendation

If you use cache storage mode and you have enough RAM on your PC, you may place `/usr/local/dpsearch/var` directory on memory based filesystem (mfs). This allow speedup both indexing and searching.

If you haven't enough RAM to fit `/usr/local/dpsearch/var`, you may place on memory filesystem any of `/usr/local/dpsearch/var/tree`, `/usr/local/dpsearch/var/url` or `/usr/local/dpsearch/var/store` directories as well.

5.3.3. URLInfoSQL command

For dbmode cache, you may use **URLInfoSQL no** command to disable storing URL Info into SQL database. But using this command, you'll be unable to use limits by language and by Content-Type.

5.3.4. MarkForIndex command

By default, DataparkSearch are marking all URLs selected for indexing as indexed for 4 hours. This prevent possible simultaneous indexing of the same URL by different indexer instance running. But for huge installation this feature can take some time for processing. You may switch off this markage using **"MarkForIndex no"** in your `indexer.conf` file.

5.3.5. CheckInsertSQL command

By default, DataparkSearch trying to insert data into SQL database regardless it's already present there. On some systems this raise some error loggings. To avoid such errors, you may enable additional checks, is the inserting data new, by specifying **CheckInsertSQL yes** command in your `indexer.conf`.

5.3.6. MySQL performance

MySQL users may declare DataparkSearch tables with `DELAY_KEY_WRITE=1` option. This will make the updating of indexes faster, as these are not logged to disk until the file is closed. `DELAY_KEY_WRITE` excludes updating indexes on disk at all.

With it indexes are processed only in memory and written onto disk as last resort, command **FLUSH TABLES** or `mysqld` shutdown. This can take even minutes and impatient user can kill `-9 mysql` server and break index files with this. Another downside is that you should run `myisamchk` on these tables before you start `mysqld` to ensure that they are okay if something killed `mysqld` in the middle.

Because of it we didn't include this table option into default tables structure. However as the key information can always be generated from the data, you should not lose anything by using `DELAY_KEY_WRITE`. So, use this option for your own risk.

5.3.7. Post-indexing optimization

This article was supplied by Randy Winch <gumby@cafes.net>

I have some performance numbers that some of you might find interesting. I'm using RH 6.2 with the 2.2.14-6.1.1 kernel update (allows files larger than 2 gig) and `mysql 2.23.18-alpha`. I have just indexed most of our site using `mnoGoSearch 3.0.18`:

mnoGoSearch statistics			
Status	Expired	Total	

200	821178	2052579	OK
301	797	29891	Moved Permanently
302	3	3	Moved Temporarily
304	0	7	Not Modified
400	0	99	Bad Request
403	0	7	Forbidden
404	30690	100115	Not found
500	0	1	Internal Server Error
503	0	1	Service Unavailable

Total	852668	2182703	

I optimize the data by dumping it into a file using `SELECT * INTO OUTFILE`, sort it using the system sort routine into word (CRC) order and then reloading it into the database using the procedure described

in the `mysql` online manual

(http://www.mysql.com/documentation/mysql/commented/manual.php?section=Insert_speed).

The performance is wonderful. My favorite test is searching for "John Smith". The optimized database version takes about 13 seconds. The raw version takes about 73 seconds.

```
Search results: john : 620241 smith : 177096
Displaying documents 1-20 of total 128656 found
```

5.3.8. Asynchronous resolver library

Using `ares` (<ftp://athena-dist.mit.edu/pub/ATHENA/ares/>), an asynchronous resolver library (`dns/ares` in FreeBSD ports collection), allow to perform DNS queries without blocking for every indexing thread. Please note, this also increase the number of concurrent queries to your DNS server.

5.4. SearchD support

5.4.1. Why using searchd

- Faster searching, especially when using ISpell, synonyms or segmenters for east asian languages. Related files are loaded into memory when `searchd` is started, while `search.cgi` loads data before every query.
- Also, `searchd` can preload url info data (20 bytes per URL indexed) and cache mode limits (4 or 8 bytes per URL depend on limit type). This allow reduce average search time.
- It is possible to distribute words index and web-server between different machines.

5.4.2. Starting searchd

To start using `searchd`:

- Copy `$PREFIX/etc/searchd.conf-dist` to `searchd.conf`.
- Edit `searchd.conf`.
- If you need preload url data to speed-up searches (this cost about 20 bytes of memory per url), add the following command to `searchd.conf`:

```
PreloadURLData yes
```

- You may also preload cache mode limits for most frequently used limit values using **PreloadLimit** command in `searchd.conf` file:

```
PreloadLimit <limit type> <limit value>
```

For example:

```
PreloadLimit tag Unix
```

-

Add the following command to `search.htm`:

```
DBAddr searchd://hostname/ or DBAddr searchd://hostname:port/, e.g.
```

```
DBAddr searchd://localhost/
```

Default port value is 7003

-

You may start several `searchd`'s children answering search queries simultaneously. Use **MaxClients** command to specify the number of `searchd`'s children. Value by default is 1.

```
MaxClients 2
```

-

Start `searchd`:

```
/usr/local/dpsearch/sbin/searchd &
```

To suppress output to `stderr`, use `-l` option. The output will go through `syslog` only (in case `syslog` support was not disabled during installation with `--disable-syslog`). In case `syslog` is disabled, it is possible to direct `stderr` to a file:

```
/usr/local/dpsearch/sbin/searchd 2>/var/log/searchd.log &
```

`searchd` just like `indexer` can be used with an option of a configuration file, e.g. relative path to `/etc` directory of `DataparkSearch` installation:

```
searchd searchd1.conf
```

or with absolute path:

```
searchd /usr/local/dpsearch/etc/searchd1.conf
```

5.5. Oracle notes

5.5.1.

5.5.1.1. Why Oracle?

Oracle is a powerful, tunable, scalable and reliable industrial RDBMS. It provides some functionalities which are absent in simple freeware RDBMS like MySQL and PostgreSQL, such as: transactions

support, concurrency and consistency, data integrity, partitioning, replication, cost-based and rule-based optimizers, parallel execution, redo logs, RAW devices and many other features. Although Oracle is a very functional database, the additional qualities like reliability impose some overhead. In fact, providing many advantages Oracle has some disadvantages. For example great tenability requires more experienced DBA, redo logs support provide great reliability against instance and media failures but requires more efficient disk system. I think you should select Oracle as a database for DataparkSearch if you want to search through hundreds of megabytes or several gigabytes of information, reliability is one of the primary concerns, need high availability of the database, and you are ready to pay higher sums for hardware and Oracle DBA to achieve better quality of service.

5.5.1.2. DataparkSearch+Oracle8 Installation Requirements

In order to install DataparkSearch with Oracle RDBMS support you must ensure the following requirements:

- Oracle8 Server must be properly installed on any computer accessible from the site where DataparkSearch are to be installed. See the documentation provided with your Oracle server.
- Oracle client software and libraries must be installed on the site where you plan to install DataparkSearch. I strongly recommend to install utilities also, it help you to test the client and server accessibility.
- glibc 2.0 or glibc 2.1. Oracle 8.0.5.X libraries are built for glibc 2.0.

5.5.1.3. Currently supported/tested platforms

Oracle versions:

- Oracle 8.0.5.X

Operation systems:

- Linux RedHat 6.1 (2.2.X + glibc 2.0)

Oracle Server may be ran on any platform supporting tcp/ip connections. I see no difficulties to port DataparkSearch Oracle driver to any commercial and freeware unix systems, any contribution is appreciated.

5.5.2. Compilation, Installation and Configuration

5.5.2.1. Compilation

Oracle 8.0.5.X and Linux RedHat 6.1

```
./Configure --with-oracle8=oracle_home_dir
make
make install
```

If you have any troubles, try to put `CC = i386-glibc20-linux-gcc` in the `src/Makefile`, this is old version of gcc compiler for glibc 2.0.

5.5.2.2. Installation and Configuration

Check whether Oracle Server and Oracle Client work properly.

First, try DataparkSearch service is accessible

```
[oracle@ant oracle]$ tnsping DataparkSearch 3

TNS Ping Utility for Linux: Version 8.0.5.0.0 - Production on 29-FEB-00 09:46:12
(c) Copyright 1997 Oracle Corporation. All rights reserved.

Attempting to contact (ADDRESS=(PROTOCOL=TCP) (Host=ant.gpovz.ru) (Port=1521))
OK (10 msec)
OK (0 msec)
OK (10 msec)
```

Second, try to connect to Oracle Server with `svrmgrl` and check whether DataparkSearch tables were created

```
[oracle@ant oracle]$ svrmgrl command='connect scott/tiger@DataparkSearch'

Oracle Server Manager Release 3.0.5.0.0 - Production

(c) Copyright 1997, Oracle Corporation. All Rights Reserved.

Oracle8 Release 8.0.5.1.0 - Production
PL/SQL Release 8.0.5.1.0 - Production

Connected.
SVRMGR> SELECT table_name FROM user_tables;
TABLE_NAME
-----
DICT
```

```

DICT10
DICT11
DICT12
DICT16
DICT2
DICT3
DICT32
DICT4
DICT5
DICT6
DICT7
DICT8
DICT9
PERFTEST
ROBOTS
STOPWORD
TAB1
URL
19 rows selected.

```

Check the library paths in /etc/ld.so.conf

```

[oracle@ant oracle]$ cat /etc/ld.so.conf
/usr/X11R6/lib
/usr/lib
/usr/i486-linux-libc5/lib
/usr/lib/qt-2.0.1/lib
/usr/lib/qt-1.44/lib
/oracle8/app/oracle/product/8.0.5/lib

```

This file should contain line `oracle_home_path/lib` to ensure DataparkSearch will be able to open `libclntsh.so`, the shared Oracle Client library

Make symbolic link:

```
ln -s /oracle8/app/oracle/product/8.0.5/network/admin/tnsnames.ora /etc
```

Correct the indexer.conf file

You should specify `DBName`, `DBUser`, `DBPass` in order that DataparkSearch can connect to Oracle Server. `DBName` is the service name, it should have the same name that was written to `tnsnames.ora` file, `DBUser` and `DBPass` are Oracle user and his password correspondingly. You can run `indexer` now.

Setting up search.cgi

Copy the file `/usr/local/dpsearch/bin/search.cgi` to `apache_root/cgi-bin/search.cgi`. Then add two lines to `apache's http.conf` file:

```

SetEnv ORACLE_HOME /oracle8/app/oracle/product/8.0.5
PassEnv ORACLE_HOME

```

Correct the `search.htm` to provide DBName, DBUser, DBPass information. `search.cgi` should work now.

Chapter 6. Subsections

6.1. Tags

Tag is a special parameter which can be given for a set of documents. The main purpose of tags is to join a number of documents into one group and then while doing search to select a group of documents to search through.

You can use **Tag** command of `indexer.conf` to give some tag value for a server or server subset. While doing search you can specify tag value to search through documents which tag matches given parameter with `t=xxx` parameter, which can be passed from HTML form. Take a look into `, indexer.conf-dist` and `search.htm-dist` for explanation and examples.

Note: For dbmode cache you need to have the following section defined in your `sections.conf` file:

```
Section tag 0 64
```

in overall you need to have the section 'tag' defined with non-zero maximum length.

6.1.1. Tag command

Tag <string>

Use this field for your own purposes. For example for grouping some servers into one group, etc... During search you'll be able to limit URLs to be searched through by their tags. Can be set multiple times before **Server** command and takes effect till the end of config file or till next **Tag** command. Default values is an empty string.

6.1.2. TagIf command

```
TagIf <tag> [Match|NoMatch] [NoCase|Case] [String|Regex] <section> <arg> [<arg> ... ]
```

Mark document by <tag> tag, if the value of `section` match the `arg` pattern given. The meaning of first three optional parameters are exactly the same as for the **Allow** command (see).

Example

```
TagIf Docs regex Title Manual
```

6.1.3. Tags in SQL version

Tag type is CHAR. CHAR type allows to use some nice features. You can use '_' and '%' LIKE wildcards in tag parameter when doing search. It makes possible that tag, like a category, does support an idea of nesting. For example, documents with tag value "AB" can be found with both "A%" and "AB" tag limits.

Tags also give a way to make an URL a member of multiple tag selections. Playing with LIKE wildcards you can easily create two or more groups.

For example, tag "ABCDE" is the member of at least these selections:

```
_BCDE
A_CDE
AB_DE
AEC_E
ABCD_
```

Note: If you have big enough database and often use tag limits, it is useful to create an index by field "tag" in "server" table. This index is not created by default.

```
CREATE INDEX srv_tag ON "server" ("tag");
```

By default, the length of tag field in url table is limited by 16 characters. If you need more, increase this length before DB creating.

Note: For cache storage mode, you can use SQL's wildcards only with **indexer**.

6.2. Categories

There is a categories editor written in Perl. You can get it in perl/cat_ed/ subdirectory of DataparkSearch installation.

Categories are similar to tag feature, but nested. So you can have one category inside another and so on.

Basic points:

- there are up to 6 nested levels;
- every level occupies 2 hex or 36 base digits;
-

parameter for setting category is *path*, which can be empty (means root category) or up to 10 chars long (means deepest category).

You can also set up symlinks, e.g. categories that are actually links to other categories. `link` database field is used for that. In the symlink last two characters should be `@@`. In example above `Moto->BMW` is a link to `Auto->BMW`.

First notice that category in the server table is set to be 11 characters long. This means you can use a valid character to keep track of categories. If you are going to keep a category tree of any size, then I would suggest using the category editor. But anyways, here's how it works.

You can use either the tag column or the category column in the server for the same thing. Or you can categorize a site in two different ways. For example you could keep track of sites that are owned by a certain company and then categorize them as well. You could use the tag option to keep of ownership and use the category option for categories. When I explain the category option, it goes the same for the tag option.

A category can be broken down any way you choose. But for it to work with the category editor, I believe for now, you have to use two characters for each level. If you use the category editor you have the choice to use a hex number going from 0-F or a 36 base number going from 0-Z. Therefore a top-level category like 'Auto' would be 01. If it has a subcategory like 'Ford', then it would be 01 (the parent category) and then 'Ford' which we will give 01. Put those together and you get 0101. If 'Auto' had another subcategory named 'VW', then its id would be 01 because it belongs to the 'Ford' category and then 02 because it's the next category. So its id would be 0102. If VW had a sub category called 'Engine' then it's id would start at 01 again and it would get the 'VW' id 02 and 'Auto' id of 01, making it 010201.

If you want to search for sites under that category then you pass it `cat=010201` in the url...so create a select box and give like that:

```
<OPTION value="01">AUTO
<OPTION value="0101">Ford
```

and so on...

Note: For dbmode cache you need to have the following section defined in your `sections.conf` file:

```
Section category 0 64
```

i.e. in overall you need to have the section 'category' defined with non-zero maximum length.

6.2.1. Category command

```
Category <string>
```

You may distribute documents between nested categories. Category is a string in hex number notation. You may have up to 6 levels with 256 members per level. Empty category means the root of category tree. Take a look into `for` for more information.

```
# This command means a category on first level:  
Category AA  
# This command means a category on 5th level:  
Category FFAABBCCDD
```

6.2.2. CategoryIf command

CategoryIf <category> [Match|NoMatch] [NoCase|Case] [String|Regex] <section> <arg> [<arg> .

Mark document by <category> category, if the value of section match arg pattern given. The meaning of first three optional parameters is exactly the same as for the **Allow** command (see).

Example

```
CategoryIf 010F regex Title "JOB ID"
```

Chapter 7. Languages support

7.1. Character sets

7.1.1. Supported character sets

DataparkSearch supports almost all known 8 bit character sets as well as some multi-byte charsets including Korean euc-kr, Chinese big5 and gb2312, Japanese shift-jis, euc-jp and iso-2022-jp, as well as UTF-8. Some multi-byte character sets are not supported by default, because the conversion tables for them are rather large that leads to increase of the executable files size. See `configure` parameters to enable support for these charsets.

DataparkSearch also supports the following Macintosh character sets: MacCE, MacCroatian, MacGreek, MacRoman, MacTurkish, MacIceland, MacRomania, MacThai, MacArabic, MacHebrew, MacCyrillic, MacGujarati.

Table 7-1. Language groups

Language group	Character sets
Arabic	cp864, ISO-8859-6, MacArabic, windows-1256
Armenian	armscii-8
Baltic	cp775, ISO-8859-13, ISO-8859-4, windows-1257
Celtic	ISO-8859-14
Central European	cp852, ISO-8859-16, ISO-8859-2, MacCE, MacCroatian, MacRomania, windows-1250
Chinese Simplified	GB2312, GBK
Chinese Traditional	Big5, Big5-HKSCS, cp950, GB-18030
Cyrillic	cp855, cp866, cp866u, ISO-8859-5, KOI-7, KOI8-C, KOI8-R, KOI8-U, MacCyrillic, windows-1251
Georgian	georgian-academy, georgian-ps, geostd8
Greek	cp869, cp875, ISO-8859-7, MacGreek, windows-1253
Hebrew	cp862, ISO-8859-8, MacHebrew, windows-1255
Icelandic	cp861, MacIceland
Indian	MacGujarati, tscii
Iranian	ISIRI3342
Japanese	EUC-JP, ISO-2022-JP, Shift_JIS
Korean	EUC-KR
Lao	cp1133
Nordic	cp865, ISO-8859-10

South Eur	ISO-8859-3
Tajik	KOI8-T
Thai	cp874, ISO-8859-11, MacThai
Turkish	cp1026, cp857, ISO-8859-9, MacTurkish, windows-1254
Unicode	sys-int, UTF-16BE, UTF-16LE, UTF-8
Vietnamese	VISCII, windows-1258
Western	cp437, cp500, cp850, cp860, cp863, IBM037, ISO-8859-1, ISO-8859-15, MacRoman, US-ASCII, windows-1252

7.1.2. Character sets aliases

Each charset is recognized by a number of its aliases. Web servers can return the same charset in different notation. For example, iso-8859-2, iso8859-2, latin2 are the same charsets. There is support for charsets names aliases which search engine can understand:

Table 7-2. Charsets aliases

armSCII-8	armSCII-8, armSCII8
Big5	big-5, big-five, big5, bigfive, cn-big5, csbig5
Big5-HKSCS	big5-hkscs, big5_hkscs, big5hk, hkscs
cp1026	1026, cp-1026, cp1026, ibm1026
cp1133	1133, cp-1133, cp1133, ibm1133
cp437	437, cp437, ibm437
cp500	500, cp500, ibm500
cp775	775, cp775, ibm775
cp850	850, cp850, cspc850multilingual, ibm850
cp852	852, cp852, ibm852
cp855	855, cp855, ibm855
cp857	857, cp857, ibm857
cp860	860, cp860, ibm860
cp861	861, cp861, ibm861
cp862	862, cp862, ibm862
cp863	863, cp863, ibm863
cp864	864, cp864, ibm864
cp865	865, cp865, ibm865
cp866	866, cp866, csibm866, ibm866

cp866u	866u, cp866u
cp869	869, cp869, csibm869, ibm869
cp874	874, cp874, cs874, ibm874, windows-874
cp875	875, cp875, ibm875, windows-875
cp950	950, cp950, windows-950
EUC-JP	cseucjp, euc-jp, euc_jp, eucjp, ujis, x-euc-jp
EUC-KR	cseuckr, euc-kr, euc_kr, euckr
GB-18030	gb-18030, gb18030
GB2312	chinese, cn-gb, csgb2312, csiso58gb231280, euc-cn, euc_cn, euccn, gb2312, gb_2312-80, iso-ir-58
GBK	cp936, gbk, windows-936
georgian-academy	georgian-academy
georgian-ps	georgian-ps
geostd8	geo8-gov, geostd8
IBM037	037, cp037, csibm037, ibm037
ISIRI3342	isiri-3342, isiri3342
ISO-2022-JP	csiso2022jp, iso_2022-jp, iso-2022-jp
ISO-8859-1	cp819, csisolatin1, ibm819, iso_8859-1, iso-8859-1, iso-ir-100, iso8859-1, iso_8859-1, iso_8859-1:1987, 11, latin-1, latin1
ISO-8859-10	csisolatin6, iso_8859-10, iso-8859-10, iso-ir-157, iso8859-10, iso_8859-10, iso_8859-10:1992, 16, latin-6, latin6
ISO-8859-11	iso_8859-11, iso-8859-11, iso8859-11, iso_8859-11, iso_8859-11:1992, tactis, thai, tis-620, tis620
ISO-8859-13	iso_8859-13, iso-8859-13, iso-ir-179, iso8859-13, iso_8859-13, 17, latin-7, latin7
ISO-8859-14	iso_8859-14, iso-8859-14, iso-ir-199, iso8859-14, iso_8859-14, iso_8859-14:1998, 18, latin-8, latin8
ISO-8859-15	iso_8859-15, iso-8859-15, iso-ir-203, iso8859-15, iso_8859-15, iso_8859-15:1998, 19, latin-0, latin-9, latin0, latin9
ISO-8859-16	iso_8859-16, iso-8859-16, iso-ir-226, iso8859-16, iso_8859-16, iso_8859-16:2000
ISO-8859-2	csisolatin2, iso_8859-2, iso-8859-2, iso-ir-101, iso8859-2, iso_8859-2, iso_8859-2:1987, 12, latin-2, latin2
ISO-8859-3	csisolatin3, iso_8859-3, iso-8859-3, iso-ir-109, iso8859-3, iso_8859-3, iso_8859-3:1988, 13, latin-3, latin3

ISO-8859-4	csisolatin4, iso 8859-4, iso-8859-4, iso-ir-110, iso8859-4, iso_8859-4, iso_8859-4:1988, 14, latin-4, latin4
ISO-8859-5	csisolatincyrillic, cyrillic, iso 8859-5, iso-8859-5, iso-ir-144, iso8859-5, iso_8859-5, iso_8859-5:1988
ISO-8859-6	arabic, asmo-708, csisolatinarabic, ecma-114, iso 8859-6, iso-8859-6, iso-ir-127, iso8859-6, iso_8859-6, iso_8859-6:1987
ISO-8859-7	csisolatingreek, ecma-118, elot_928, greek, greek8, iso 8859-7, iso-8859-7, iso-ir-126, iso8859-7, iso_8859-7, iso_8859-7:1987
ISO-8859-8	csisolatinhebrew, hebrew, iso 8859-8, iso-8859-8, iso-ir-138, iso8859-8, iso_8859-8, iso_8859-8:1988
ISO-8859-9	csisolatin5, iso 8859-9, iso-8859-9, iso-ir-148, iso8859-9, iso_8859-9, iso_8859-9:1989, 15, latin-5, latin5
KOI-7	iso-ir-37, koi-7, koi7
KOI8-C	cskoi8C, koi8-c, koi8c
KOI8-R	cskoi8r, koi8-r, koi8r
KOI8-T	cskoi8t, koi8-t, koi8t
KOI8-U	cskoi8u, koi8-u, koi8u
MacArabic	macarabic
MacCE	cmac, macce, maccentraleurope, x-mac-ce
MacCroatian	maccroatian
MacCyrillic	maccyrillic, x-mac-cyrillic
MacGreek	macgreek
MacGujarati	macgujarati
MacHebrew	machebrew
MacIceland	macisland
MacRoman	csmacintosh, mac, macintosh, macroman
MacRomania	macromania
MacThai	macthai
MacTurkish	macturkish
Shift_JIS	csshiftjis, ms_kanji, s-jis, shift-jis, shift_jis, sjis, x-sjis
sys-int	sys-int
tscii	tscii
US-ASCII	ansi_x3.4-1968, ascii, cp367, csascii, ibm367, iso-ir-6, iso646-us, iso_646.irv:1991, us, us-ascii

UTF-16BE	utf-16, utf-16be, utf16, utf16be
UTF-16LE	utf-16le, utf16le
UTF-8	utf-8, utf8
VISCII	csviscii, viscii, viscii1.1-1
windows-1250	cp-1250, cp1250, ms-ee, windows-1250
windows-1251	cp-1251, cp1251, ms-cyr, ms-cyrl, win-1251, win1251, windows-1251
windows-1252	cp-1252, cp1252, ms-ansi, windows-1252
windows-1253	cp-1253, cp1253, ms-greek, windows-1253
windows-1254	cp-1254, cp1254, ms-turk, windows-1254
windows-1255	cp-1255, cp1255, ms-hebr, windows-1255
windows-1256	cp-1256, cp1256, ms-arab, windows-1256
windows-1257	cp-1257, cp1257, winbaltrim, windows-1257
windows-1258	cp-1258, cp1258, windows-1258

7.1.3. Recoding

`indexer` recodes all documents to the character set specified in the **LocalCharset** command in your `indexer.conf` file. Internally recoding is implemented using Unicode. Please note that if some recoding can't convert a character directly from one charset to another, DataparkSearch will use HTML numeric character references to escape this character (i.e. in form `&#NNN`; where `NNN` - a character code in Unicode). Thus, for any **LocalCharset** you do not lost any information about indexed documents, but on **LocalCharset** selection depend the database volume you will get after indexing.

7.1.4. Recoding at search time

You may display search results in any charset supported by DataparkSearch. Use **BrowserCharset** command in `search.htm` to select charset for search results. This charset may be different from **LocalCharset** specified. All recodings will done automatically.

7.1.5. Document charset detection

`indexer` detects document character set in this order:

```
"Content-type: text/html; charset=xxx"
```

```
<META NAME="Content-Type" CONTENT="text/html; charset=xxx">
```

Selection of this variant may be switch off by command: **GuesserUseMeta no** in your `indexer.conf`.

Defaults from "Charset" field in Common Parameters

7.1.6. Automatic charset guesser

DataparkSearch has an automatic charset and language guesser. It currently recognizes more than 100 various charsets and languages. Charset and language detection is implemented using "N-Gram-Based Text Categorization" (<http://www.maxime.net.ru/doc/guess.en.shtml>) technique. There is a number of so called "language map" files, one for each language-charset pair. They are installed under `/usr/local/dpsearch/etc/langmap/` directory by default. Take a look there to check the list of currently provided charset-language pairs. Guesser works fine for texts bigger than 500 characters. Shorter texts may not be guessed well.

7.1.6.1. LangMapFile command

Load language map for charset and language guesser from the given file. You may specify either absolute file name or a name relative to DataparkSearch `/etc` directory. You may use several **LangMapFile** commands.

```
LangMapFile langmap/en.ascii.lm
```

7.1.6.2. Build your own language maps

To build your own language map use `dpguesser` utility. In addition, you need to collect file with language samples in charset desired. For new language map creation, use the following command:

```
dpguesser -p -c charset -l language < FILENAME > language.charset.lm
```

You can also use `dpguesser` utility for guessing document's language and charset by existing language maps. To do this, use following command:

```
dpguesser [-n maxhits] < FILENAME
```

For some languages, it may be used few different charset. To convert from one charset supported by DataparkSearch to another, use `dpconv` utility.

```
dpconv [OPTIONS] -f charset_from -t charset_to [configfile] < infile > outfile
```

You may also specify `-e` switch for `dpconv` to use HTML escape entities for input, and `-E` switch - for output.

By default, both `dpguesser` and `dpconv` utilities is installed into `/usr/local/dpsearch/sbin/` directory.

DataparkSearch can update language and charset maps automatically while indexing, if remote server is supply exactly specified language and charset with pages. To enable this function, specify the following command in your `indexer.conf` file:

```
LangMapUpdate yes
```

By default, DataparkSearch uses only first 8192 bytes of each file indexed to detect language and charset. You may change this value using **GuesserBytes** command. Use value of 0 to use all text from document indexed.

```
GuesserBytes 16384
```

7.1.7. Default charset

Use **RemoteCharset** command in `indexer.conf` to choose the default charset of indexed servers.

7.1.8. Default Language

You can set default language for Servers by using `DefaultLang` `indexer.conf` variable. This is useful while restricting search by URL language.

```
DefaultLang <string>
```

Default language for server. Can be used if you need language restriction while doing search.

```
DefaultLang en
```

7.1.9. LocalCharset command

Defines the charset which will be used to store data in database. All other character sets will be recoded into given charset. Take a look into [for detailed explanation](#) how to choose a **LocalCharset** depending on languages used on your site(s). This command should be used once and takes global effect for the config file. Take a look into documentation to check whole list of supported charsets. Default LocalCharset is iso-8859-1 (latin1).

```
LocalCharset koi8-r
```

7.1.10. ForceIISCharset1251 command

This option is useful for users which deals with Cyrillic content and broken (or misconfigured ?) Microsoft IIS web servers, which tends to not report charset correctly. This is really dirty hack, but if this option is turned on it is assumed that all servers which reports as 'Microsoft' or 'IIS' have content in Windows-1251 charset. This command should be used only once in configuration file and takes global effect. Default: no

```
ForceIISCharset1251 yes
```

7.1.11. RemoteCharset command

```
RemoteCharset <charset>
```

<charset> is default character set for the server in next **Server**, **Realm** or **Subnet** command(s). This is required only for "bad" servers that do not send information about charset in header: "Content-type: text/html; charset=some_charset" and do not have <META NAME="Content" Content="text/html; charset="some_charset"> Can be set before every **Server**, **Realm** or **Subnet** command and takes effect till the end of config file or till next **RemoteCharset** command. Default value is iso-8859-1 (latin1).

```
RemoteCharset iso-8859-5
```

7.1.12. URLCharset command

```
URLCharset <charset>
```

<charset> is character set for the URL argument in next **Server**, **Realm** or **URL** command(s). This command specify character set only for arguments in commands follow and havn't effect on charset detection for indexing pages. Have less priority than **RemoteCharset**. Can be set before every **Server**, **Realm** or **URL** command and takes effect till the end of config file or till next **URLCharset** command. Default value is ISO-8859-1 (latin1).

```
URLCharset KOI8-R
```

7.1.13. CharsToEscape command

```
CharsToEscape "\"&<>![ ]"
```

Use this command in your search template to specify the list of characters to escape for \$&(x) search template meta-variables.

7.2. Making multi-language search pages

Original idea instructions by Craig Small <csmall@eye-net.com.au>. Some minor changes by Alex Barkov <bar@mnogosearch.org>.

It is often required to allow for different languages which means different search.htm files depending on what language users have set in their browser.

Further installation should be done in three steps.

Installing several templates.

The general idea is to have one `search.php` or `search.cgi` file and then many `search.[language].htm` files. You also have a `search.htm` file (usually a symlink to `search.en.htm`) for the default.

If the name of the script is `search.en.php` (or `search.en.cgi`) then both CGI and PHP front-ends will be looking for `/somewhere/etc/search.en.htm` assuming that `/somewhere/etc/` is `/etc/` directory of DataparkSearch installation.

You would then populate `/somewhere/etc/` with all the `search.htm` files so `/somewhere/etc` has:

<code>search.en.htm</code>	English template
<code>search.pl.htm</code>	Polish template
<code>search.ru.htm</code>	Russian template
<code>search.htm</code>	Symlink to English template

Installing front-ends

Create a directory and put `search.cgi` or `search.php` there (along with the include files if you want, but I fiddle with the `php include_path` and put them elsewhere).

Then setup the symlinks:

<code>search.cgi</code>	Original file
<code>search.en.cgi</code>	symlink
<code>search.pl.cgi</code>	symlink
<code>search.ru.cgi</code>	symlink

Or in the case of PHP front-end:

<code>search.php</code>	Original file
<code>search.en.php</code>	symlink
<code>search.pl.php</code>	symlink
<code>search.ru.php</code>	symlink

Configuring Apache

Then you need to make apache understand what weirdness you are doing here. So you need to get negotiation happening and some magic with the indexes. I used `.htaccess` file but you could stick it in the apache config proper.

```
AddLanguage en .en
AddLanguage pl .pl
AddLanguage ru .ru
```

```
DirectoryIndex search search.cgi (or search.php)
Options FollowSymlinks MultiViews
```

7.2.1. How does it work?

You type url `http://myhost/mydir/search` *no slash at end !!*

Your browser says "I like english (well language negotiation en) "

Apache finds `search.en.cgi` (DirectoryIndex gives `search`, MultiViews gives the `en.cgi`)

The `SCRIPT_FILENAME` which is used in both `search.cgi` and `search.php` is `somepath/search.en.cgi`

Note: Most other variables will give the wrong result, either `search` or `search.cgi`)

Your hack in `config.inc` means you will use `search.en.htm`.

So what happens if the user wants, say, German? Well there is no `search.de.cgi` (`search.de.php`) so the first bit of DirectoryIndex fails, so it tries the second one, `search.php` OK, they get the page in English, but it's better than a 404.

This does work, you may need some more apache fiddling to get negotiation to work because I am testing this on a server that already has it setup, so I may have missed something.

7.2.2. Possible troubles

You may get some language negotiation problems caused by:

- Dumb caches that don't follow standards
- Dumb versions of browsers that don't follow standards
- Dumb users fiddling with language settings and putting weird stuff in.

The apache team is working on some workarounds for most of these, if possible. For a reasonably heavily used web site you can expect an email about it once a week or so.

7.3. Segmenters for Chinese, Japanese, Korean and Thai languages

Chinese, Japanese, Korean and Thai writings have no spaces between words in phrase as in western languages. Thus, while indexing documents in these languages, it's need additionally to segment phrases into words.

Sometimes, a text in Chinese, Japanese, Korean or Thai can be typed with a space between every hieroglyph for better view. In this case, you may use "**ResegmentChinese yes**", "**ResegmentJapanese**

yes", **"ResegmentKorean yes"** or **"ResegmentThai yes"** commands to index a text typed in such way. With resegmenting enabled, all spaces between characters are removing and then all the text is segmenting again using DataparkSearch's segmenters (see below).

7.3.1. Japanese language phrase segmenter

For Japanese language phrase segmenting the one of ChaSen (<http://chasen.aist-nara.ac.jp/>), a morphological system for Japanese language, or MeCab (<http://cl.aist-nara.ac.jp/~taku-ku/software/mecab/>), a Japanese morphological analyser, is used. Thus, you need one of these systems to be installed before DataparkSearch's configuring and building.

To enable Japanese language phrase segmenting use `--enable-chasen` or `--enable-mecab` switch for **configure**.

7.3.2. Chinese language phrase segmenter

For Chinese language phrase segmenting the frequency dictionary of Chinese words is used. And segmenting itself is done by dynamic programming method to maximize the cumulative frequency of produced words.

To enable Chinese language phrase segmenting it's need to enable the support for Chinese charsets while DataparkSearch configuring, and specify the frequency dictionary of Chinese words by **LoadChineseList** command in `indexer.conf` file.

```
LoadChineseList [charset dictionaryfilename]
```

By default, the GB2312 charset and `mandarin.freq` dictionary is used.

Note: You need to download frequency dictionaries from our web site, or from one of our mirrors, see

7.3.3. Thai language phrase segmenter

For Thai language phrase segmenting the frequency dictionary of Thai words is used. And segmenting itself is done as for Chinese language.

To enable Thai language phrase segmenting it's need to specify the frequency dictionary of Thai words by **LoadThaiList** command in `indexer.conf` file.

```
LoadThaiList [charset dictionaryfilename]
```

By default, the `tis-620` charset and `thai.freq` dictionary is used.

Note: You need to download frequency dictionaries from our web site, or from one of our mirrors, see

.

7.3.4. Korean language phrase segmenter

For Korean language phrase segmenting the frequency dictionary of Korean words is used. And segmenting itself is done as for Chinese language.

To enable Korean language phrase segmenting it's need to specify the frequency dictionary of Korean words by **LoadKoreanList** command in `indexer.conf` file.

```
LoadKoreanList [charset dictionaryfilename]
```

By default, the `eu-kr` charset and `korean.freq` dictionary is used.

Note: You need to download frequency dictionaries from our web site, or from one of our mirrors, see

.

7.4. Multilingual servers support

Some web-servers can handle language negotiation for documents language. In this case, for one URL exist several copies in different languages.

For indexing all pages of such servers, **VaryLang** command is used. It specify list of languages separated by spaces. These languages will used for indexing URL with multi-language versions.

Usage example:

```
VaryLang "ru en fr"
```

index will fetch all document copies in Russian, English and French languages.

Chapter 8. Searching documents

8.1. Using search front-ends

8.1.1. Performing search

Open your preferred front-end in Web browser:

`http://your.web.server/path/to/search.cgi`

To find something just type words you want to find and press SUBMIT button. For example: **mysql** **odbc**. DataparkSearch will find all documents that contain word **mysql** and/or word **odbc**. Best documents having bigger weights will be displayed first.

To find a phrase, simple enclose it in quotes. For example: "**uncontrollable sphere**".

8.1.2. Search parameters

DataparkSearch front-ends support the following parameters given in CGI query string. You may use them in HTML form on search page.

Table 8-1. Available search parameters

q	text parameter with search query
vq	text parameter with search query in the Verity Query Language (prefix variant), see . To use this parameter, you need to leave empty the q parameter.
s	characters sequence, specify results sorting order. Small caps specify ascendant sorting, upper caps - descendant. Following characters can be used: R or r - for sorting by relevance, P or p - for sorting by PopularityRank, I or i - for sorting by Importance (multiplication of relevance and PopularityRank), A or a - for sorting by sum of relevance and PopularityRank, D or d - for sorting by last modified date. Default value: RP.

ps	page size, number of search results displayed on one page, 20 by default. Maximum page size is 100. This value does not allow passing very big page sizes to avoid server overload and might be changed with MAX_PS definition in <code>search.c</code> .
np	page number, starting by 0, 0 by default (first page)
p	page number starting by 1. Suitable for use with OpenSearch
m	search mode. Currently "all", "any", "near" and "bool" values are supported.
wm	word match. You may use this parameter to choose word match type. There are "wrđ", "beg", "end" and "sub" values that respectively mean whole word, word beginning, word ending and word substring match.
t	Tag limit. Limits search through only documents with given tag. This parameter has the same effect with -t indexer option
c	Category limit. Take a look into for details .
ul	URL limit, URL substring to limit search through subsection of database. It supports SQL % and _ LIKE wildcards. This parameter has the same effect with -u indexer option. If relative URL is specified <code>search.cgi</code> inserts % signs before and after "ul" value when compiled with SQL support. It allows to write URL substring in HTML from to limit search, for example <OPTION VALUE="/manual/"> instead of VALUE="%/manual/%". When full URL with schema is specified <code>search.cgi</code> adds % sign only after this value. For example for <OPTION VALUE="http://localhost/"> <code>search.cgi</code> will pass <code>http://localhost/%</code> in SQL LIKE comparison. Not supported for cache storage mode.
wf	Weight factors. It allows changing different document sections weights at a search time. Should be passed in the form of hex number. Check the explanation below.
g	Language limit. Language abbreviation to limit search results by <code>url.lang</code> field.
tmpl	Template filename (without path). To specify template file other standard <code>search.htm</code> .

type	Content-Type limit. Content-type to limit search results by url.content_type field. For cache mode storage this should be exact match. For SQL-modes it may be sql-like pattern.
sp	Words forms limit. =1, if you need search all forms (include spelling suggestions, if aspell support is enabled) for entered words. =0, if you need search only entered words. Default value is 1. You may switch it to 0 for faster search.
sy	Synonyms limit. =1, if you need add synonyms for entered words. =0, do not use synonyms. Default value is 1. You may switch it to 0 for faster search.
empty	Use limits to show results if no query words is entered (only for cache mode). =yes, to show results from limits, if no query words is entered (default). =no, do not show results from limits, if no query words is entered.
dt	Limit by time. Three types is supported. If dt is set to back, that means you want to limit result to recent pages, and you should specify this recentness in variable dp. If dt is set to er, that means the search will be limited to pages newer or older than date given. Variable dx is newer/older flag (1 means newer or after, -1 means older or before). Date is specified in variables dm, dy, dd. If dt is set to range, that means search within given range of dates. Variables db and de are used here and stands for beginning and end date. All times in cache mode measured in a hour precision.
dp	Limit by recentness, if dt value is back. It should be specified in xxxA[yyyB[zzzC]] format. Spaces are allowed between xxx and A and yyy and so on). xxx, yyy, zzz are numbers (can be negative!), A, B, C can be one of the following (the letters are the same as in strptime/strftime functions): s - second, M - minute, h - hour, d - day, m - month, y - year. Examples: 4h30M - 2 hours and 30 minutes 1Y6m-15d - 1 year and six month minus 15 days 1h-60M+1s - 1 hour minus 60 minutes plus 1 second

dx	is newer/older flag (1 means newer or after, -1 means older or before), if dt value is er.
dm	Month, if dt value is er. 0 - January, 1 - February, ... 11 - December.
dy	Year, if dt value is er. Four digits. For example, 1999 or 2001.
dd	Day, if dt value is er. 1...31.
db	Beginning date, if dt value is range. Each date is string in the form dd/mm/yyyy, there dd is day, mm is month and yyyy is four-digits year.
de	End date, if dt value is range. Each date is string in the form dd/mm/yyyy, there dd is day, mm is month and yyyy is four-digits year.

8.1.3. Changing different document parts weights at search time

It is possible to pass "wf" HTML form variable to `search.cgi`. "wf" variable represents weight factors for specific document parts. Currently body,title,keywords,description,url parts, crosswords as well as user defined META and HTTP headers are supported. Take a look into "Section" part of `indexer.conf-dist`.

To be able use this feature it is recommended to set different sections IDs for different document parts in "Section" `indexer.conf` command. Currently up to 256 different sections are supported.

Imagine that we have these default sections in `indexer.conf`:

```
Section body      1  256
Section title     2  128
Section keywords  3  128
Section description 4  128
```

"wf" value is a string of hex digits ABCD. Each digit is a factor for corresponding section weight. The most right digit corresponds to section 1. For the given above sections configuration:

- D is a factor for section 1 (body)
- C is a factor for section 2 (title)
- B is a factor for section 3 (keywords)
- A is a factor for section 4 (description)

Examples:

`wf=0001` will search through body only.

`wf=1110` will search through title,keywords,description but not

through the body.

`wf=F421` will search through:

Description with factor 15 (F hex)

Keywords with factor 4

Title with factor 2

Body with factor 1

By default, if "wf" variable is omitted in the query, all sections factors are 1, it means all sections have the same weight.

By default, DataparkSearch uses fast relevance calculation. In this case, only zero and non-zero values for "wf" variable takes an effect (this allows only include/exclude specified sections in search results). To enable full support for dynamic section weight, you need specify `--enable-fullrel` option for `configure` when configuring DataparkSearch.

8.1.4. Using front-end with an.shtml page

When using a dynamic.shtml page containing SSI that calls `search.cgi`, i.e. `search.cgi` is not called directly as a CGI program, it is necessary to override Apache's `SCRIPT_NAME` environment attribute so that all the links on search pages lead to the dynamic page and not to `search.cgi`.

For example, when a.shtml page contains a line `<--#include virtual="search.cgi">`, `SCRIPT_NAME` variable will still point to `search.cgi`, but not to the.shtml page.

To override `SCRIPT_NAME` variable we implemented a `DPSEARCH_SELF` variable that you may add to Apache's `httpd.conf` file. Thus `search.cgi` will check `DPSEARCH_SELF` variable first and then `SCRIPT_NAME`. Here is an example of using `DPSEARCH_SELF` environment variable with `SetEnv/PassEnv` Apache's `httpd.conf` command:

```
SetEnv DPSEARCH_SELF /path/to/search.cgi
PassEnv DPSEARCH_SELF
```

8.1.5. Using several templates

It is often required to use several templates with the same `search.cgi`. There are actually several ways to do it. They are given here in the order how `search.cgi` detects template name.

`search.cgi` checks environment variable `DPSEARCH_TEMPLATE`. So you can put a path to desired search template into this variable.

`search.cgi` checks path info part of URL available in the `PATH_INFO` environment variable. E.g.

`http://localhost/cgi-bin/search.cgi/search1.html` uses `search1.htm` as its template, and `http://localhost/cgi-bin/search.cgi/search2.html` uses `search2.htm`, and so on.

`search.cgi` also supports Apache internal redirect. It checks `REDIRECT_STATUS` and `REDIRECT_URL` environment variables. To activate this way of template usage you may add these lines in Apache `srn.conf`:

```
AddType text/html .zhtml
AddHandler zhtml .zhtml
Action zhtml /cgi-bin/search.cgi
```

Put `search.cgi` into your `/cgi-bin/` directory. Then put HTML template into your site directory structure under any name with `.zhtml` extension, for example `template.zhtml`. Now you may open search page: `http://www.site.com/path/to/template.zhtml` You may use any unused extension instead of `.zhtml` of course.

If the above two ways fail, `search.cgi` opens a template which has the same name with the script being executed using `SCRIPT_NAME` environment variable. `search.cgi` will open a template `ETC/search.htm`, `search1.cgi` will open `ETC/search1.htm` and so on, where ETC is `DataparkSearch /etc` directory (usually `/usr/local/dpsearch/etc`). So, you can use the same `search.cgi` with different templates without having to recompile it. Just create one or several hard or symbolic links for `search.cgi` or copy it and put corresponding search templates into `/etc` directory of `DataparkSearch` installation.

Take a look also into Making multi-language search pages section

8.1.6. Search operators

The operator **allin<section>**:, where `<section>` is the name of a section, defined in `sections.conf` file (or in any `dpsearch`'s configuration file by **Section** command) with non-zero section number (see), that operator allows to limit the search domain for a query word by the section specified.

This operator differ from limiting search domain using `&wf=` CGI-variable in a way, that such limit is imposing only on query words specified after this operator.

For example, if you have the following commands in `sections.conf` file

```
Section body 1 256
Section title 2 128
Section url 3 0 strict
```

then you can use the following operators in search query: **allinbody:**, **allintitle:** and **allinurl:**.

For the query **computer allintitle: science** it will be found the documents that contain the word "science" in the title and the word "computer" in any document section.

8.1.7. Advanced boolean search

If you want more advanced results you may use query language. You should select "bool" search mode in the search from.

`DataparkSearch` understands the following boolean operators:

AND or **&** - logical AND. For example, `"mysql & odbc"` or `"mysql AND odbc"` - `DataparkSearch` will find any URLs that contain both "mysql" and "odbc".

NEAR - NEAR operator, identical to AND operator, but come true if both words are within 16 words of each other. For example, `"mysql NEAR odbc"` - `DataparkSearch` will find any URLs that contain both "mysql" and "odbc" within 16 words of each other.

ANYWORD or ***** - ANYWORD operator, identical to **AND** operator, but come true if both words have any one word between and left operand have lesser position than right operand. For example, "**mysql** ***** **odbc**" - DataparkSearch will find any URLs that contain both "mysql" and "odbc" within any word between, for example, any document with "mysql via odbc" phrase.

OR or **|** - logical OR. For example, "**mysql** **|** **odbc**" or "**mysql** **OR** **odbc**" - DataparkSearch will find any URLs that contain word "mysql" or word "odbc".

NOT or **~** - logical NOT. For example, "**mysql** **&** **~** **odbc**" or "**mysql** **AND** **NOT** **odbc**" - DataparkSearch will find URLs that contain word "mysql" and do not contain word "odbc" at the same time. Note that **~** just excludes given word from results. Query "**~** odbc" will find nothing!

() - group command to compose more complex queries. For example "(mysql | msql) **&** **~** postgres". Query language is simple and powerful at the same time. Just consider query as usual boolean expression.

8.1.8. The Verity Query Language, VQL

Only the prefix variant of the Verity Query Language is supported by DataparkSearch.

Also, only the following subset of VQL operators is supported by DataparkSearch:

Table 8-2. VQL operators supported by DataparkSearch

<ACCRUE>	equal to OR operator in boolean mode.
<AND>	equal to AND operator in boolean mode.
<ANY>	equal to OR operator in boolean mode.
<NEAR>	equal to NEAR operator in boolean mode.
<NOT>	equal to NOT operator in boolean mode.
<OR>	equal to OR operator in boolean mode.
<PHRASE>	equal to a phrase in boolean mode.
<WORD>	is considered as an empty operator.

8.1.9. How search handles expired documents

Expired documents are still searchable with their old content.

8.2. mod_dpsearch module for Apache httpd

Since version 4.19 DataparkSearch also provide the `mod_dpsearch.so` module for Apache web server.

8.2.1. Why using mod_dpsearch

- As for **searchd** (see), **mod_dpsearch** can hold preloaded in memory some data to speed-up searches.
- In additional, **mod_dpsearch** hold in memory last used search template. This save time on template loading and parsing for every request since second.
- As a plus, the **mod_dpsearch** itself already loaded into memory when search request come from user, while **search.cgi** usually loads from disk for every search request.

8.2.2. Configuring mod_dpsearch

To enable this extension, add `--enable-apache-module` switch to **configure**. In addition, the `mod_dpsearch.so` shared library will be created and installed into Apache tree. Then you need activate this module by adding following line into Apache configuration file:

```
LoadModule dpsearch_module      libexec/mod_dpsearch.so
AddModule mod_dpsearch.c

<IfModule mod_dpsearch.c>
DataparkSearchdConf /usr/local/dpsearch/etc/modsearchd.conf
    <Location /search>
        SetHandler dpsearch
        DataparkSearchTemplate /usr/local/dpsearch/etc/modsearch.htm
    </Location>
    <Location /storedoc>
        SetHandler dpstoredoc
        DataparkStoredocTemplate /usr/local/dpsearch/etc/modstoredoc.htm
    </Location>
</IfModule>
```

There are three configuration directives supported by this module: `DataparkSearchdConf`, `DataparkSearchTemplate` and `DataparkStoredocTemplate`. The `DataparkSearchdConf` optional directive specify a **searchd** related configuration file. It may be only one per server. The `DataparkSearchdTemplate` directive specify a search template file. The `DataparkStoredocTemplate` directive specify a storedoc template file. There can be several templates specified per servers, by one per location. If `DataparkSearchdConf` directive specified, there no need specify `DBAddr` command in templates.

8.3. How to write search result templates

DataparkSearch users have an ability to customize search results (output of `search.cgi`). You may do it by providing template file `search.htm`, which should be located in `/etc/` directory of DataparkSearch installation.

Template file is usual HTML file, which is divided into sections. Keep in mind that you can just open template file in your favorite browser and get the idea of how the search results will look like.

Note: Each templates line should not exceed 1024 bytes.

Each section begins with `<!--sectionname-->` and ends with `<!--/sectionname-->` delimiters, which should reside on a separate line.

Each section consists of HTML formatted text with special meta symbols. Every meta symbol is replaced by it's corresponding string. You can think of meta symbols as of variables, which will have their appropriate values while displaying search results.

Format of variables is the following:

```
$ (x) - plain value
$&(x) - HTML-escaped value and search words highlighted.
$*(x) - HTML-escaped value.
$%(x) - value escaped to be used in URLs
$^(x) - search words highlighted.
$(x:128) - value truncated to the first 128 bytes, if longer.
$(x:UTF-8) - value written in UTF-8 charset. You may specify any charset supported.
$(x:128:right) - value truncated to the last 128 bytes, if longer.
```

8.3.1. Template sections

The following section names are defined:

8.3.1.1. TOP section

This section is included first on every page. You should begin this section with `<HTML><HEAD>` and so on. Also, this is a definitive place to provide a search form. There are two special meta symbols you may use in this section:

```
$(self) - argument for FORM ACTION tag
$(q) - a search query
$(cat) - current category value
$(tag) - current tag value
$(rN) - random number (here N is a number)
```

If you want to include some random banners on your pages, please use \$(rN). You should also place string like "RN xxxx" in 'variables' section (see below), which will give you a range 0..xxxx for \$(rN). You can use as many up random numbers as you want.

Example: \$(r0), \$(r1), \$(r45) etc.

Simple *top* section should be like this:

```
<!--top-->
<HTML>
<HEAD>
  <TITLE>Search Query: $(q)</TITLE>
</HEAD>
<BODY>

<FORM METHOD=GET ACTION="$(self)">
  <INPUT TYPE="hidden" NAME="u1" VALUE="">
  <INPUT TYPE="hidden" NAME="ps" VALUE="20">
  Search for: <INPUT TYPE="text" NAME="q" SIZE=30
  VALUE="$&(q)">
  <INPUT TYPE="submit" VALUE="Search!"><BR>
</FORM>
<!--/top-->
```

There are some variables defined in FORM.

lang limit results by language. Value is a two-letter language code.

```
<SELECT NAME="lang">
<OPTION VALUE="en" SELECTED="$(lang)">English
.....
</SELECT>
```

u1 is the filter for URL. It allows you to limit results to particular site or section etc. For example, you can put the following in the form

Search through:

```
<SELECT NAME="u1">
<OPTION VALUE="" SELECTED="$(u1)">Entire site
<OPTION VALUE="/manual/" SELECTED="$(u1)">Manual
<OPTION VALUE="/products/" SELECTED="$(u1)">Products
<OPTION VALUE="/support/" SELECTED="$(u1)">Support
</SELECT>
```

to limit your search to particular section.

The expression `SELECTED="$(u1)"` in example above (and all the examples below) allows the selected option to be reproduced on next pages. If search front-end finds that expression it prints the string `SELECTED` only in the case `OPTION VALUE` given is equal to that variable.

`ps` is default page size (e.g. how many documents to display per page).

`q` is the query itself.

`pn` is `ps*np`. This variable is not used by DataparkSearch, but may be useful for example in `<!INCLUDE CONTENT="...">` directive if you want to include result produced by another search engine.

Following variables are concerning advanced search capabilities:

- `m` can be used to choose default search type if your query consists of more than one word. In case `m=any`, the search will try to find at least one word, in case `m=all`, the search is more restrictive - all words should be in the document. If `m=bool` query string is considered as a boolean expression.

- `dt` is time limiting type. There are three types supported.

If `'dt'` is `'back'`, that means you want to limit result to recent pages, and you should specify this "recentness" in variable `'dp'` in the form `xxxA[yyyB[zzzC]]`. Spaces are allowed between `xxx` and `A` and `yyy` and so on). `xxx`, `yyy`, `zzz` are numbers (can be negative!) `A`, `B`, `C` can be one of the following (the letters are the same as in `strptime/strftime` functions):

```
s - second
M - minute
h - hour
d - day
m - month
y - year
```

Examples:

```
4h30m      - 2 hours and 30 minutes
1Y6M-15d   - 1 year and six month minus 15 days
1h-60m+1s  - 1 hour minus 60 minutes plus 1 second
```

If `'dt'` is `'er'` (which is short for newer/older), that means the search will be limited to pages newer or older than date given. Variable `dx` is newer/older flag (1 means "newer" or "after", -1 means "older" or "before"). Date is separated into fields as follows:

```
'dm' - month (0 - January, 1 - February, ..., 11 - December)
'dy' - year (four digits, for example 1999 or 2000)
'dd' - day (1...31)
```

If `'dt'` is `'range'`, that means search within given range of dates. Variables `'db'` and `'de'` are used here and stands for beginning and end date. Each date is string in the form `dd/mm/yyyy`, there `dd` is day, `mm` is month and `yyyy` is four-digits year.

This is the example of FORM part where you can choose between different time limiting options.

```
<!-- 'search with time limits' options -->
<TR><TD>
<TABLE CELLPADDING=2 CELLSPACING=0 BORDER=0>
<CAPTION>
Limit results to pages published within
a specified period of time.<BR>
<FONT SIZE=-1><I>(Please select only one option)
</I></FONT>
```

```

</CAPTION>
<TR>
<TD VALIGN=center><INPUT TYPE=radio NAME="dt"
VALUE="back" CHECKED></TD>
<TD><SELECT NAME="dp">
<OPTION VALUE="0" SELECTED="$ (dp) ">anytime
<OPTION VALUE="10M" SELECTED="$ (dp) ">in the last ten minutes
<OPTION VALUE="1h" SELECTED="$ (dp) ">in the last hour
<OPTION VALUE="7d" SELECTED="$ (dp) ">in the last week
<OPTION VALUE="14d" SELECTED="$ (dp) ">in the last 2 weeks
<OPTION VALUE="1m" SELECTED="$ (dp) ">in the last month
<OPTION VALUE="3m" SELECTED="$ (dp) ">in the last 3 months
<OPTION VALUE="6m" SELECTED="$ (dp) ">in the last 6 months
<OPTION VALUE="1y" SELECTED="$ (dp) ">in the last year
<OPTION VALUE="2y" SELECTED="$ (dp) ">in the last 2 years
</SELECT>
</TD>
</TR>
<TR>
<TD VALIGN=center><INPUT type=radio NAME="dt" VALUE="er">
</TD>
<TD><SELECT NAME="dx">
<OPTION VALUE="1" SELECTED="$ (dx) ">After
<OPTION VALUE="-1" SELECTED="$ (dx) ">Before
</SELECT>

```

or on

```

<SELECT NAME="dm">
<OPTION VALUE="0" SELECTED="$ (dm) ">January
<OPTION VALUE="1" SELECTED="$ (dm) ">February
<OPTION VALUE="2" SELECTED="$ (dm) ">March
<OPTION VALUE="3" SELECTED="$ (dm) ">April
<OPTION VALUE="4" SELECTED="$ (dm) ">May
<OPTION VALUE="5" SELECTED="$ (dm) ">June
<OPTION VALUE="6" SELECTED="$ (dm) ">July
<OPTION VALUE="7" SELECTED="$ (dm) ">August
<OPTION VALUE="8" SELECTED="$ (dm) ">September
<OPTION VALUE="9" SELECTED="$ (dm) ">October
<OPTION VALUE="10" SELECTED="$ (dm) ">November
<OPTION VALUE="11" SELECTED="$ (dm) ">December
</SELECT>
<INPUT TYPE=text NAME="dd" VALUE="$ (dd) " SIZE=2 maxlength=2>
,
<SELECT NAME="dy" >
<OPTION VALUE="1990" SELECTED="$ (dy) ">1990
<OPTION VALUE="1991" SELECTED="$ (dy) ">1991
<OPTION VALUE="1992" SELECTED="$ (dy) ">1992
<OPTION VALUE="1993" SELECTED="$ (dy) ">1993
<OPTION VALUE="1994" SELECTED="$ (dy) ">1994
<OPTION VALUE="1995" SELECTED="$ (dy) ">1995
<OPTION VALUE="1996" SELECTED="$ (dy) ">1996
<OPTION VALUE="1997" SELECTED="$ (dy) ">1997
<OPTION VALUE="1998" SELECTED="$ (dy) ">1998

```

```

<OPTION VALUE="1999" SELECTED="{$ (dy) ">1999
<OPTION VALUE="2000" SELECTED="{$ (dy) ">2000
<OPTION VALUE="2001" SELECTED="{$ (dy) ">2001
</SELECT>
</TD>
</TR>
</TR>
<TD VALIGN=center><INPUT TYPE=radio NAME="dt" VALUE="range">
</TD>
<TD>
Between
<INPUT TYPE=text NAME="db" VALUE="{$ (db) " SIZE=11 MAXLENGTH=11>
and
<INPUT TYPE=text NAME="de" VALUE="{$ (de) " SIZE=11 MAXLENGTH=11>
</TD>
</TR>
</TABLE>
</TD></TR>
<!-- end of stl options -->

```

8.3.1.2. *BOTTOM* section

This section is always included last in every page. So you should provide all closing tags which have their counterparts in top section. Although it is not obligatory to place this section at the end of template file, but doing so will help you to view your template as an ordinary html file in a browser to get the idea how it's look like.

Below is an example of bottom section:

```

<!--bottom-->
<P>
<HR>
<DIV ALIGN=right>
<A HREF="http://www.maxime.net.ru/">
<IMG SRC="dpsearch.gif" BORDER=0
ALT="[Powered by DataparkSearch search engine software]">
</A>
</BODY>
</HTML>
<!--/bottom-->

```

8.3.1.3. *RESTOP* section

This section is included just before the search results. It's a good idea to provide some common search results. You can do so by using the next meta symbols:

- \$(first) - number of First document displayed on this page

- `$(last)` - number of Last document displayed on this page
- `$(total)` - total number of found documents
- `$(WE)` - search results with full statistics of every word form search
- `$(W)` - search results with information about the number of the word form found and the number of all word forms found delimited with "/" sign for every search word, e.g. if the search result is `test : 25/73`, it means that the number of word form "test" found is 25, and the number of all its forms ("test", "tests", "testing", etc.) found is 73.
- `$(WS)` - search results in short form with the number of all word forms found.
- `$(SearchTime)` - search query execution time.
- `$(ndocs)` - number of documents in database.

Below is an example of 'restop' section:

```
<!--restop-->
<TABLE BORDER=0 WIDTH=100%>
<TR>
<TD>Search<BR>results:</TD>
<TD><small>$(WE)</small></TD>
<TD><small>$(W)</small></TD>
</TR>
</TABLE>
<HR>
<CENTER>
Displaying documents $(first)-$(last) of total <B>$(total)</B> found.
</CENTER>
<!--/restop-->
```

8.3.1.4. RES section

This section is used for displaying various information about every found document. The following meta symbols are used:

- `$(URL)` Document URL
- `$(Title)` Document Title
- `$(Score)` Document Rating (as calculated by DataparkSearch)

- `$(Body)` Document text, the document excerpt, if stored is used, or the first couple of lines, otherwise, to give an idea of what the document is about).
- `$(Content-Type)` Document Content-type (for example, `text/html`)
- `$(Last-Modified)` Document Last-Modified date
- `$(Content-Length)` Document Size in bytes
- `$(FancySize)` Document Size in bytes, kilobytes or megabytes, what best match.
- `$(Order)` Overall Document Number (in order of appearance), i.e. from 1 to `$(total)`.
- `$(Pos)` Document Number on the page (in order of appearance), i.e. from 1 to `$(ps)`.
- `$(meta.description)` Document Description (from META DESCRIPTION tag)
- `$(meta.keywords)` Document Keywords (from META KEYWORDS tag)
- `$(DY)` Document category with links, i.e. `/home/computers/software/www/`
- `$(CL)` Clone List (see for details)
- `$(BrowserCharset)` Charset used to display search results
- `$(PerSite)` Total number of document from this site, if grouping by site is enabled, =0 otherwise.

Note: It is possible to specify maximum number of characters returned by any of the above variables. E.g. `$(URL)` may return a long URL that may break page table structure. To specify maximum number of characters in the displayed URL's, use `$(URL:xx)`, where `xx` - maximum number of characters:

`$(URL:40)`

will return a URL, and if it is longer than 40 character, only 40 characters will be displayed including the ending points:

`http://very.long.url/path/veery/long/...`

Here is an example of res section:

```
<!--res-->
<DL><DT>
<b>$(Order) </b><a href="$(URL)" TARGET="_blank">
<b>$(Title) </b></a> [ <b>$(Score) </b>] <DD>
$(Body) ...<BR>
<b>URL: </b>
<A HREF="$(URL)" TARGET="_blank">$(URL) </A>$(Content-Type) <BR>
$(Last-Modified), $(Content-Length) bytes<BR>
<b>Description: </b>$(meta.description)<br>
<b>Keywords: </b>$(meta.keywords)<br>
</DL>
<UL>
$(CL)
</UL>
<!--/res-->
```

8.3.1.5. *CLONE* section

The contents of this section is included in result just instead of \$(CL) meta symbol for every document clone found. This is used to provide all URLs with the same contents (like mirrors etc.). You can use the same \$(D*) meta symbols here as in 'res' section. Of course, some information about clone, like \$(DS), \$(DR), \$(DX) will be the same so it is of little use to place it here.

Below is an example of 'clone' section.

```
<!--clone-->
<li><A HREF="$ (DU) " TARGET="_blank">$ (DU) </A>  ( $ (DC) )  $ (DM)
<!--/clone-->
```

8.3.1.6. *RESBOT* section

This is included just after last 'res' section. You usually give a navigation bar here to allow user go to next/previous results page.

This is an example of 'resbot' section:

```
<!--resbot-->
<HR>
<CENTER>
Result pages: $ (NL) $ (NB) $ (NR)
</CENTER>
<!--/resbot-->
```

Navigator is a complex thing and therefore is constructed from the following template sections:

8.3.1.7. *navleft, navleft_nop* section

These are used for printing the link to the previous page. If that page exists, <!--navleft--> is used, and on the first page there is no previous page, so <!--navleft_nop--> is used.

```
<!--navleft-->
<TD><A HREF="$ (NH) "><IMG...></A><BR>
<A HREF="$ (NH) ">Prev</A></TD>
<!--/navleft-->

<!--navleft_nop-->
<TD><IMG...><BR>
<FONT COLOR=gray>Prev</FONT></TD>
<!--/navleft_nop-->
```

8.3.1.8. navbar0 section

This is used for printing the current page in the page list.

```
<!--navbar0-->
<TD><IMG...><BR>$(NP) </TD>
<!--navbar0-->
```

8.3.1.9. navright, navright_nop section

These are used for printing the link to the next page. If that page exists, <!--navright--> is used, and on the last page <!--navright_nop--> is used instead.

```
<!--navright-->
<TD>
<A HREF="$(NH) "><IMG...></A>
<BR>
<A HREF="$(NH) ">Next</A></TD>
<!--/navright-->

<!--navright_nop-->
<TD>
<IMG...>
<BR>
<FONT COLOR=gray>Next</FONT></TD>
<!--/navright_nop-->
```

8.3.1.10. navbar1 section

This is used for printing the links to the other pages in the page list.

```
<!--navbar1-->
<TD>
<A HREF="$(HR) ">
<IMG...></A><BR>
<A HREF="$(NH) ">$(NP) </A>
</TD>
<!--/navbar1-->
```

8.3.1.11. notfound section

As its name implies, this section is displayed in case when no documents are found. You usually give a little message saying that and maybe some hints how to make search less restrictive.

Below is an example of notfound section:

```

<!--notfound-->
<CENTER>
Sorry, but search hasn't returned results.<P>
<I>Try to compose less restrictive search query or check spelling.</I>
</CENTER>
<HR>
<!--/notfound-->

```

8.3.1.12. *noquery* section

This section is displayed in case when user gives an empty query. Below is an example of *noquery* section:

```

<!--noquery-->
<CENTER>
You haven't typed any word(s) to search for.
</CENTER>
<HR>
<!--/noquery-->

```

8.3.1.13. *error* section

This section is displayed in case some internal error occurred while searching. For example, database server is not running or so. You may provide the following meta symbol: `$ (E)` - error text.

Example of error section:

```

<!--error-->
<CENTER>
<FONT COLOR="#FF0000">An error occured!</FONT>
<P>
<B>$ (E)</B>
</CENTER>
<!--/error-->

```

8.3.2. Variables section

There is also a special variables section, in which you can set up some values for search.

Special variables section usually looks like this:

```

<!--variables
DBAddr      mysql://foo:bar@localhost/search/?dbmode=single

```

```

VarDir          /usr/local/dpsearch/var/
LocalCharset    iso-8859-1
BrowserCharset  iso-8859-1
TrackQuery     no
Cache          no
DetectClones    yes
HlBeg          <font color="blue"><b><i>
HlEnd          </i></b>
R1             100
R2             256
Synonym         synonym/english.syn
ResultContentType text/xml
Locale         fr_FR.ISO_8859-1
-->

```

Note: Database option **DBAddr** like in `indexer.conf`, host part in **DBAddr** argument takes affect for natively supported databases only and does not matter for ODBC databases. In case of ODBC use database name part of **DBAddr** to specify ODBC DSN.

VarDir command specifies a custom path to directory that indexer stores data to when use with cache mode. By default `/var` directory of DataparkSearch installation is used.

LocalCharset specifies a charset of database. It must be the same with `indexer.conf` **LocalCharset**.

BrowserCharset specifies which charset will be used to display results. It may differ from **LocalCharset**. All template variables which correspond data from search result (such as document title, description, text) will be converted from **LocalCharset** to **BrowserCharset**. Contents of template itself is not converted, it must be in **BrowserCharset**.

Use "**Cache** yes/no" to enable/disable search results cache.

Use "**DetectClones** yes/no" to enable/disable clones detection.

Use "**GroupBySite** yes/no" to enable/disable grouping results by `url.site_id`.

Note: If `searchd` is used you should place **GroupBySite** in your `searchd.conf` file, or pass it as CGI parameter.

If cache storage mode is used, you need also create SITE limit (see).

Use **PagesInGroup** command to specify the number of additional results from the same site when google-like grouping is enabled.

You may use **MaxSiteLevel** command to specify maximal domain name level using for `site_id` calculation. Default value: 2. One exception: three or less letter domains at level 2 count as domain names at level 1. For example: `domain.ext` - level 2, `www.domain.ext` - level 3, `domain.com.ext` - level 2. A negative value for **MaxSiteLevel** mean grouping performs on per directory basis, i.e. for level -1 `www.site.ext/dir1/` and `www.site.ext/dir2` group as different sites.

HlBeg and **HlEnd** commands are used to configure search results highlighting. Found words will be surrounded in those tags.

There is an **Alias** command in `search.htm`, that is similar to the one in `indexer.conf`, but it affects only search results while having no effect on indexing. See [for details](#).

R1 and **R2** specify ranges for random variables `$(R1)` and `$(R2)`.

Synonym command is used to load specified synonyms list. Synonyms file name is either absolute or relative to `/etc` directory of DataparkSearch installation.

DateFormat command is used to change Last-Modified date format output. Use `strftime` function meta-variables for your own format string.

Note: If `searchd` is used, you may specify **DateFormat** in your `searchd.conf` file, but there you should enclose this string in quotes (`"`), or pass it as CGI parameter.

Log2stderr yes/no command is used to enable error logging to `stderr`.

ResultsLimit command is used to limit maximum number of results shown. If `searchd` is used, this command may be specified in `searchd.conf`.

ResultContentType command is used to specify Content-Type header for results page. Default value: `text/html`.

Locale command is used to specify LC_ALL locale settings for search results output. Default value: `unspecified` (uses the value specified before in system settings).

8.3.3. Includes in templates

You may use `<!--INCLUDE Content="http://hostname/path"-->` to include external URLs into search results.

WARNING: You can use <!--INCLUDE> ONLY in the following template sections:

```
<!--top-->
<!--bottom-->
<!--restop-->
<!--resbot-->
<!--notfound-->
<!--error-->
```

This is an example of includes usage:

```
<!--top-->
...
<!--INCLUDE CONTENT="http://hostname/banner?query=${q}"-->
...
<!--/top-->
```

8.3.4. Conditional template operators

DataparkSearch is support conditional operators in search templates:

```
<!-- IF NAME="Content-Type" Content="application/pdf" -->

<!-- ELIF NAME="Content-Type" Content="text/plain" -->

<!-- ENDIF -->
```

It's possible to use nested conditional operators. This give much power for search template construction. See samples in `etc/search.htm-dist` file.

8.3.5. Security issues

WARNING: Since the template file contains such info as password, it is highly recommended to give the file proper permissions to protect it from reading by anyone but you and search program. Otherwise your passwords may leak.

8.4. Designing search.html

This section is assuming that you are using the CGI front end.

8.4.1. How the results page is created

The file `etc/search.htm` consists of a number of blocks delimited by HTML comments that start with `<!--comment-->` and end with `<!--/comment-->`.

The `<!--variables-->` block is only used by `search.cgi`. The other blocks form part of the results output depending on the situation.

The blocks `<!--top-->` and `<!--bottom-->` are always returned to the user as the top and bottom part of the output respectively.

There are three series of `<!--restop-->`, `<!--res-->` and `<!--resbot-->` blocks. The first series is returned to users that have requested long results (default), the second one to those that have requested short results and the third one to those that have requested results as URL only. All three blocks must be present in `search.htm`. Furthermore there is a series of navigation blocks and the blocks `<!--notfound-->`, `<!--noquery-->` and `<!--error-->`. The latter are returned occasionally instead of results.

Any HTML that is outside the pre-defined blocks in `search.htm` is completely ignored.

Thus, the output of `search.cgi` will always be something like this:

```
top
restop          top          top          top
res             or  notfound  or  error    or  noquery
```

```
resbot          bottom          bottom          bottom
(navigation)
bottom
```

The navigation part is built in the same way, with the elements that pertain to each results page. For example, <!--navleft--> and <!--navright--> are used to link to the previous and next results pages, while <!--navXXX_nop--> is used when there are no more pages in one or either direction.

8.4.2. Your HTML

The simplest HTML is provided ready for use in `etc/search.htm-dist`. It is advisable that you use this until your back-end works fine.

Once you decide to add bells and whistles to your search, you have two options. One is to keep the simple design of `search.htm`, but make it part of a frame set. This way you can add elements such as menus etc in a frame and keep the output of `search.htm` in another.

The other option is to incorporate your entire design in `search.htm`. If you fully understand the "blocks" system described above, this should not be too difficult. The one most important factor is to keep track of elements that need to be opened in one block and closed in another.

For example, you might want a page in tables that looks like this:

```

|          top   table
| .....
|          .
| left      .
|          .
|          .          main table
| table     .
|          .
|          .

```

If you are planning to put your results in the main table, you can put all the HTML code in the <!--top--> block of `search.htm`, up to and including the opening of the main table (`<table><tr><td>`). If you then put the closing of the main table and the closing tags of the page in the <!--bottom--> block (`</table></tr></td></body></html>`) and leave all other blocks unformatted, you will have the design of your choice and all your results in the right place.

In a more complicated design, where you want to format results individually, you can apply the same method as long as you keep track of the opening and closing of HTML elements. You must either open and close them in the same block, or make sure that any possible combination of blocks will result in properly opened and closed HTML tags.

What you cannot do without editing the source code, is change the order in which the blocks are parsed. Taking the above example, let's assume that you want your page to look like this:

logo	banner ads
.....	
choices	results
search	
button	

To get this, you need to have everything except the results and navigation in the `<!--top-->` block, since that is the only block that can draw the page even if there are no results at all. In this case your `search.htm` would look like this:

```
<!--variables-->
  [your configuration]
<!--/variables-->

<!--top-->
<html>
<body>
<table>
  <tr colspan="2">
    <td>[logo, banner ads]</td>
  </tr>
  <tr>
    <td>[search form]</td>
    <td>
      [all other blocks in search.htm except "bottom"]
    </td>
  </tr>
</table>
<!--/top-->

<!--bottom-->
  [closing elements like the DataparkSearch link
   and a link to the webmaster]
</td>
</tr>
</table>
</body>
</html>
<!--/bottom-->
```

The individual blocks can be formatted individually as long as that formatting is closed within each block. Thus, nothing stops you from doing things like

```

<!--error-->
<table>
<tr><td bgcolor="red">
  <font color="#ffffff">
    [error variables]
  </font>
</tr></td>
</table>
<!--error-->

```

as long as such formatting is opened and closed properly within the same block.

8.4.3. Forms considerations

Most modern browsers can handle forms that stretch over different tables, but writing such forms is against all standards and is bad HTML. Unless you really can't avoid it, don't do it.

For example,

```

<table>
<tr><td>
  <form>
    <input type="text" name="something">
    <input type="radio" name="button1">
    <input type="radio" name="button2">
  </form>
</tr></td>
</table>

```

is fine, but

```

<table>
  <tr><td>
    <form>
      <input type="text" name="something">
    </tr></td>
  </table>
<table>
  <tr><td>
    <input type="radio" name="button1">
    <input type="radio" name="button2">
  </form>
</tr></td>
</table>

```

is *not*.

Note that the input forms in `search.htm` can be changed at will. The default is drop-down menus, but nothing stops you from using radio buttons or hidden input or even text boxes. For instance, where `search.htm` says

```
Results per page:
<SELECT NAME="ps">
<OPTION VALUE="10" SELECTED="$ (ps) ">10
<OPTION VALUE="20" SELECTED="$ (ps) ">20
<OPTION VALUE="50" SELECTED="$ (ps) ">50
</SELECT>
```

you can very well substitute

```
<input type="radio" name="ps" value="10" checked="$ (ps) ">
<input type="radio" name="ps" value="20" checked="$ (ps) ">
<input type="radio" name="ps" value="50" checked="$ (ps) ">
```

which will result in three radio buttons instead of a drop-down menu, with "20" as the default and the exact same functionality. What you obviously cannot do is provide multiple-choice menus like `<type="checkbox">` or `<select multiple>`.

Note that you can also use the

```
<input type="hidden" name="XX" value="YY">
```

format if you want to set other defaults than the pre-defined and not allow the user to change them.

8.4.4. Relative links in `search.htm`

It might be worth mentioning that `search.htm` is parsed from your `cgi-bin` directory. The position of this directory in relation to your document root is determined by the web server, independently of its actual position in the file system. Almost invariably is `http://your_document_root/cgi-bin/`. Since `search.cgi` lives in `cgi-bin`, any links to images etc in `search.htm` will assume `cgi-bin` as the base directory. Therefore, if you have a file system structure like

```
home/
home/your_document_root/
home/your_document_root/img/
home/cgi-bin/
```

the correct relative link from `search.cgi` to images in `img/` would still be

```

```

despite the fact that it doesn't match the file system structure.

8.4.5. Adding Search form to other pages

To place a search form to any of your pages, please place the following code where you would like the form to be displayed:

```
<FORM
  METHOD=GET
  ACTION="http://path-to-search.cgi">
    <INPUT TYPE="text" NAME="q" VALUE="">
    <INPUT TYPE="submit" VALUE="Search!">

</FORM>
```

8.5. Relevance

8.5.1. Ordering documents

DataparkSearch by default sorts results first by relevency and second by popularity rank.

8.5.2. Relevance calculation

In indexing, DataparkSearch divides every document onto sections. A section is any part of the document, for example, for HTML documents this may be TITLE or META Description tag.

In addition to sections, some document factors are also take in account for relevance calculation: the average distance between query words, the number of query word occurrences, the position of first occurrence of a query word, the difference between the distribution of query word counts and the uniform distribution.

In searching, DataparkSearch compares every document found against an "ideal" document. The "ideal" document should have query words in every section defined and should have also the predefined values of additional factors.

Since sections definition located only in `indexer.conf` file, use **NumSections** command in `searchd.conf` or in `search.htm` to specify the number od section used. By default, this value is 256. But note, **NumSections** do not affect document ordering, only the relevance value.

Table 8-3. Configure-time parameters to tune relevance calculation (switches for configure)

<code>--enable-fullrel</code>	This option enables full version of relevance calculation. Value by default: disabled (or fast relevance calculation).
<code>--disable-reldistance</code>	This option disables accounting of average word distance for relevance calculation. Value by default: enabled.

<code>--disable-relposition</code>	This option disables accounting of first query word position for relevance calculation. Value by default: enabled.
<code>--disable-relwrdcnt</code>	This option disables accounting of word counts for relevance calculation. Value by default: enabled.
<code>--with-avgdist=NUM</code>	This option specify the NUM as the best average distance between words in document found. Value by default: 464.
<code>--with-bestpos=NUM</code>	This option specify the NUM as the best value of first word position in document found. Value by default: 4.
<code>--with-bestwrdcnt=NUM</code>	This option specify the NUM as the best number of each query word in document found. Value by default: 11.
<code>--with-distfactor=NUM</code>	This option specify the NUM as a factor for average word distance for relevance calculation. Value by default: 0.2.
<code>--with-lessdistfactor=NUM</code>	This option specify the NUM as factor of average word distance in relevance calculation when average distance is less than value specified with <code>--with-avgdist</code> . Default value is <code>--with-distancefactor</code> multiply by 2.
<code>--with-posfactor=NUM</code>	This option specify the NUM as factor for difference between first query word position in document found and best position specified by <code>--with-bestpos</code> option. Value by default: 0.5.
<code>--with-lessposfactor=NUM</code>	This option specify NUM as factor of first word position in relevance calculation when it less than value specified with <code>--with-bestpos</code> . Default value is <code>--with-posfactor</code> multiply by 4.
<code>--with-wrdcntfactor=NUM</code>	This option specify the NUM as factor for difference between count of query words in document found and the best value specified by <code>--with-bestwrdcnt</code> option. Value by default: 0.4.
<code>--with-lesswrdcntfactor=NUM</code>	This option specify NUM as factor of word count in relevance calculation when this word count is less than value specified with <code>--with-bestwrdcnt</code> . Default value is <code>--with-wrdcntfactor</code> multiply by 10.
<code>--with-wrdunifactor=NUM</code>	This option specify the NUM as factor for difference of query word counts from uniform distribution. Value by default: 1.5.

8.5.2.1. A full method of relevance calculation.

Let x is the weighted sum of all sections. The weights for these sections are define by `wf` parameter (see). Let y is the weighted sum of differences between values of additional factors of document found and corresponding values of additional factors of the "ideal" document. And let xy is the weighted sum of sections where at least one query word has been found. Then value of relevance for a document found is calculates as: $0.5 * (x + xy) / (x + y)$.

8.5.2.2. A fast method of relevance calculation.

Let x is the number of bits used in weighted values of all sections defined. Let y is the weighted sum of differences between additional factors of document found and corresponding values of the "ideal" document. And let xy is the number of bits where weighted values of sections of the "ideal" document are different to weighted values of sections of document found. Then value of document relevance is calculates as: $(x - xy) / (x + y)$.

8.5.3. Popularity rank

DataparkSearch support two methods for popularity rank calculation. A method used in previous versions called "Goo", and new method is called "Neo". By default, the `Goo` method is used. To select desired PopRank calculation method use `PopRankMethod` command:

```
PopRankMethod Neo
```

You need enable links collection by `CollectLinks yes` command in your `indexer.conf` file for Neo method and for full functionality of `Goo` method. But this slow down a bit indexing speed. By default, links collection is not enabled.

By default, only intersite links (i.e. links from a page on one site to a page on an another site) are taken in account for the popularity rank calculation. If you place `PopRankSkipSameSite no` command in `indexer.conf` file, `indexer` take all links for this purpose.

You may assign initial value for page popularity rank using `DP.PopRank META` tag (see).

8.5.3.1. "Goo" popularity rank calculation method

The popularity rank calculation is made in two stages. At first stage, the value of `Weight` parameter for every server is divide by number of links from this server. Thus, the weight of one link from this server is calculated. At second stage, for every page we find the sum of weights of all links pointed to this page. This sum is popularity rank for this page.

By default, the value of `Weight` parameter is equal to 1 for all servers indexed. You may change this value by `Weight` command in `indexer.conf` file or directly in server table, if you load servers configuration from this table.

If you place `PopRankFeedBack yes` command in `indexer.conf` file, `indexer` will calculate site weights before page rank calculation. To do that, `indexer` calculate sum of popularity rank for all pages

from same site. If this sum will great 1, the weight for site set to this sum, otherwise, site weight is set to 1.

If you place `PopRankUseTracking yes` command in `indexer.conf` file, **indexer** will calculate site weight as the number of tracked queries with restriction on this site.

If you place `PopRankUseShowCnt yes` command in `search.htm` (or `searchd.conf`) file, then for every result shown to user corresponding `url.shows` value will be increased on 1, if relevance for this result is great or equal to value specified by `PopRankShowCntRatio` command (default value is 25.0). If you place `PopRankUseShowCnt yes` in `indexer.conf` file, **indexer** will add to url's PopularityRank the value of `url.shows` multiplied by value, specified in `PopRankShowCntWeight` command (default value is 0.01).

8.5.3.2. "Neo" popularity rank calculation method

For this method is supposed all pages are neurons and links between pages are links between neurons. So it's possible use an error back-propagation algorithm to train this neural network. Popularity rank for a page is the activity level for corresponding neuron. See short description of The Neo Popularity Rank for web pages (<http://www.maxime.net.ru/doc/Neo-en.pdf>).

You may use `PopRankNeoIterations` command to specify the number of iterations of the Neo Popularity Rank calculation. Default value is 3.

By default, the Neo Popularity Rank is caclulated along with indexing. To speed up indexing, you may postpone Popularity Rank execution using **PopRankPostpone** command:

```
PopRankPostpone yes
```

Then you may calculate the Neo Popularity Rank after indexing in same way as for method Goo, i.e.:
indexer -TR

8.5.4. Boolean search

Please note that in case of boolean searching for two or more words, you have to enter operators (&, |, ~, AND, OR, NOT, NEAR, ALL, etc.). I.e. it is necessary to enter **a & book** instead of **a book**. See also .

8.5.5. Crosswords

This feature allows to assign words between `<a href="xxx">` and `</a>` also to a document this link leads to. To enable Crosswords, please use **CrossWords yes** command in `indexer.conf` and `search.htm`, and define **crosswords** section in `sections.conf` file.

8.5.6. The Summary Extraction Algorithm (SEA)

The **Summary Extraction Algorithm (SEA)** builds the summary of three or more the most relevant sentences of the each document indexed, if this document consists of six or more sentences. To enable this feature, add this command to your `seaction.conf` file:

```
Section sea x y
```

where `x` - the number of section and `y` - the maximum length of this section value, leave 0, if you do not want show this in result pages. If you specify `y` non-zero, you may use `$(sea)` meta-variable in your search template to show the summary in result pages.

Related configuration directives:

The **SEASentenceMinLength** command specify the minimal length of sentence to be used in summary construction using the SEA. Default value: 64.

The **SEASentences** command uses to specify the maximal number of sentences with length greater or equal to the value specified by the **SEASentenceMinLength** command, which are used for summary construction in the SEA. Default value: 32. Since the summary construction using SEA is nonlinear expensive (affects only indexing), you may adjust this value according to desired indexing performance.

This algorithm of automatic summary construction is based on ideas of Rada Mihalcea described in the paper Rada Mihalcea and Paul Tarau, An Algorithm for Language Independent Single and Multiple Document Summarization, in Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP), Korea, October 2005 (<http://www.cs.unt.edu/~rada/papers/mihalcea.ijcnlp05.pdf>).

Differences in DataparkSearch's SEA:

- Initial weights for graph edges are calculates as a measure of similarity between 3-gram distributions of corresponding sentences.
- All initial values for graph vertexes are equal to some initial value ($1 / (\text{number of sentences} + 1)$ in current implementtion).
- The Neo PopRank algorithm is used as ranking algorithm to iterate values assigned to vertexes.

After indexing of document collection with this section defined, you may use `$(sea)` meta-variable in your template to show summary for a search result.

8.6. Search queries tracking

DataparkSearch supports query tracking.

When doing a search, front-end uses table `qtrack` to store query words, client IP address, a number of found documents and current UNIX timestamp in seconds and table `qinfo` to store all search parameters.

To enable tracking, add the `trackquery` parameter to **DBAddr** command (see) in your search template. For example:

```
DBAddr postgres://user:pass@localhost/search/?dbmode=cache&trackquery
```

Note: If you use `searchd`, you should add this parameter only in your `searchd.conf` file.

You may use **TrackDBAddr** command to specify different database to store query tracking data. This database should have the same `qtrack` and `qinfo` tables as in `DataparkSearch`'s database.

Query tracking is useful to have a statistics of your search engine usage. To make a search queries summary, you may execute, for example, this SQL expression:

```
SELECT qwords, count(*), sum(found), avg(found) FROM qtrack GROUP BY qwords;
```

8.7. Search results cache

Search results cache allows `search.cgi` to make very fast response on recently used queries as well as user's navigation through the pages of the same result.

Search results cache is disabled by default. You may use **Cache yes** command in `search.htm` to enable results caching. If you use `searchd`, add "**Cache yes**" command to `searchd.conf` file.

Search cache is located in `$PREFIX/var/cache/` subdirectory, where `$PREFIX` is `DataparkSearch` installation base directory. Each result is stored in a separate file.

By defaults, search results cache is not deleted automatically. You have to delete it every time after indexer's work to avoid displaying non-recent cached results. Or you may specify a refresh period for search results cache using **HoldCache** command:

```
HoldCache <time>
```

For `<time>` format see description of **Period** command in .

```
HoldCache 3h
```

8.8. Fuzzy search

8.8.1. Ispell

When `DataparkSearch` is used with ispell support enabled, it automatically extend search query by all grammatical forms of the query words. E.g. search front-end will try to find the word "test" if "testing" or "tests" is given in search query.

8.8.1.1. Two types of ispell files

DataparkSearch understands two types of ispell files: **affixes** and **dictionaries**. Ispell affixes file contains rules for words and has approximately the following format:

```
Flag V:
    E    > -E, IVE      # As in create> creative
    [^E] > IVE          # As in prevent > preventive
Flag *N:
    E    > -E, ION      # As in create > creation
    Y    > -Y, ICATION  # As in multiply > multiplication
    [^EY] > EN          # As in fall > fallen
```

Ispell dictionary file contains words themselves and has the following format:

```
wop/S
word/DGJMS
wordage/S
wordbook
wordily
wordless/P
```

8.8.1.2. Using Ispell

To make DataparkSearch support ispell you must specify **Affix** and **Spell** commands in `search.htm` file. The format of commands:

```
Affix [lang] [charset] [ispell affixes file name]
Spell [lang] [charset] [ispell dictionary filename]
```

The first parameter of both commands is two letters language abbreviation. The second is ispell files charset. The third one is filename. File names are relative to DataparkSearch `/etc` directory. Absolute paths can be also specified.

Note: Simultaneous loading of several languages is supported, e.g.:

```
Affix en iso-8859-1 en.aff
Spell en iso-8859-1 en.dict
Affix de iso-8859-1 de.aff
Spell de iso-8859-1 de.dict
```

Will load support for both English and German languages.

If you use `searchd`, add the same commands to `searchd.conf`.

When DataparkSearch is used with ispell support it is recommended to use `searchd`, especially for several languages support. Otherwise the starting time of `search.cgi` increases.

8.8.1.3. Customizing dictionary

It is possible that several rare words are found in your site which are not in ispell dictionaries. In such case, an entry with longest match suffix is taking to produce word forms.

But you can also create the list of such words in plain text file with the following format (one word per line):

```
rare.dict:
-----
webmaster
intranet
.....
www
http
-----
```

You may also use ispell flags in this file (for ispell flags refer to ISpell documentation). This will allow not writing the same word with different endings to the rare words file, for example "webmaster" and "webmasters". You may choose the word which has the same changing rules from existing ispell dictionary and just to copy flags from it. For example, English dictionary has this line:

```
postmaster/MS
```

So, webmaster with MS flags will be probably OK:

```
webmaster/MS
```

Then copy this file to `/etc` directory of DataparkSearch and add this file by **Spell** command in DataparkSearch configuration:

During next reindexing using of all documents new words will be considered as words with correct spelling. The only really incorrect words will remain.

8.8.1.4. Where to get Ispell files

You may find ispell files for many of languages at this page (<http://fmg-www.cs.ucla.edu/geoff/ispell-dictionaries.html>).

For Japanese language there exist quasi-ispell files suitable for use with DataparkSearch only. You may get this data from our web site (<http://www.dataparksearch.org/>) or from one of our mirrors. See .

8.8.2. Aspell

With Aspell support compiled, it's possible automatically extend search query by spelling suggestions for query words. To enable this feature, you need to install Aspell (<http://aspell.net/>) at your system before DataparkSearch build. Then you need to place **AspellExtensions yes** command into your `indexer.conf` and `search.htm` (or into `searchd.conf`, if `searchd` is used) files to activate this feature.

Automatically spelling suggestion for search query words is going only if `sp` search parameter is set, see .

8.8.3. Synonyms

DataparkSearch also support a synonyms-based fuzzy search.

Synonyms files are installed into `etc/synonym` subdirectory of DataparkSearch installation. Large synonyms files you need to download separately from our web site, or from one of our mirrors, see .

To enable synonyms, add to `search.htm` search template commands like `Synonym <filename>`, e.g.:

```
Synonym synonym/english.syn
Synonym synonym/russian.syn
```

Filenames are relative to `etc` directory of DataparkSearch installation or absolute if begin with `/`

If you use `searchd`, add the same commands to `searchd.conf`.

You may create your own synonyms lists. As an example you may take the English synonyms file. In the beginning of the list please specify the following two commands:

```
Language: en
Charset:  us-ascii
```

- Language - standard (ISO 639) two-letter language abbreviation.
- Charset - any charset supported by DataparkSearch (see).

Optionally you may specify following command in the list:

```
Thesaurus: yes
```

This command enable thesaurus mode for synonyms list. For this mode, only words at one line treats as synonyms.

8.8.4. Accent insensitive search

Since version 4.17 DataparkSearch also support an accent insensitive search.

To enable this extension, use **AccentExtensions** command in your `search.htm` (or in `searchd.conf`, if `searchd` is used) to make automatically accent-free copies for query words, and in your `indexer.conf` config file to produce accent-free word's copies to store in database.

```
AccentExtensions yes
```

If **AccentExtensions** command is placed before **Spell** and **Affix** commands, accent-free copies for those data also will be loaded automaticaly.

8.8.5. Acronyms and abbreviations

Since version 4.30 DataparkSearch also support an acronyms and abbreviation based fuzzy search.

Acronyms files are installed into `etc/acronym` subdirectory of DataparkSearch installation.

To enable acronyms, add to `search.htm` search template commands like `Acronym <filename>`, e.g.:

```
Acronym acronym/en.fido.acr
Acronym acronym/en.acr
```

Filenames are relative to `etc` directory of DataparkSearch installation or absolute if begin with `/`

If you use `searchd`, add the same commands to `searchd.conf`.

You may create your own acronyms lists. As an example you may take the English acronyms file. In the beginning of the list please specify the following two commands:

```
Language: en
Charset:  us-ascii
```

- Language - standard (ISO 639) two-letter language abbreviation.
- Charset - any charset supported by DataparkSearch (see).

Please send your own acronym files to `<maxime@maxime.net.ru>`, if you want share its with other users.

Chapter 9. Miscellaneous

9.1. Reporting bugs

When reporting bugs, please specify DataparkSearch version and provide us as much information about your problem as possible. Such information as your platform and OS details, database version, database statistics like number of URLs in database or probably count of records in different tables would be very helpful to find and fix possible bugs.

Please, submit bug reports using our Bug Reporting System (<http://www.dataparksearch.org/cgi-bin/bt.pl>) at DataparkSearch web site (<http://www.dataparksearch.org/>). Please do not send reports to mailing list or to personal authors addresses!

9.1.1. Core dump reports

If `indexer` or `search.cgi` die during their work and produce core, it would be very helpful to send us `gdb` (The GNU Debugger) output. To do this, please make the following steps. E.g. your binary is "indexer" and core file name is `indexer.core` (or may be just `core` on some platforms).

Run GNU Debugger with executable as the first argument and with core file as the second:

```
gdb indexer indexer.core
```

Some information about the crash location will appear:

```
Core was generated by `indexer'.
Program terminated with signal 8, Floating point exception.
Reading symbols from /usr/lib/libc.so.3...done.
Reading symbols from /usr/libexec/ld-elf.so.1...done.
#0  0x80483f3 in main () at indexer.c:4
4      printf("%d",0/0);
```

Then type `thread apply all backtrace` command:

```
(gdb) thread apply all bt
#0  0x80483f3 in main () at indexer.c:4
#1  0x804837d in _start ()
```

Send us either first and second outputs or just a screenshot of `gdb` session.

9.2. Using libdpsearch library

The `libdpsearch` is available for using it in third party applications. You can easily add search into your own application using `library` and include files installed in `/lib` and `/include` DataparkSearch directories. Each application which uses `libdpsearch` must have `dpsearch.h` header file included.

9.2.1. dps-config script

When compiled with one of supported SQL back-end, `libdpsearch` requires some dependent libraries, for example `libmysqlclient`. You can find `dps-config` script in `/bin` directory of DataparkSearch installation. This script helps to take in account required dependencies. `dps-config` script can take several options in it's command line. By default `dps-config` outputs all available options:

```
Usage: ./dps-config [OPTIONS]
Options:
    [--version]
    [--libs]
    [--cflags]
```

When executed with `--libs` command line option `dps-config` outputs all required to `libdpsearch` linker flags, for example:

```
# ./dps-config --libs
-lm -L/usr/local/mysql/lib/mysql -lmysqlclient \
-L/usr/local/dpsearch/lib -ldpsearch
```

So you may insert `dps-config --libs` into CC compiler command line:

```
cc myprog.c -o myprog `dps-config --libs`
```

9.2.2. DataparkSearch API

There is no detailed description of DataparkSearch API yet. This is because API is currently under rapid development and may have major changes from version to version. You may use `search.c` as an example of application which uses `libdpsearch` library.

9.3. Database schema

Full database schema used by DataparkSearch is defined in appropriate sql-scripts for database creation located under `create` subdirectory.

Table 9-1. `server` table schema

<code>rec_id</code>	Unique record identifier.
<code>enabled</code>	A flag to enable/disable record for indexer .
<code>url</code>	URL or pattern.
<code>tag</code>	Tag value.
<code>category</code>	Categories table <code>rec_id</code> .
<code>command</code>	=S - this record is a server. =F - this record is a filter.
<code>ordre</code>	Sorting key, it define records order for <code>server</code> table loading.
<code>parent</code>	If not null, this record is added automatically by indexer and <code>url</code> field contain a server name accepted on record pointed by this filed value.
<code>weight</code>	This record weight for PopRank calculation.
<code>pop_weight</code>	One link weight from pages of this server. Calculated automatically. Manually change will have no effect.

Other server's parameters store in `srvinfo` table. Possible values for several parameters is given in table below.

Table 9-2. Several server's parameters values in `srvinfo` table

sname value	Possible sval values.
<code>Alias</code>	Alias used for <code>url</code> .
<code>Period</code>	Reindexing period in seconds.
<code>DeleteOlder</code>	How much time to hold URLs before deleting them from the database.
<code>RemoteCharset</code>	Default charset value.
<code>DefaultLang</code>	Default language value.
<code>Request.Authorization</code>	For basic authorization.
<code>Request.Proxy</code>	Proxy server to access documents from this resource.
<code>Request.Proxy-Authorization</code>	Proxy server authorization.

sname value	Possible sval values.
MaxHops	Maximum depth of way in "mouse" clicks from start url.
Index	A flag to enable/disable documents indexing.
Follow	=0, "page" =1, "path" =2, "site" =3, "world"
Robots	A flag to enable/disable robots.txt file using.
DetectClones	A flag to enable/disable "clones" detection.
MaxNetErrors	Maximum network errors for this server.
NetDelayTime	Indexing delay time if a network error is occurred.
ReadTimeout	Network timeout value.
match_type	=0, DPS_MATCH_FULL - full coincidence. =1, DPS_MATCH_BEGIN - pattern is a URL prefix. =2, DPS_MATCH_SUBSTR - pattern is a URL substring. =3, DPS_MATCH_END - pattern is a URL suffix. =4, DPS_MATCH_REGEX - pattern is a regular expression. =5, DPS_MATCH_WILD - pattern is a wildcards pattern (* and ? wildcards may be used). =6, DPS_MATCH_SUBNET - < not yet supported >.
case_sense	=1, - case insensitive comparison. =0, - case sensitive comparison.
nomatch	=1, - URLs not match this record is accepted. =0, - URL match this record is accepted.

sname value	Possible sval values.
Method	Specify a document action for this command. =Allow, - all corresponding documents will be indexed and scanned for new links. =Disallow, - all corresponding documents will be ignored and deleted from database. =HrefOnly, - all corresponding documents will be only scanned for new links (not indexed). =CheckOnly, - all corresponding documents will be requested by HTTP HEAD request, not HTTP GET, i.e. inly brief info about documents (size, last modified, content type) will be fetched. =Skip, - all corresponding documents will be skipped while indexing. =CheckMP3, - all corresponding documents will be checked for MP3 tags along if its Content-Type is equal to audio/mpeg. =CheckMP3Only, - is equal to CheckMP3, but if MP3 tag is not present, processing on Content-Type will not be taken. =TagIf, - all documents will be maked by tag specified. =CategoryIf, - all documents will be maked by category specified. =IndexIf, - all documents will be indexed, if the value of section specified match the pattern given. =NoIndexIf, - all documents will be ignored and deleted from database, if the value of section specified match the pattern given.
Section	Section name used in pattern matching for IndexIf and NotIndexIf methods.

Appendix A. Donations

If you like the DataparkSearch Engine and want to encourage further development, feel free to make a donation (at Kagi) (<http://order.kagi.com/?6CYPQ>) to support this project. Any donation is greatly appreciated.

The following individuals have made donations to support DataparkSearch development, and deserve credit for it:

- Many people prefer not to be listed, and the details of many others were not recorded (If anyone wishes to send details of their donation, we would be very happy to include them).

Index

- Accent insensitive search, ??
- Acronyms and abbreviations, ??
- Aspell, ??
- Authors, 3
- Boolean search
 - advanced, ??
 - relevance, 118
- Bugs
 - reporting, ??
- Categories, 75
- Changelog latest, 1
- Charsets, 78
- Clear database, 12
- Clones, 16
- Command
 - AccentExtensions, ??
 - Acronym, ??
 - ActionSQL, 53
 - AddType, 27, 38
 - Affix, ??
 - Alias, 22, 25, ??
 - AliasProg, 24
 - Allow, 34
 - AspellExtensions, ??
 - AuthBasic, 13, 44
 - Bind, 45
 - BodyPattern, 60
 - BrowserCharset, ??, 108
 - Cache, ??
 - Category, 76
 - CategoryIf, 77
 - CharsToEscape, ??
 - CheckInsertSQL, 67
 - CheckMp3, 36
 - CheckMp3Only, 36
 - CheckOnly, 35
 - ColdVar, ??
 - CollectLinks, 117
 - Cookies, 41
 - CrawlDelay, 42
 - CrossWords, 118
 - DataparkSearchdConf, ??
 - DataparkSearchTemplate, ??
 - DataparkStoredocTemplate, ??
 - DateFormat, 109

DBAddr, 31, ??
DefaultLang, 84
DeleteOlder, 37
DetectClones, 16, ??
Disallow, 35
DocTimeOut, 41
DoExcerpt, 55
DoStore, 55
ExcerptSize, 55
ExpireAt, 39
FastHrefCheck, 43
FlushServerTable, 26
ForceIISCharset1251, ??
GroupBySite, 108
GuesserBytes, ??
GuesserUseMeta, ??
HIBeg, 57, ??
HIEnd, 57, ??
HoldBadHrefs, 37
HoldCache, ??
HrefOnly, 36
HrefSection, 43
HTDBAddr, 46
HTDBDoc, 46
HTDBLimit, 46
HTDBList, 46
HTDBText, 47
HTTPHeader, 34
Include, 31
Index, 43
IndexDocSizeLimit, 33
IndexIf, 37
LangMapFile, ??
LangMapUpdate, ??
Limit, 65
Listen, 55
LoadChineseList, 88
LoadKoreanList, ??
LoadThaiList, 88
LocalCharset, ??, ??
Locale, ??
Log2stderr, ??
LogLevel, 53
LogsOnly, 63
LongestTextItems, 45
MarkForIndex, 66
MaxClients, 69
MaxDepth, 40

MaxDocSize, 33
MaxDocsPerServer, 40
MaxHops, 39
MaxNetErrors, 40
MaxSiteLevel, 108
MaxWordLength, 33
Mime, 27
MinDocSize, 33
MinWordLength, 33
MirrorHeadersRoot, 52
MirrorPeriod, 52
MirrorRoot, 52
NetErrorDelayTime, 41
NewsExtensions, 32
NoIndexIf, 37
NoStore, 55
NumSections, ??
OptimizeAtUpdate, 45
OptimizeInterval, 56
OptimizeRatio, 56
PagesInGroup, ??
ParserTimeOut, 27
Period, 11, 38
PeriodByHops, 39
PopRankFeedBack, ??
PopRankMethod, ??
PopRankNeoIterations, ??
PopRankPostpone, 118
PopRankShowCntRatio, ??
PopRankShowCntWeight, 118
PopRankSkipSameSite, ??
PopRankUseShowCnt, ??
PopRankUseTracking, ??
PreloadLimit, 69
PreloadURLData, 68
ProvideReferer, 45
Proxy, 44
ProxyAuthBasic, 43
R*, ??
ReadTimeOut, 41
Realm, 19
RealmDB, 21
RemoteCharset, ??, ??
ResegmentChinese, 87
ResegmentJapanese, ??
ResegmentKorean, ??
ResegmentThai, 87
ResultContentType, 109

- ResultsLimit, ??
- ReverseAlias, 24
- Robots, 42
- RobotsPeriod, 42
- SEASentenceMinLength, ??
- SEASentences, ??
- Section, 42
- Server, 19
- ServerDB, 21
- ServerTable, 25
- ServerWeight, 44
- SkipUnreferred, 45
- Spell, ??
- StopwordFile, 16
- Store, 55
- StoredFiles, 56
- SubDocCnt, 60
- SubDocLevel, 60
- Subnet, 20
- SubnetDB, 21
- Synonym, ??, ??
- SyslogFacility, 33
- Tag, 74, 74
- TagIf, 74
- TrackDBAddr, ??
- TrackHops, 40
- URL, 21
- URLCharset, ??
- URLDataFiles, 62
- URLDB, 21
- URLDumpCacheSize, 34
- URLInfoSQL, 66
- URLSelectCacheSize, 34
- UseCRC32URLId, 34
- UseDateHeader, 39
- UseRemoteContentType, 38
- VarDir, 32, 55, ??
- VaryLang, 89
- Weight, 117
- WrdFiles, 62
- Content-Encoding, 15
- Contributors, 3
- Creating SQL table structure, 11
- Crosswords, ??
- Data acquisition, 53
- Database schema, ??
- Database statistics, 12
- Disclaimer of DataparkSearch, 3

- DMALLOC, 8
- Document
 - excerpts, 55
 - summary, ??
- Donations, ??
- dpconv, 83
- dpguesser, 83
- DPS_URL environment variable, 28
- Dropping SQL table structure, 12
- Features, 1
- HTDB
 - Indexing SQL database tables, 46
 - variables, 47
- Indexing binaries output, 50
- Installation
 - problems, 9
 - requirements, 5
 - steps, 7
- Introduction, 1
- Ispell, ??
- libdpsearch, 126
- Link validation, 13
- META
 - Content-Type, 58
 - Description, 59
 - DP.PopRank, 59
 - Keywords, 59
 - Refresh, 58
 - Robots, 59
- META tags, 58
- Mirroring, 52
- mod_dpsearch, ??
- Multi-language, ??
- Oracle
 - notes, 69
- Parallel indexing, 13
- Parameter method
 - Allow, 17
 - CheckMP3, 18
 - CheckMP3Only, 18
 - CheckOnly, 17
 - Disallow, 17
 - HrefOnly, 17
 - Skip, 17
- Parsers, 26
 - Charsets, 28
 - third-party, 28
- Performance issues, 66

- Phrase segmenter
 - Chinese language, ??
 - Japanese language, ??
 - Korean language, ??
 - Thai language, ??
- Popularity rank, ??
- Relevance, ??
 - fast method, 117
 - full method, ??
- SEA
 - The Summary Extraction Algorithm, ??
- Search parameters, ??
- Search results cache, 120
- searchd, 68
- Stopwords, 16
- Storage modes
 - cache mode, 62
 - crc mode, 61
 - crc-multi mode, 62
 - multi mode, 61
 - single mode, 61
- stored, 55
- Summary Extraction Algorithm, The. ??
- Synonyms, 123
- Syslog, 53
- Tag parser, 58
- Tags, 74
- Template section
 - BOTTOM, 102
 - CLONE, 105
 - error, ??
 - navbar0, ??
 - navbar1, ??
 - navright, navright_nop, ??
 - noquery, ??
 - notfound, 106
 - RES, ??
 - RESBOT, ??
 - RESTOP, ??
 - TOP, ??
- Template variable
 - \$(Body), ??
 - \$(BrowserCharset), ??
 - \$(CL), ??
 - \$(Content-Length), ??
 - \$(Content-Type), ??
 - \$(DY), ??
 - \$(E), ??

- \$(FancySize), 104
- \$(first), 102
- \$(last), ??
- \$(Last-Modified), ??
- \$(meta.description), ??
- \$(meta.keywords), ??
- \$(ndocs), 103
- \$(Order), ??
- \$(PerSite), ??
- \$(Pos), ??
- \$(Score), ??
- \$(sea), ??
- \$(SearchTime), ??
- \$(Title), ??
- \$(total), 103
- \$(URL), ??
- \$(W), ??
- \$(WE), 103
- \$(WS), ??
- Templates, ??
 - sections, ??
 - Variables section, ??
- tracking of search queries, ??
- Where to get DataparkSearch, 3