



# Indexing For Intranet Thai Search Engine

by

Mr. Veerasak Athornchaikul

Submitted in Partial Fulfillment of  
the Requirements for the Degree of  
Master of Science  
in Information Technology  
Assumption University

May, 2001

198020

# **Indexing For Intranet Thai Search Engine**

by  
**Mr. Veerasak Athornchaikul**

**Submitted in Partial Fulfillment of  
the Requirements for the Degree of  
Master of Science  
in Information Technology  
Assumption University**

May, 2001

# The Faculty of Science and Technology

## Thesis Approval

Thesis Title                      Indexing for Intranet Thai Search Engine

By                                      Mr. Veerasak Athornchaikul

Thesis Advisor                      Dr. Wisanu Tuntawiroon


Academic Year                      3/2000

---


The Department of Information Technology, Faculty of Science and Technology of Assumption University has approved this final report of the **twelve** credits course. **IT7000 Master Thesis**, submitted in partial fulfillment of the requirements for the degree of Master of Science in Information Technology.

Approval Committee:

  
-----  
(Dr. Wisanu Tuntawiroon)  
Advisor

  
-----  
(Dr. Naruetep Choakjarernwanit)  
Committee Member


  
-----  
(Dr. Thatchai Chuenchom)  
Committee Member

  
-----  
(Asst. Prof. Dr. Somnuk Keretho)  
Representative of Ministry of  
University Affairs

---

Faculty Approval:

  
-----  
(Asst. Prof. Dr. Thotsapon Sortrakul)  
Director

  
-----  
(Asst. Prof. Dr. Pratit Santiprabhob)  
Dean

May / 2001

## ACKNOWLEDGEMENTS

The Author would like to thank my advisor, Dr. Wisanu Tuntawiroon who devoted his valuable time to give precious guidance, and constant encouragement; my committees Dr. Naruethep Chokcharoenwatana and Dr. Thatchai Chuenchom for their valuable comment and suggestions. I also would like to thank everyone at Planet Communications Asia Co., Ltd. who help me test this search engine and one of my friends who get me any ideas for coding of this program.

Finally, thank my family for their continual and vocal encouragement for me to finish this thesis. Perhaps I should also acknowledge their encouragement for me not to start another one.



## ABSTRACT

In the world of Internet, World Wide Web (WWW) becomes more popular for everyone. With the huge amount of information they provide, WWW are rapidly growing. A search engine is an effective tool used for searching and locating information from the huge network (Internet). However, a search engine couldn't give every piece of the information users need because of its manual working process. The search engine has to find the information of each web page manually. As a result, the information in the database isn't really up to date. Search Engine needs to be improved to effectively retrieve information for each web page and automatically generate a database.

In this thesis, we would like to propose a new generation of Thai search engine for organization. The outstanding thing about this search engine is that it has an automatic tool to retrieve information for each web page and generate a database. This search engine uses the relationship between the keyword and URL to increase speed in searching and uses ranking to improve the quality of the search's result. Consequently, this search engine can produce fast and good results because we use the hit of rank in each web page and indexing of each keyword to be the factors of the searching method. We applied selected nature of human to rank the URL and create a relationship between the keyword and URL. Each time a URL is selected, the ranking is increased.

In conclusion, this new search engine will give us a new way of searching for the information on the World Wide Web. More popular websites will be shown on the high rank of the database. However, the old version search engine using cut text software might not cover every index, if it has less database. This new search engine saves us time and energy to search the information. We will receive more accurate and broader information of what we are looking for. At the same time, the database will be automatically updated as well.

## TABLE OF CONTENTS

ACKNOWLEDGEMENTS	i
ABSTRACT	ii
LIST OF FIGURES	vi
LIST OF TABLES	viii
CHAPTER 1 INTRODUCTION	1
1.1 Introduction	1
1.2 Objective and scope of thesis	2
1.3 Outline of this thesis	3
CHAPTER 2 OVERVIEW OF SEARCH ENGINE	4
2.1 What is Search Engine	4
2.1.1 Spider	5
2.1.2 Index	5
2.1.3 Search Engine Mechanism	6
2.2 The differences among Search Engines, Directories, and Hybrid Search Engines	6
2.2.1 Search Engines	6
2.2.2 Directories	6
2.2.3 Hybrid Search Engines	6
2.3 Search Engine Method of Text Searching	7
2.3.1 Keyword Searching	7
2.3.2 The Problem with Keyword Searching	8
2.3.3 Concept-Based Searching	9
2.4 How Search Engines Rank Web Pages	9
2.5 Major Search Engine	12



2.6 Compare Search Engine Features	14
2.6.1 Search Engine Math Command	14
2.6.2 Power Searching Command	15
2.6.3 Search Assistance Features	16
2.6.4 Display Features	17
2.6.5 Boolean Command	18
2.6.6 Search Engine Size	19
2.7 Literature Review	19
2.8 State of problem	27
CHAPTER 3 THE PROPOSED METHOD	29
3.1 Introduction of The Proposed Method	29
3.2 The Context Diagram of The Proposed Method	30
3.2.1 Level 1 of Process 1: Search data from database	32
3.2.2 Level 1 of Process 3: Update Database	34
3.2.3 Level 1 of Process 4: Delete Unused Database	36
3.2.4 Level 1 of Process 5: Gather information from web server in organization	37
3.3 ER Diagram of The Proposed Method	40
3.4 Pre-Processing of the Proposed Search Engine	40
CHAPTER 4 DATA ANALYSIS	44
4.1 Theory of searching's time with my Proposed Search Engine	44
4.2 Comparison time for searching between Proposed Search Engine and a Search Engine that uses only URL database for searching	46
4.3 Comparison the position of searching's result between Proposed and a Search Engine that uses only URL database for searching	49

4.4 Experiment with the Proposed Search Engine in the organization	51
4.5 Comparison main features between the Proposed Search Engine and other Intranet Search Engines	54
4.6 Conclusion from these experiments	55
CHAPTER 5 CONCLUSION AND FUTUREWORK	56
5.1 Conclusion	56
5.2 Future work	56
BIBLIOGRAPHY	58
APPENDIX A: QUESTIONNAIRE	60
APPENDIX B: DATA DICTIONARY	64
APPENDIX C: NETWORK DIAGRAM	66
APPENDIX D: SEARCH ENGINE AND SPIDER MANUAL	67
APPENDIX E: SPIDER'S SOURCE CODE	71
APPENDIX F: SEARCH ENGINE MECHANISM'S SOURCE CODE	132



## LIST OF FIGURES

Figure 2-1: Compare Search Engine Size between major Search Engine	19
Figure 3-1: Context Diagram of The Proposed Method	31
Figure 3-2: Level 1 of Process 1: Search data from database	32
Figure 3-3: Level 1 of Process 3: Update Database	35
Figure 3-4: Level 1 of Process 4: Delete Unused Database	36
Figure 3-5: Level 1 of Process 5: Gather information from web server in organization	39
Figure 3-6: ER Diagram of The Proposed Method	40
Figure 3-7: Pre-Processing of creating index	43
Figure 4-1: Searching time from theoretical formula	46
Figure 4-2: Comparison searching's time between Search Engine that use only URL database for searching and Proposed Search Engine	48
Figure 4-3: Comparison the position of searching's result in each time (in the case of the first page)	49
Figure 4-4: Comparison the position of searching's result in each time (in the case of the last page)	50
Figure 4-5: Comparison the position of searching's result in each time (in the case of unrecognized page)	51
Figure 4-6: Graph showing the number of time accessed and total average accesss time	53
Figure C-1: Network diagram for Testing Search Engine	66
Figure D-1: Input Screen of The Proposed Search Engine	67
Figure D-2: Display Screen of The Proposed Search Engine	68
Figure D-3: Display Screen of The Proposed Search Engine (show page number)	69



## LIST OF TABLES

Table 2-1: Compare Search Engine Math Command between major Search Engines	15
Table 2-2: Compare Power Searching Command between major Search Engines	16
Table 2-3: Compare Search Assistance Features between major Search Engines	17
Table 2-4: Compare Display Features between major Search Engines	18
Table 2-5: Compare Boolean Command between major Search Engines	19
Table 4-1: Searching time from theoretical formula	45
Table 4-2: Searching time of the Search Engine that uses only URL database for searching	47
Table 4-3: Searching time of the Proposed Search Engine	47
Table 4-4: Number of accesses and total average access time for each user	53
Table 4-5: Comparison main features between the Proposed Search Engine and other Intranet Search Engine	54
Table A-1: Information of Questionnaire	63
Table B-1: Data Dictionary of Keyword Table	64
Table B-2: Data Dictionary of Related Table	64
Table B-3: Data Dictionary of URL Table	65

# CHAPTER 1

## INTRODUCTION

### 1.1 Introduction

Nowadays in the rapidly growing of WWW, there are a large number of web sites increasing by the minute. These web sites provide the varieties of information. To locate and search the information on web sites, one needs the effective tool that is called “Search Engine” [1].

Among Web site searching services, or so called search engine, presently available are AltaVista, Lycos, Yahoo, Hotbot, Infoseek etc. Some of them consist of a real search engine mechanism while others keep their web page listing in their directory. Both types are different when you consider their web page index creating basis.

Directory basis is different from the original search engine in the way that adjusting or adding information must be managed by a system administrator, not automatically recognized or managed by the system. For directory basis, a web site owner who would like his site to appear in the directory has to contact the directory administrator. Then the administrator will categorize and store the web site in the database. A directory storage system may provide the more appropriate information for a user than the original search engine. It is because the administrator has previously categorized the information already. Yahoo is an example of the directory searching service systems.

According to a search engine in its original meaning, Hotbot, AltaVista and Infoseek for example, there should be a software system containing a robot program, which

automatically read web pages from any sites then create indexes. After that the search engine will verify any links in each web page and generate a sub index for the page afterwards.

According to Internet user needs nowadays, searching for Thai web pages becomes more important, especially in an organization, which has its own Intranet with web service available. In these circumstances, a search engine, which supports bilingual use, may be required to query both Thai and English information. For example, both Catcha and Siamguru were designed to work by using a directory basis in order to serve all users nationwide. Creating a Thai index is particularly difficult because one Thai word can have several meanings. This is the critical point for a Thai-support search engine.

In this thesis, I will first describe how the Search Engine works, and then develop the new method for storing the keywords, use the searching rank of each web site in improving the speed of searching, implement the automatic tool that is used for retrieving information for each web page, and also develop the Thai Search Engine in an organization which has its own Intranet.

### 1.2 Objective and scope of thesis

The objective of this work is to understand about a search engine, how it works, study how a spider works and gather information for a web page, design the new database for a search engine, especially the methodologies used to improve speed in searching and user's satisfaction with the search result by gathering the web pages that the user selected. Moreover, we implement a Thai search engine that uses organization to better understand this proposal.

### 1.3 Outline of this thesis

This thesis is organized as follows

*Chapter 1:* First is the introduction objective and scope of thesis.

*Chapter 2:* First is an overview of a search engine, what is search engine is, the differences between search engines, directories, and hybrid search engines, search engine methods of text searching, how search engines rank web pages, major search engines, literature review, and state of problems.

*Chapter 3:* This chapter describes about the Proposed Search Engine.

*Chapter 4:* This chapter provides the experiment with the Proposed Search Engine.

*Chapter 5:* Summarize the results and the future work.





## CHAPTER 2

### OVERVIEW OF SEARCH ENGINE

#### 2.1 What is Search Engine?

**Definition:** A Search Engine is an interactive tool that enables users to locate information available via the World Wide Web. Search engines provide a “fill-out” form and other interfaces so the user can type in a query, submit the request, and retrieve a list of resources that match the search criteria. The hypertext environment makes it possible to offer a link directly from the list of results to the resources themselves. A single search engine cannot cover every available web resource, but many do contain references to millions of resources. Results may vary from one search engine to the next.

A Search Engine uses automated tools and programs to gather resources. These tools often referred to as wanderers, worms, spiders, crawlers, and robots, search thousands of web sites worldwide, collect information, and store the information in the database. The index of a search engine's database is created from the collected information e.g. title, body text, Meta tag, and URL etc. There is no criteria selection for the collection of information

A Search engine might well be called a search engine service or a search service. As such, it consists of three components [2].

- Spider: Program that traverses the Web from link to link, identifying and reading pages
- Index: Database containing a copy of each web page gathered by the spider

- Search Engine mechanism: Software that enables users to query the index and that usually returns results in relevancy ranked order

### **2.1.1 Spider**

The spider is also called the crawler or web robot. The spider visits a web page, reads it, and then follows links to other pages within the site. This is what it means when someone refers to a site being “spidered” or “crawled”. The spider returns to the site on a regular basis, such as every month or two, to look for changes [3].

A spider is a program that automatically traverses the Web’s hypertext structure by retrieving a document, and recursively retrieving all documents that are referenced. Note that “recursive” here doesn’t limit the definition to any specific traversal algorithm; even if a spider applies some heuristic methods means to the selection and order of documents to visit and to space out requests over a long space of time, it is still a spider.

Normal Web browsers are not spiders, because they are operated by a human, and don’t automatically retrieve referenced documents.

### **2.1.2 Index**

The index is everything that a spider finds. The index sometimes called the catalog, like a giant book containing a copy of every web page that the spider finds. If a web page changes, then this book is updated new information.

Sometimes it can take a while for new pages or changes that the spider finds to be added to the index. Thus, a web page may have been “spidered” but not yet “indexed”. Until it is indexed or added to the index, it is not available to those searching with the search engine.

### **2.1.3 Search Engine Mechanism**

This is the program that sifts through the millions of pages recorded in the index to find matches to a search and rank them in order of what it believes is most relevant.

All search engines have the basic parts described above, but there are differences in how these parts are tuned. That is why the same search on different search engines often produces different results.

## **2.2 The differences among Search Engines, Directories, and Hybrid Search Engines**

### **2.2.1 Search Engines**

Search engines create their listings automatically. Search engines crawl the web, then people search through what they have found. If web pages have changed, search engines eventually find these changes. This change can affect how they are listed. Page titles, body, and other elements all play a role.

### **2.2.2 Directories**

A directory depends on humans for its listings. A short description of web page was submitted to the directory. A search looks for matches only in the descriptions submitted. Changing web pages has no effect on listing. Things that are useful for improving a listing with a search engine have nothing to do with improving a listing in a directory. The only exception is that a good site, with good content, might be more likely to get reviewed than a poor site.

### **2.2.3 Hybrid Search Engines**

Some search engines maintain an associated directory. Being included in a search engine's directory is usually a combination of luck and quality. Sometimes web page can

submit for review, but there is no guarantee that it will be included. Reviewers often keep an eye on sites submitted to announcement places, and then choose to add those that look appealing.

## **2.3 Search Engine Method of Text Searching**

Search engines use software robots to survey the Web and build their databases. Web documents are retrieved and indexed. When you enter a query at a search engine web site, your input is checked against the search engine's keyword indices. The best matches are then returned to you as hits.

There are two primary methods of text searching that is keyword and concept [4].

### **2.3.1 Keyword Searching**

This is the most common form of text search on the Web. Most search engines do their text query and retrieval using keywords.

Unless the author of the Web document specifies the keywords for her document (this is possible by using meta tags in the latest version of HTML)[5], it's up to the search engine to determine them. Essentially, this means that search engines pull out and index words that are believed to be significant. Words that are mentioned towards the top of a document and words that are repeated several times throughout the document are more likely to be deemed important.

Some sites index every word on every page. Others index only part of the document.

For example

- Lycos indexes the title, headings, subheading are the hyperlinks to other sites, along with the first 20 lines of text and the 100 words that occur most often.
- Infoseek uses a full-text indexing system, picks up every word in the text except commonly occurring stop word such as “a,” “an,” “the,” “is,” “and,” “or,” and “www.”
- Hotbot ignores stop words.
- AltaVista claims to index all words, even the articles, “a,” “an,” and “the.” Some of the search engines discriminate upper case from lower case; others store all words without reference to capitalization.

### 2.3.2 The Problem with Keyword Searching

- Keyword searches have a tough time distinguishing between words that are spelled the same way, but mean something different

For example

Hard cider, a hard stone, a hard exam, and the hard drive on your computer.

This often results in hits that are completely irrelevant to your query.

- Some search engines also have trouble with so-called stemming

For example

If you enter the word “big,” should they return a hit on the word, “bigger?”

What about singular and plural words? What about verb tenses that differ from the word you entered by only an “s,” or an “en”?

- Search engines also cannot return hits on keywords that mean the same, but are not actually entered in your query. A query on heart disease would not return a document that uses the word “cardiac” instead of “heart.”

### 2.3.3 Concept-Based Searching

Concept-based search systems try to determine what users mean, not just what they say. In the best circumstances, a concept-based search returns hits on document that are “about” the subject/theme users are exploring, even if the words in the document do not precisely match the words they enter into the query.

There are various methods of building concept-based systems, some of which are highly complex, relying on sophisticated linguistic and artificial intelligence theory. A numerical approach is currently the best known on concept-based searching. This approach determines meaning by calculating the frequency with which certain important words appear. When several words or phrases that are tagged to signal a particular concept appear close to each other in a text, the search engine concludes, by statistical analysis that the piece is “about” a certain subject.

For example

The word heart, when used in the medical/health context, would be likely to appear with such words as coronary, artery, lung, stroke, cholesterol, pump, blood, attack, and arteriosclerosis. If the word heart appears in a document with others words such as flowers, candy, love, passion, and valentine, a very different context is established, and the search engine returns hits on the subject of romance.

## 2.4 How Search Engines Rank Web Pages



A web page, or document, can contain various kinds of content (as opposed to display or presentation options like sound, animation or frames), some of which is not viewed in the browser [7].

- **Title** – an embedded description provided by the document designer; viewable in the title bar (it is also used as the description of a newly created bookmark by most browser)
- **Description** – a type of meta tag which provides a short, summary description provided by the document designer; is not viewable on the actual page; this is frequently the description of the document shown on the documents listings by the search engines that use meta tags [6].
- **Keywords** – another type of Meta tag consisting of a listing of keywords that the document designer wants search engines to identify the document. These too, are not viewable on the actual page
- **Body** – the actual, viewable content of the document.

Search engines may index all or some of these content fields when they are storing a document on their databases. (Over time, engines have tended to index fewer words and fields.) Then, using proprietary algorithms that differ substantially from engine to engine, when a search query is evaluated by that engine, its listing of document results is presented in order of relevance. Because of these differences in degree of indexing and algorithms used, the same document listed on different search engines can appear at a much higher or lower ranking (order of presentation) than on other engines.

Though not hard and fast, and highly variable from engine to engine, four factors tend to influence greatly the ranking of a document in a given query:

**Order a keyword term appears** – keyword terms that appear sooner in the document's listing or index tend to be ranked higher

**Frequency of keyword term** - keywords that appear multiple times in a document's index tend to be ranked higher

**Occurrence of keyword in the title** – keywords that appear in the document's title, or perhaps meta tag description or keyword description fields, can be given higher weight than terms only in the document body

**Rare, or less frequent, keywords** – rare or unusual keywords that do not appear as frequently in the engine's index database are often ranked more highly than common terms or keywords.

Some engines, notably Excite, attempt to “infer” what users mean in query based on its context. Thus, the meaning of *heart* can differ if the context of search is cardiac disease as opposed to Valentine's Day. The methods by which these inferences are made are statistically based on the occurrence of some words in conjunction with others. Though useful for simpler queries, such inference techniques tend to break down when the subject of the query or its modifier do not fit expected query relationships. For commonly searched topics, this is generally not a problem; it is a disadvantage to standard full-text indexing.

Cottage industries have emerged to help Web site developers place themselves higher in the search engines' listings (it is clearly more valuable to be within the first few listings sent to a user than be buried hundreds, or thousands, of documents lower). A constant battle is being waged between the engines and high listings from jimmying the system to “unfair” advantage.

Crude, early attempts to *Spam* search engines to get higher listings included adding hidden terms like *sex* that were searched frequently but not the real subject of the document. Other techniques were to use certain keywords repeatedly, such as *cars cars cars cars cars* to get a higher frequency rating. Another was to cram the page with high-interest terms using the same color as the overall Web page, thus *hiding* the added keywords. The leading search engines have caught on to these and now have automated ways to prevent the worst of these spamming techniques.

More subtle techniques, however, are hard to prevent, for example, a listing for ski resorts in Utah could also add hidden tags for “Caribbean” or “beach resort” knowing that wealthy Caribbean travelers may also be looking to take ski vacations. If the searchers ask for Caribbean vacations they may logically wonder why they have gotten a listing for Utah ski resorts. It is because of such techniques (among others) that they can sometimes get document listings from a search that seemingly have nothing to do with their query.

So, differences in how search services rank documents, how developer’s themselves choose to characterize their Web documents, and just simple errors in how computer process and index these pages can all lead to highly variable ranking results from different search services.

## 2.5 Major Search Engine

The major search services on the Internet are essential starting points for users seeking information. As such, they routinely are some of the most visited locations on the Web. Search services can be divided into two groups.

- Commercial-Commercial search services go to the effort to catalog information on the Internet to attract attention and advertising revenues.
- Non-commercial-Non-commercial services exist for many different reasons.

There are more than 1,000 search services presently on the Web [8]. There is a dozen or more big, major Internet search services:

- Alta Vista [<http://www.altavista.digital.com>]
- Google [<http://www.google.com>]
- Excite [<http://www.excite.com>]
- LookSmart [<http://www.looksmart.com>]
- Hotbot [<http://www.hotbot.com>]
- Infoseek [<http://www.infoseek.com>]
- Lycos [<http://www.lycos.com>]
- Magellan [<http://www.mckinley.com>]
- Mining [<http://home.miningco.com>]
- NetFind [<http://www.aol.com>]
- Northern Light [<http://www.nlsearch.com>]
- WebCrawler [<http://www.WebCrawler.com>]
- Yahoo [<http://www.yahoo.com>]
- GoTo [<http://www.goto.com>]

There are also metasearch services that provide a central access point to multiple of these services. Notable names are:

- Metacrawler [<http://www.metacrawler.com>]
- Inference FIND [<http://www.inference.com/infind/>]
- SavvySearch [<http://guaraldi.cs.colostate.edu:2000>]

For major Search Engine in Thailand, There are many Search Engines:

- Catcha.co.th [<http://www.catcha.co.th>]
- Siamguru [<http://www.siamguru.com>]
- Sanook [<http://www.sanook.com>]
- Lycos Asia [<http://th.lycosasia.com>]
- ThaiSearch [<http://www.thaisearch.org>]
- ThaiSeek [<http://www.thaiseek.com>]

Search engine uses ‘spiders’ or ‘robots’ to go out and retrieve individual Web pages or documents, either because they’ve found them themselves, or because the Web site has asked to be listed. Search engines tend to “index” (record by word) all of the terms on a given Web document. Or they may index all of the terms within the first few sentences, the Web site title, or the document’s Meta tags [5]. Due to the ever-changing nature of the Internet, the services must re-sample their sites on a periodic basis. Some of these services re-sample their site on a weekly or less-frequent basis.

## 2.6 Compare Search Engine features

### 2.6.1 Search Engine Math Commands

Command	How	Supported By
<b>Include Term</b>	+	All but LookSmart
<b>Exclude Term</b>	-	All but LookSmart
<b>Phrase</b>	“ ”	All but LookSmart (Note: automatic at AltaVista, Google)
<b>Match Any Term</b>	Auto	AltaVista (queries five words or more), Excite, GoTo, LookSmart, Netscape, Snap, WebCrawler, Yahoo
	Menu	AOL Search, HotBot, Lycos, MSN Search
	Other	Northern Light (use OR); Google (impossible)
<b>Match All Terms</b>	Auto	AltaVista (queries 2 to 4 words long), AOL Search, Google, HotBot, Lycos, MSN Search, Northern Light
	Other	Can be done at all using + symbol or menu options

Table 2-1: Compare Search Engine Math Command between major search engines

### 2.6.2 Power Searching Command

Command	How	Supported By
<b>Title Search</b>	title:	AltaVista, GoTo, HotBot, MSN Search, Northern Light, Snap
	other	some above via menus, Lycos (via menu), Yahoo
	none	AOL Search, Excite, Google, LookSmart, Netscape, WebCrawler
<b>Site Search</b>	domain:	GoTo, HotBot, MSN Search, Snap
	other	AltaVista (host:), Lycos (via menu)
	none	AOL Search, Excite, Google, LookSmart, Northern Light, Netscape, WebCrawler, Yahoo



<b>URL Search</b>	url:	AltaVista, Northern Light
	other	Lycos (via menu), Yahoo (u:)
	none	AOL Search, Excite, Google, GoTo, HotBot, LookSmart, MSN Search, Netscape, Snap, WCrawler
<b>Link Search</b>	link:	AltaVista, Google
	linkdomain:	GoTo, HotBot, MSN Search, Snap (only for root URLs; use menu for sub-URLs)
	none	AOL Search, Excite, LookSmart, Netscape, WebCrawler, Yahoo (n/a)
<b>Wildcard</b>	*	AOL Search, AltaVista, HotBot, MSN Search, Northern Light, Snap, Yahoo
	none	Excite, Google, GoTo, LookSmart, Lycos, WebCrawler
<p style="text-align: center;"><b>Notes</b></p> <p>At AOL Search, Title, Site, URL and Link search sometimes work as with HotBot, if you select search "Web only" on the Search Options page.</p>		

Table 2-2: Compare Power Searching Command between major search engines

### 2.6.3 Search Assistance Features

Feature	Offered By	Notes
Related Searches	AltaVista, AOL Search, GoTo, HotBot, iWon, Snap, Yahoo	IWon related search only works for its Direct Hit results
Clustering	AltaVista, Google, GoTo, HotBot, iWon, MSN Search, Northern Light	Excite has some clustering features
Find Similar	AOL Search, Google	

Stemming (Yes = on by default)	Lycos, Northern Light	Enable via form at HotBot, iWon, MSN Search, Snap
Date Range	AltaVista, HotBot, iWon, MSN Search, NBCi, Northern Light, Yahoo	
Search Within	AltaVista, Lycos, HotBot	
Case Sensitive	AVista, <i>Partially:</i> HotBot, NLight	
Direct Hit/ Popularity Ranking	HotBot, iWon, LookSmart, Lycos, MSN Search, Snap	
RealNames Links	AltaVista, Google, iWon MSN Search	
Spidered Version	Google	

Table 2-3: Compare Search Assistance Features between major search engines

#### 2.6.4 Display Features

Feature	Supported by
Results Shown By Default (10 unless noted)	AOL Search, Direct Hit, FAST, Excite, Google, GoTo (40) HotBot, iWon, Lycos, MSN Search (15), NBCi, Netscape Search, Northern Light, Yahoo (20)
Increase Number Of Results?	AltaVista, Excite, FAST, Google, HotBot, MSN Search, NBCi, Yahoo
See 20 Results	AltaVista, Excite, FAST, Google, HotBot, MSN Search, NBCi, Yahoo

See 50 Results	AltaVista, Excite, FAST, Google, HotBot, MSN Search, NBCi, Yahoo
See 100 Results	FAST, Google, HotBot, NBCi, Yahoo
Sort By Date	MSN Search, Northern Light
Date Displayed?	AltaVista, HotBot, Northern Light
Display Titles Only?	AltaVista, Excite, HotBot, MSN Search
Other Major Customize Options	AltaVista (& Raging Search)

Table 2-4: Compare Display Features between major search engines

2.6.5 Boolean Command

Command	How	Supported By
Or	OR	All but...
	None	Google, LookSmart, Yahoo
And	AND	All but...
	None	Google, LookSmart, Yahoo
Not	NOT	All but...
	AND NOT	AltaVista, MSN Search,
	None	Google, LookSmart, Yahoo
Nesting	( )	All but...
	None	Google, LookSmart, Yahoo
Near	NEAR	AOL Search (specify number) AltaVista & MSN Search (10 words), Lycos (25 words), WebCrawler (2 words)

	None	Others
<b>Notes</b> At AltaVista, Boolean only works on advanced search page At Excite-powered services, Boolean commands must be in UPPERCASE At Inktomi-powered services, set menu to "Boolean phrase"		

Table 2-5: Compare Boolean Command between major search engines

2.6.6 Search Engine Size

From charts below this, this is the meaning of each key: GG=Google, FAST=FAST, AV=AltaVista, INK=Inktomi, WT=WebTop.com, NL=Northern Light, EX=Excite.

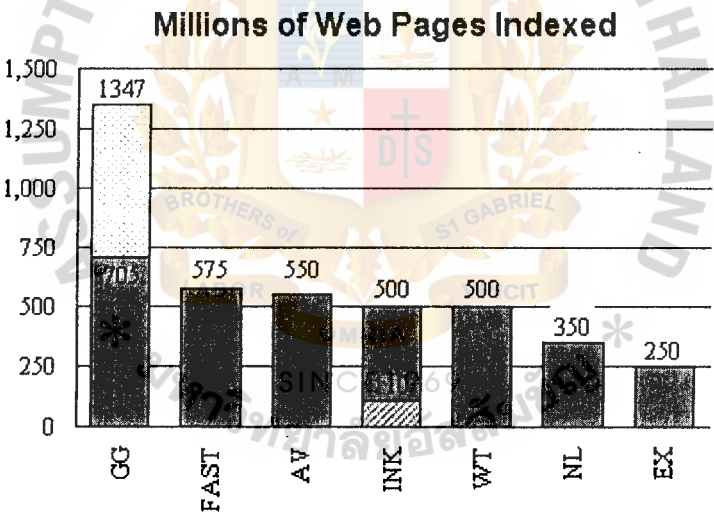


Figure 2-1: Compare Search Engine Size between major search engines

2.7 Literature Review

Few papers are proposed how to improve the performances of the Search Engine that satisfy the users' need. Here are the literature reviews on the related topics.

**Literature Review on Labs'-eye view: Altavista Search Engine 3.0 [Herb Bethoney, eWEEK, 2000]**

This paper proposes the method in creating file-indexing Search Engines. They don't follow hyperlinks the way Web crawlers do. Instead, they simple index all files in a given directory or directories, and then categorize the files by subject. A Search Engine can the search for terms within a page rather than just the page's header. This process yields more relevant search hits because the typical Search Engines will not return the searching's result in the related topic to the user They will show only the generally word or phrase that is the user's input.

**Literature Review on Altavista pushes the search envelope [Jim Rapoza, eWEEK, 2000]**

This paper, Altavista proposes the application to help the companies improve lackluster search capabilities. This application has also improved the engine's linguistic capabilities, enabling it to do advanced searches and handle phrases, misspelling and synonyms. In this development including support for several languages, as well as translations which feature will be provided in the same search interface. This application will not only be provided extended search capabilities but also delivered highly accurate results, returning the most appropriate pages among the top five results for pretty much every query.

**Literature Review on Catcha.com, First Asian Portal to Offer Ask Jeeves' Popularity-Based Web Search [Peggy Lee, Catcha.co.th, 2000]**

This paper proposes the using the new technology of Ask Jeeves. Ask Jeeves is a leading provider of intuitive, intelligent Web interaction solutions to deliver a humanized online experience. Jeeves Popularity SearchSM utilizes patented popularity technology that

reflects the collective experience of millions of Web searchers, enabling a customer-driven approach that increases the relevance of search results in response to customer queries. Catcha.com tries to merge the existing system with Ask Jeeves to improve the searching performance.

**Literature Review on Excite Changes Look, Results [The Search Engine Report, 2000]**

This paper, Excite proposed a new look to its search results that also coincides with a new ranking system. Excite is making better use of link analysis than it has in the past to improve its search results. "News Search" allows user to search for "Web News" articles from over 350 news sites, which are checked several times per day. Excite provided Category Search and user can search for matching categories or web sites that come out of Excite's directory, which is primarily powered by LookSmart. Some listings remain from the days when Excite assembled the directory itself. In searching from Category Search, user can get the variety of information on the same group of the input keyword.

**Literature Review on A Search Engine Worth Gambling on [Lisa Moskowitz, special to PC World, 1999]**

This paper, Google proposed the method for increasing speed in searching from large database. Google has indexed approximately 100 million of the estimated 800 million Web pages. It can list pages it hasn't indexed; but those pages aren't ranked for relevance. That extends Google's present page reach to about 300 million. Google's site offers a clean, simple interface--no e-mail, directory lists, or pulsing banner ads for increasing speed.



## **Literature Review on AltaVista Europe Debuts Helpful Search Engine Features**

**[Danny Sullivan, The Search Engine Report, May 2001]**

This paper, AltaVista Europe has been proposed a helpful search management feature and new thumbnail images that appear next to some search results. AltaVista called this search management feature that "MySearch". "MySearch" is a search management tool that will automatically record your most recent 25 searches via its "Search Tracker," plus allow you to easily save them into a "My Favorite Searches" area, so that they can easily be rerun in the future. But in this feature must work with cookies technology. So if you change computer or web browser which is different from first of retrieved, you cannot use this feature. For new thumbnail image, AltaVista Europe has also begun including images as part of its regular search results. Known internally as Image Enhanced Results, or IER, this is where some listings in search results have images associated with them. Try a search for "london" at AltaVista UK or "eiffel tower" at AltaVista France to see examples of this. The thought is that including images can help improve relevancy -- or at least the user's ability to judge results more relevant to their query than others.

## **Literature Review on Google Acquires Deja Newsgroup Service [Danny Sullivan, searchenginewatch.com, March 2001]**

This paper, Google already purchased the Deja newsgroup and is now running them within its own site. Despite saving the service, Google came under criticism from hardcore Deja users upset about lost functionality during the transition period. Google has added some more features to the service of Deja too. And users of Google will be able to access all of Deja's archives, which stretch back to 1995. The archives, which contain 500 million messages, are currently only accessible to the middle of last year. So Google can provide a lot of results to user from integrated database of Deja newsgroup.

**Literature Review on iLOR Makes Google Even Better[Danny Sullivan, searchenginewatch.com, April 2001]**

This paper, iLOR is a new search service that takes the power and relevancy of Google's results and adds on some nifty features that many searchers may find useful. After performing a search, iLOR brings back Google results. Unlike Google, you can hover over any of the listings to reveal four options called "LORlinks."

First option, the "Put In My List" option adds the listings to a new browser window that opens up. You can then continue to pick out all the listings you are interested in, building up a custom list of your picks. The list can also be emailed or added to your bookmarks.

Second option, the "Go Now" option is very nice for those who want to explore a particular listing but then hate hitting the "Back" button to get back to their original search results. When you use Go Now, a new window opens that "anchors" you to the original search results. Whenever you want to return to them, you just use the link in the anchor window to get back instantly.

Third option, the "Open In Task Bar" option opens the listing in a new window, but it then immediately minimizes the window, so you can still see the original results list. I'm one of those people who constantly opens new windows from results, and if you are like me, this will save you from window clutter.

Fourth option, the "Open In New Window" option does exactly that, opening the listing in a new window without minimizing it. This is the least spectacular option, since right-clicking on a listing usually brings up opening a result in a new window as an available choice.

For these options, users don't need to download anything. Users can use these services with any web browsers. This is the new way of searching.

**Literature Review on How to increase traffic to your web site without spending advertising dollars [Fredrick Marckini, iProspect.com, Inc, 2001]**

This paper, WebPosition Gold is the first product to combine all the following features designed to launch your Web site to the top of the search engine results.

1. Generates HTML pages designed to rank near the top of the search results.
2. Analyzes your existing Web pages and gives plain-English advice on how to improve them.
3. Includes a simple, built-in HTML editor for fast and easy changes.
4. Assists in uploading your new and changed pages.
5. Submits your pages to the major search engines automatically.
6. Reports your positions on each search engine for each keyword you are targeting.
7. Tracks the number of visitors to your site, where they came from, and what keywords they used to find you.

Form these features, are the new way to increase traffic for your web site and you can save a lot of money in advertisement. A top 10 ranking in a major search engine like AltaVista, Lycos, or Hotbot will often generate more targeted traffic than an expensive banner advertising campaign - and, a good search engine position is like highly targeted advertising that is both free, and effective.

**Literature Review on Google Joins Forces With Vodafone Global Platform And Internet Services To Bring Advanced Search Capabilities To Mobile Internet Portal [Marshall Simmonds, about.com, April 2001]**

This paper, Google search engine has been selected by the Vodafone Global Platform and Internet Services Group (VGP) as the exclusive search engine provider for its worldwide mobile Internet portal, branded Vizzavi. The Google search engine, which offers Vizzavi users access to more than 1.3 billion web pages, will be integrated with other VGP partner technologies to provide the most comprehensive content and services Internet portal for both desktop PC and wireless users. From this service, Google can provide users around the globe with access to the best search experience possible.

**Literature Review on Internet Services To New Search Engine Spider Crawls the Web [Mark Joyner, Aesop.com, April 2001]**

This paper, Aesop.com launches new search engine web with 2.5 million web pages indexed. Spider gathered all of the information. Aesop.com believes we can make a significant contribution to the Internet community by offering a new paradigm that will cut down the amount of time it takes searchers to find relevant information with new search technology.

**Literature Review on AltaVista Moves to Eliminate Spam [Paige Bridgers, Web Optimization Network, April 19, 2001]**

This paper, the "spamming" of search engines, or the repeated submitting of a URL, is a common occurrence. For those that maintain the search engines "spam" is more than a nuisance, it is a waste of time and money. Alta Vista, like all search engines, fights in the ongoing battle against "spam." Their main adversary in this war is the automated submission software, which, with the touch of a button, can submit thousands of URLs to nearly all search engines. However, in a recent effort to eliminate "spam," Alta Vista may finally have made headway against the use of these automated submission scripts. Alta

Vista has adjusted its URL submission procedure to include a verification code that must be entered by a human. After the code is entered, the submitter is limited to submitting a maximum of 5 URLs.

**Literature Review on NativeMinds Selects Inktomi Search Technology to Enhance Natural Language Virtual Representative Solutions [Yahoo.com, May 2001]**

This paper, NativeMinds, a leading provider of software to build automated, natural language customer service representatives, announced a technology agreement with Inktomi Corp., developer of scalable Internet infrastructure software, to enhance the online customer self-service experience by adding highly relevant corporate Web site search results to its sophisticated virtual representative-assisted responses. At the core of NativeMinds' solutions are automated customer service agents called virtual representatives, or vReps(TM) for short. vReps manage a company's online customer interactions by understanding users' natural language questions and providing fast, effective answers in the form of two-way conversation without the need for human intervention. In addition to conversational replies, vReps provide information gathered from multiple sources, including CRM systems, knowledgebases, document search results, and Web site content. NativeMinds vReps will use Inktomi Search Software to effectively answer general research questions by combining its immediate, two-way conversational response with a list of appropriate search results (URLs) from the host company's Web site.

**Literature Review on Ranking Placement of Keyphrases In Order of Importance [Heather Lloyd-Martin, The rank write roundtable newsletter, May 2001]**

This paper, recommend the techniques about how search engine rank web site.



1. The search engine figures that hyperlink keyphrases are important. Every hyperlink on your web site should include a keyphrase.
2. In headlines and sub headlines should include at least on keyphrases. Search Engine will think that headlines are important, highly relevant text.
3. Spread keyphrases throughout body document is better than include in the first couple line of text.
4. The search engine gives the meta keyword with the less weight.

## **2.8 State of problem**

Most of the Search Engines are still working in directory style; they aren't the real Search Engines. That means, in creating index and managing database of the Search Engine are worked by human. If web site has updated, list of Search Engine in database will not be updated immediately. User cannot get the good list in searching. Another problem of the Search Engine is the results of Search Engine don't meet the users' satisfaction because each word can be identified in many meanings. Search Engine cannot understand what the user wants from each word.

Exactly, the main Search Engines in Thailand aren't the real Search Engine. They work in directory style [9]. Thai word is very difficult to identify the meaning because each word has several meanings. These Thai Search Engines still meet these problems, which cannot identify the meaning of each word or what the user wants from the input word. The meaning of each word or the related word has to input or update manually by human. For cutting the word, Thai Search Engines use the cut text software in cutting. This software needs a huge database of Thai word in cutting each word and it will select only the longest word to create the keyword index on the database. That is the problem.

Search Engine that returns lots of lists in selecting, doesn't mean that all of the lists will meet the users' requirement. It doesn't need to return a lot of them, but it should return only what user really needs.

For normal Search Engine, it will have 4 factors for ranking each web page: 1. order a keyword term appear, 2. frequency of keyword term, 3. occurrence of keyword in the title, and 4. rare, or less frequent of keyword. The problem will occur, if webmaster intends to create spam on his website or do something that make search engine rank his website. In this case, it makes Search Engine return the bad results to user.





## **CHAPTER 3**

### **THE PROPOSED METHOD**

#### **3.1 Introduction of The Proposed Method**

Search Engine is the important tool for searching the wanted information on the WWW. The typical Search Engines are the web directories that don't have a tool for automatically retrieving the information for the WWW. The information of each web site on Search Engine database should update frequently, if those web sites have been updated. For each searching, user will get the poor list of results. The good Search Engine should provide only what the users need user's need as much as it can.

After review the literatures, I can design and propose a model of search engine [10] by adopting some concepts and theories from those literatures and the problems of search engine. So the main propose of this thesis, I will use the searching's rank of each web site for improving the quality of result, the index's relationship between keyword with URL for improving speed of searching, and also developing Thai Search Engine in an organization which has its own Intranet [11].

For the new method of storing the keyword, I will use the keyword which user inputs to make new keyword and use the URL which user selects [12] to increase rank and make the relationship with input keyword. I propose this method because the cut text software will cut the longest word from the database and it will know only the word that stores in the cut text software's database. In this proposed method, the system cannot only create the variety of keywords from user, but also learn which keyword has related with URL from user.

For using the searching's rank, it will increase the rank of URL which user selected. This method will help increase the quality of searching 's result, and using the index's relationship between keyword and URL will increase speed in searching because every time in searching, the system will check the keyword at the "Keyword Database" first. If input's keyword has related with keyword in database, the process will go to the "Related Database" to check the relationship with URL. If it has related, it will retrieve the related web page from URL Database and order by rank of URL (in keyword group), the highest rank will show first (popularity of URL).

### 3.2 The Context Diagram of The Proposed Method

From Figure 3-1, it will show the context diagram [13] that contains 5 main processes:

Process 1-Search data for DB [15]: it is the process for searching input keyword from database.

Process 2-Display results to user: it is the process for providing URL detail to display to user: it will get the sorted URL detail from Process 1. In this process will display 10 URL detail in each page.

Process 3-Update DB: it is the process for updating database after user selects some URL.

Process 4-Delete unused DB: it is the process for deleting the unused database.

Process 5-Gather the information from web server in organization: it is the process for gathering the information from each web page.

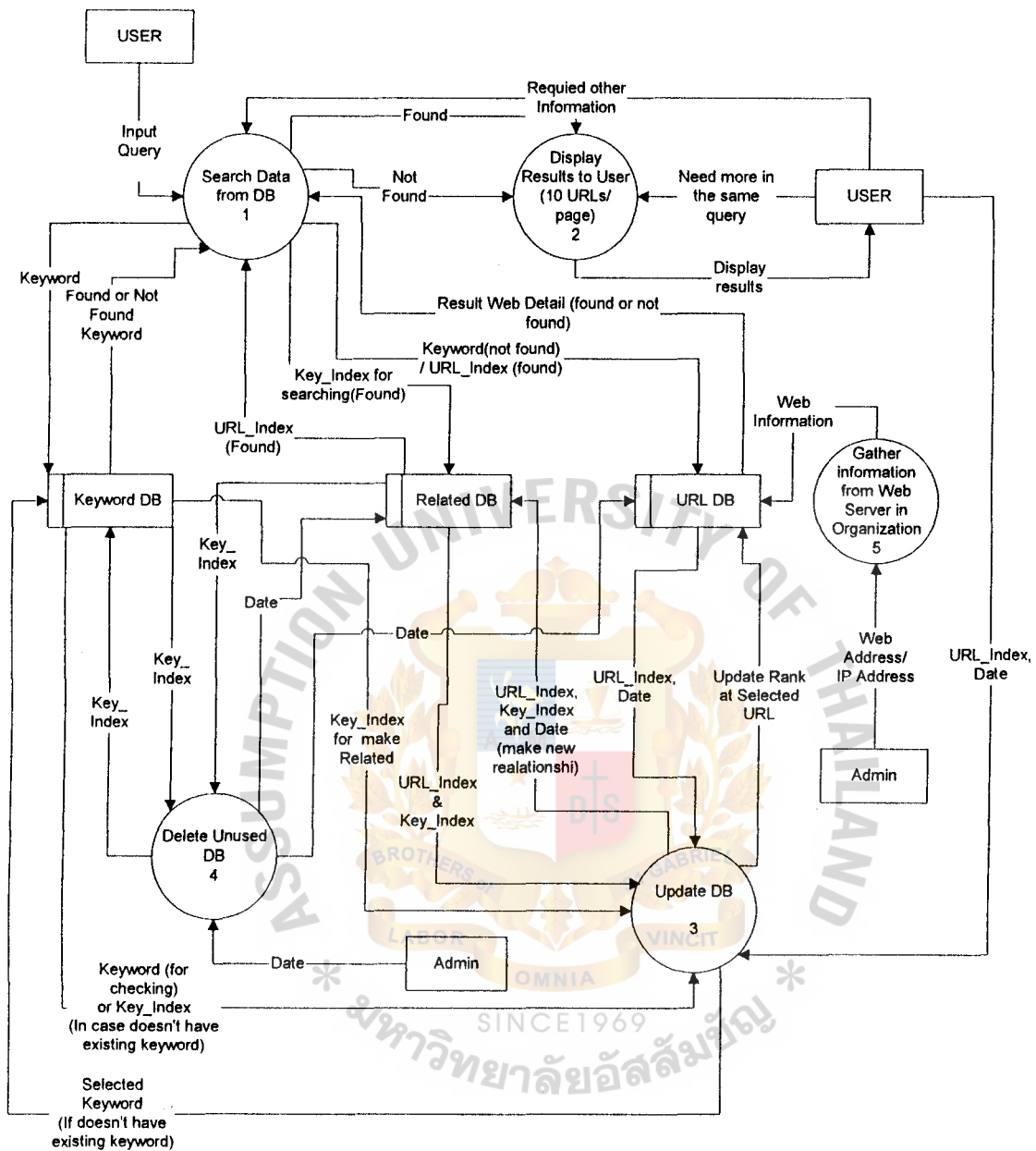


Figure 3-1: Context Diagram of The Proposed Method

3.2.1 Level 1 of Process 1: Search data from database

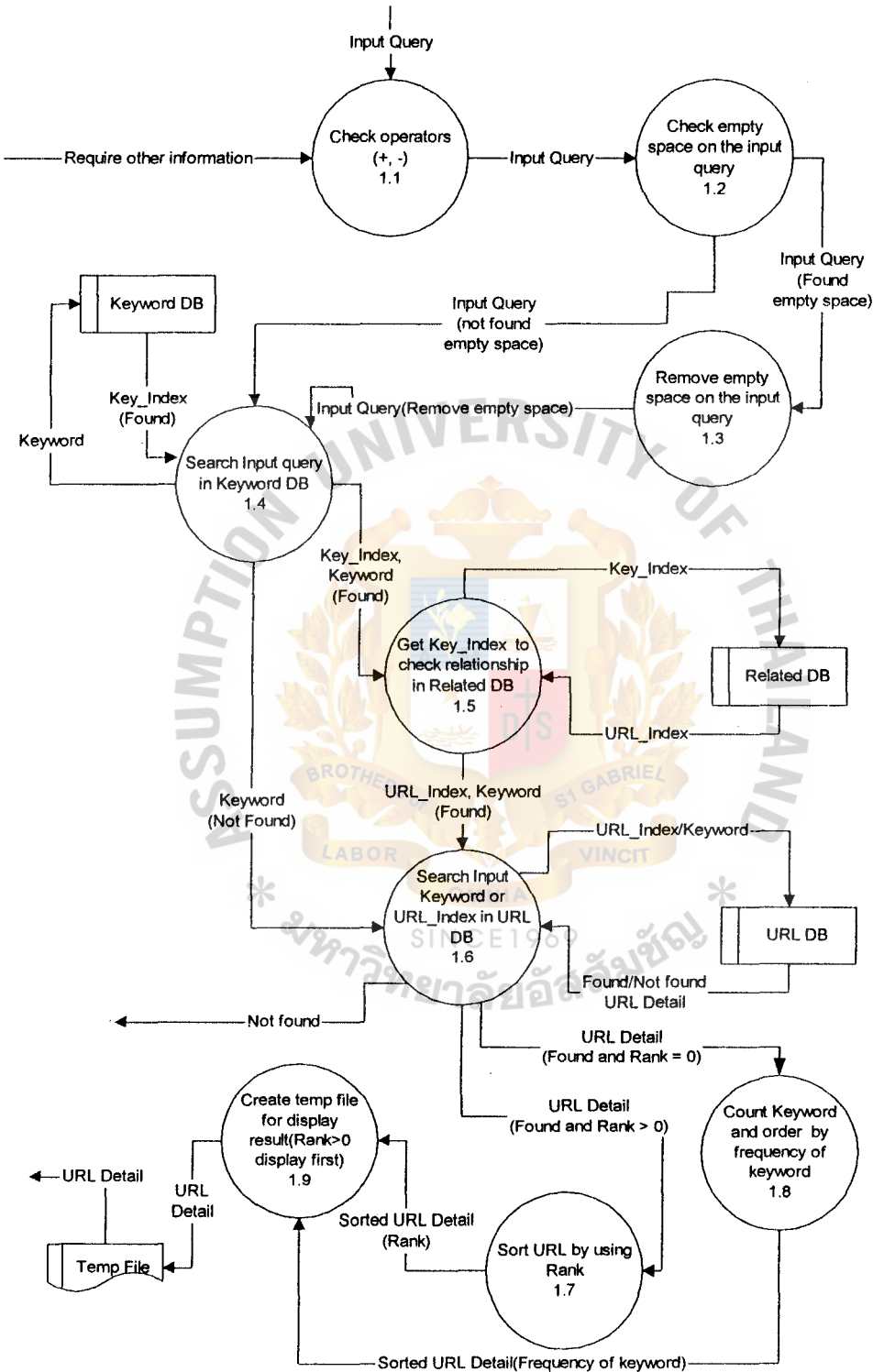


Figure 3-2: Level 1 of Process 1: Search data from database

From Figure 3-2, it is the Process 1: Search Data from Database is the process for searching in Keyword, Related, and URL database by using input query of user. Process 1 has 9 sub processes:

Process 1.1: Check operators (+, -): it will check input query that it has operator or not. If it has these operators, it will translate the meaning of operators and send with keyword to the process 1.2.

Process 1.2: Check empty space on the input query: this process will check the empty space of input query because when input query was kept in database, it shouldn't have empty space.

Process 1.3: Remove empty space on the input query: if the input query has the empty space, it will send to this process for remove it.

Process 1.4: Search input query in Keyword Database: it receives input query (no empty space and meaning of operator (if it has)) to search in Keyword Database. If it is found, it will get the Key\_Index for searching in Related Database (Process 1.5). If it doesn't find, it will send keyword to search from URL Database (Process 1.6).

Process 1.5: Get Key\_Index to check relationship in Related Database: it receives Key\_Index from Process 1.4, this process will check relationship of Key\_Index with URL\_Index and send URL\_Index to get URL detail in Process 1.6

Process 1.6: Search Input Keyword or URL\_Index in URL Database: in this process will work with URL Database by using Keyword or URL\_Index for searching URL detail. By this process, it has got 2 results: First, it will or will not find URL detail, if the process searches from Keyword. Second, if the process searches for URL\_index, it will provide the URL detail that user wants.

Process 1.7: Sort URL by using Rank: this process will get all URL detail and rank of each URL. It will sort URL by using rank (descending) and send sorted URL to Process 1.9.

Process 1.8: Count keyword and order by frequency of keyword: it will count the inputted keyword of web page's result that has rank = 0, and order it by frequency of keyword in each web page. Web page that contains a lot of inputted keyword will show first.

Process 1.9: Create temp file for display result: it will create temp file that contains sorted URL detail (rank > 0 will show first) for display to user.

### 3.2.2 Level 1 of Process 3: Update Database

From Figure 3-2, it is the Process 3: Update Database is the process of updating Keyword, Related, and URL database after user has selected the URL. Process 3 has 5 sub processes:

Process 3.1: Check selected URL and inputted keyword in DB: in this process, it receives selected URL and inputted keyword from user, it will send selected URL to add rank in Process 3.2, get URL\_Index back to check relationship in Process 3.4. For inputted keyword, the process will check the keyword in database if it contains this keyword or not. If keyword database doesn't have this selected keyword, the process will send keyword to database to add new database.

Process 3.2: Add Rank at selected URL in URL Database: it gets selected URL from Process 3.1 and comes to update selected URL's rank in URL database.

Process 3.3: Create new Keyword in Keyword Database: This process will add the selected keyword to be the new keyword in Keyword Database and send Key\_Index to create relationship in Related URL.

Process 3.4: Check relationship of Keyword and URL: it will check keyword and URL that have relationship in related database or not? (in case at found keyword in keyword database) If they don't have relationship, this process will send URL\_Index and Key\_Index to Process 3.5 for adding.

Process 3.5: Create relationship of URL and Keyword in Related DB: this process will create relationship between URL\_Index and Key\_Index.

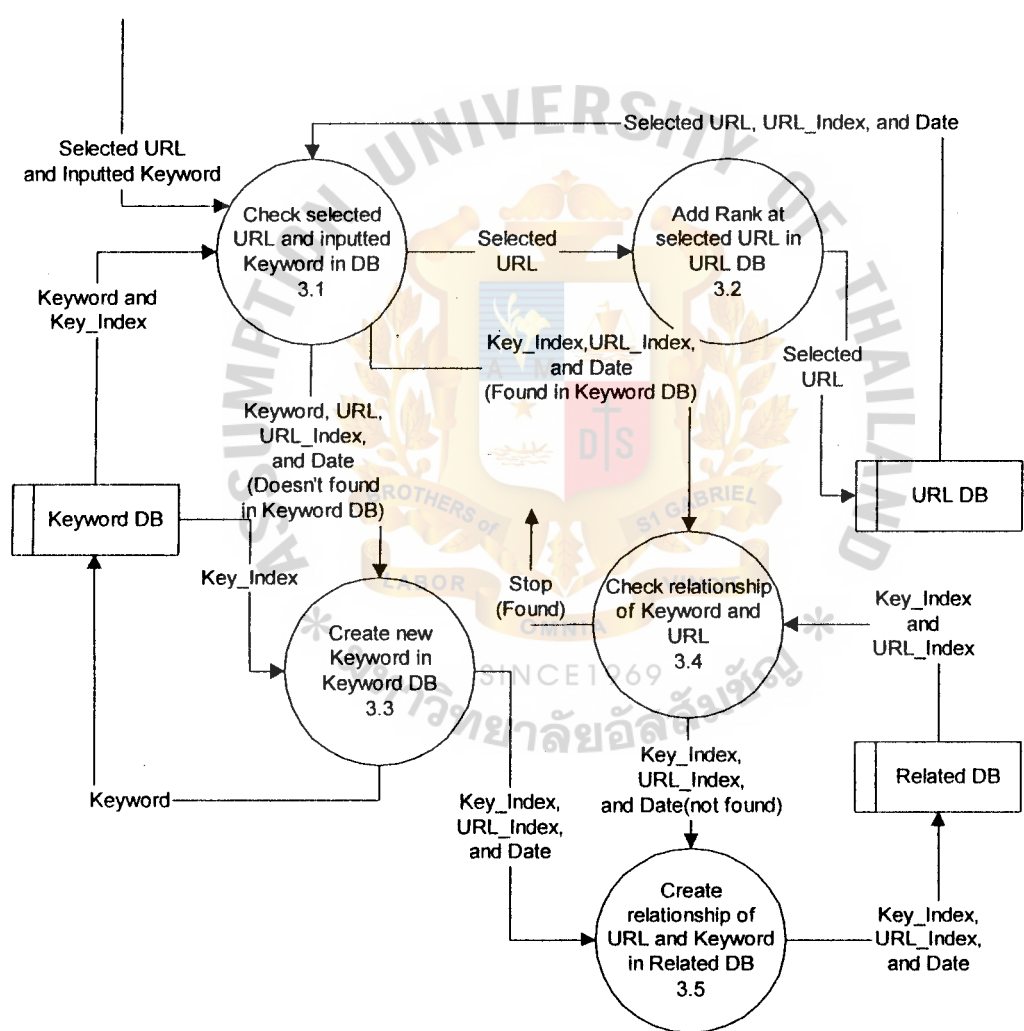


Figure 3-3: Level 1 of Process 3: Update Database



3.2.3 Level 1 of Process 4: Delete Unused Database

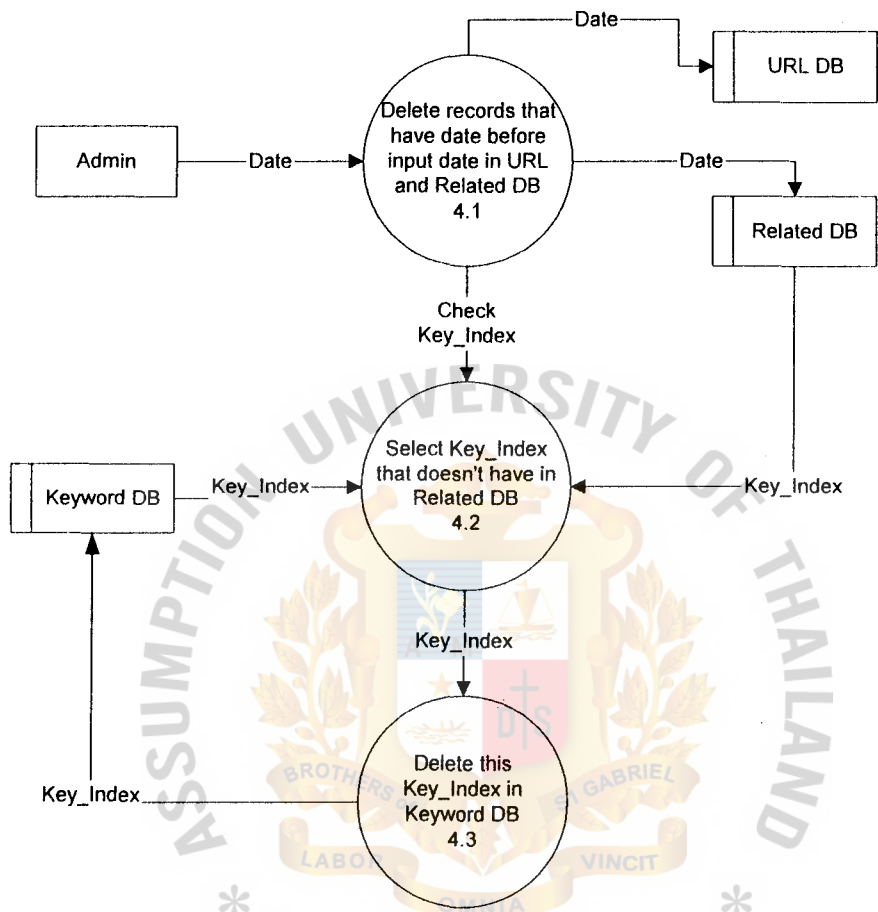


Figure 3-4: Level 1 of Process 4: Delete Unused Database

From Figure 3-4, it is the Process 4: Delete Unused Database is the process for maintenance Keyword, Related, and URL database by deleting unused database. Process 4 has 3 sub processes:

Process 4.1: Delete records that have date before input date in URL and Related Database: In this process, administrator will input date for delete record that is older than input date in URL and Related Database

Process 4.2: Select Key\_Index that doesn't have in Related DB: in this process will check which Key\_Index in Keyword database that doesn't have in Related database.

Process 4.3: Delete this Key\_Index in Keyword database: it will get the Key\_Index that doesn't have in Related database. It will delete this Key\_Index in Keyword database.

### **3.2.4 Level 1 of Process 5: Gather information from web server in organization**

From Figure 3-5, it is the Process 5: Gather information from web server in organization is the process of gathering and collecting URL information from web server in organization, Administrator can input range of IP address or web site name in gathering information. Process 5 has 15 sub processes

Process 5.1: Check Input whether it is IP Address or Web address? : It will check input type, if it is web address; the process has to send to convert IP address to another process. If administrator input the range of IP Address, this process will get the last IP Address to store in the temp file and the first IP Address to check in the next process.

Process 5.2: Convert to IP Address: this process will receive web address from Process 5.1 and convert to IP Address.

Process 5.3: Keep last IP Address of input searching range in temp file: it will receive the last IP Address from Process 5.1 and keep this IP to compare in the temp file.

Process 5.4: Check timeout of this IP Address: this process will check timeout of each input IP Address.

Process 5.5: Increase IP Address: this process will increase IP Address (+1) that timeout (Process 5.4) or more web page (Process 5.14).

Process 5.6: Check IP, this is more than the last IP (or blank file) or not: after increase IP Address, Process 5.5 will send to this process to check this IP Address whether it is more than last IP Address or not? If it is still less, it will send back to Process 5.4.

Process 5.7: Check whether this IP is Web Server (http://) or not? : It will use IP Address to check whether this IP Address provides Web Server or not?

Process 5.8: Get Web Page Detail: After the process proves this IP Address is web server, it will get the detail from web page.

Process 5.9: Remove HTML TAG: After getting web page detail, it will remove all common tag in this page.

Process 5.10: Get all links on Web Page (Only the processing IP): At Process 5.8, it will send web detail to this process to find link on this page and keep it in temp file.

Process 5.11: Compare the Existing links in temp file: this process will receive links of web page from previous process, it will firstly take these links to compare with the existing link before adding them in the temp file. The process will send links that don't duplicate with the existing link to the next process.

Process 5.12: Add link that doesn't have in temp file: it will get link from process 5.11 and add them to temp file.

Process 5.13: Add to URL database: this process will get web page detail and IP address from Process 5.9. It will add these details to URL database.

Process 5.14: Mark the accesses web page: this process will get web page detail from the previous process. It will use this web page to mark in temp file that is already retrieved information from this web page to prevent from looping to retrieve this page again.



3.3 ER Diagram of The Proposed Method

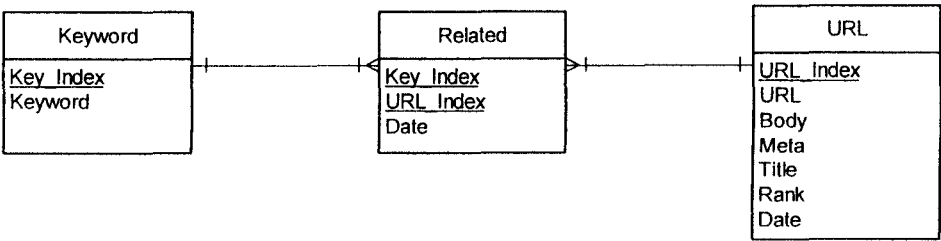


Figure 3-6: ER Diagram of The Proposed Method

From Figure 3-6, this is the ER diagram [14] that shows the database’s relationship of the Proposed Method. It has 3 main tables: Keyword, Related, and URL. The Related table has Key\_Index to be the primary key and URL\_Index to be the secondary key and also has relationship with Keyword and URL table in many to one.

The process’s step of the Proposed Method for using database, the system (spider) will start to collect the web page information and store in the URL table, and when user inputs the keyword for searching, after that the system will return the result to user, and user selects some URL. The system will create the selected keyword to be keyword in the Keyword table, create the relationship between this keyword with the selected URL in Related table, and increase rank in the selected URL in URL table.

3.4 Pre-Processing of the Proposed Search Engine

For development of this Proposed Search Engine, I have to setup:  
Software:

- 1. Windows 2000 Server

2. Internet Information Service 5 (IIS5)
3. MS. SQL7
4. Visual Basic 6
5. Internet Explorer 5

#### Hardware (Server)

1. CPU Pentium II 400 MHz.
2. RAM 128 MB.
3. VGA card
4. Hard Drive 8. GB.
5. CD-ROM
6. Floppy Disk
7. LAN Card 10/100

#### Network

1. DNS Name
2. 1 dedicated IP Address
3. High Speed network

#### Database

1. Collect information from each web server in organization by using spider.
2. Use cut text software for making index that is popular and makes base relationship between keyword and URL. We can select popular keyword, input in cut text database and let's cut text software generate index from URL database.

Main processes of creating keyword by use cut text software; we divide into 5 main processes:

Process 1: Retrieve keyword: In this process will retrieve keyword from database, and check whether it still has keyword for retrieving in cut text database or not.

Process 2: Check this keyword in keyword database: it will check keyword that retrieves from cut text database with keyword database. If it has keyword in database it will send command to process 1 again to retrieve another keyword. If it doesn't have in keyword database, it will send this keyword to process 3.

Process 3: Search keyword in URL database: in this process will use this keyword to search in URL database, if it finds, it will send to process 4 to add in keyword database.

Process 4: Add keyword into keyword database: add this keyword into keyword database and send key\_index of this keyword to next process to make relationship with URL that contains this keyword.

Process 5: Create relationship: It will get key\_index and URL\_index to make relationship.



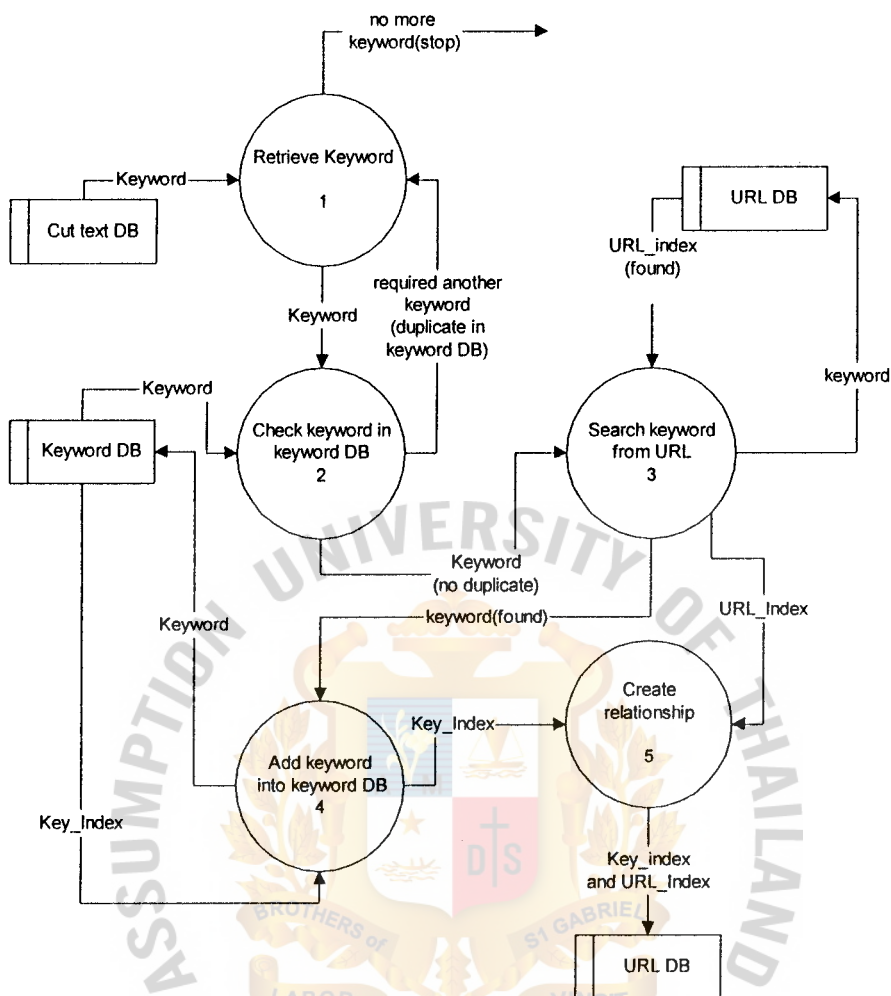


Figure 3-7: Pre-Processing of creating index

## CHAPTER 4

### DATA ANALYSIS

#### 4.1 Theory of searching's time with my proposed Search Engine

For my proposed Search Engine, I develop search algorithm to use with the relationship between keyword and URL. From this algorithm, I can generate the formula for finding the average time of searching with each keyword because each searching with each keyword will not provide the same result and time, they depend on the relationship between keyword and URL. In this formula has many factors to use for calculating such as, size of URL's database, and relationship between keyword and URL.

This is the formula for calculating the average time of searching

$$Y = (0.9)^{X+B} + S$$

Y: Average time (Sec.)

X: Number of relationship between keyword and URL

S: Size of database, we can find the value of S by using "number of record in URL Table/625" (~625 URL's records use 1 Second for searching).

B: The average of the related value that calculated from

$$B = \text{Log}(0.5) / \text{Log}(0.9)$$

$$B = 0.73249$$

Log(0.5): is the different between the highest and the lowest average time of searching.

Log(0.9): we use 0.9 because if you got a lot relationship between URL and Keyword.

Searching's time will go to near 1. So we use 0.9 to be the value for calculating.

After calculation, we can get the constant value that use for this formula.

$$Y = (0.9)^{X+0.73249} + S$$

From this formula, we will test with the simulated information, by using value of relationship between keyword and URL from 0 to 100, and 2500 records in URL Table.

Relationship	Average Time (Sec)
0	4.92
10	4.32
20	4.11
30	4.04
40	4.014
50	4.0047
60	4.0016
70	4.0005
80	4.0002
90	4.00007
100	4.00002

Table 4-1: Searching time from theoretical formula

From these results, we can make graph in the figure 4-1.

According to the graph, it shows that if you have the relationship between the keyword and URL, time for searching will be reduced. That is the main feature that I develop for this Search Engine.

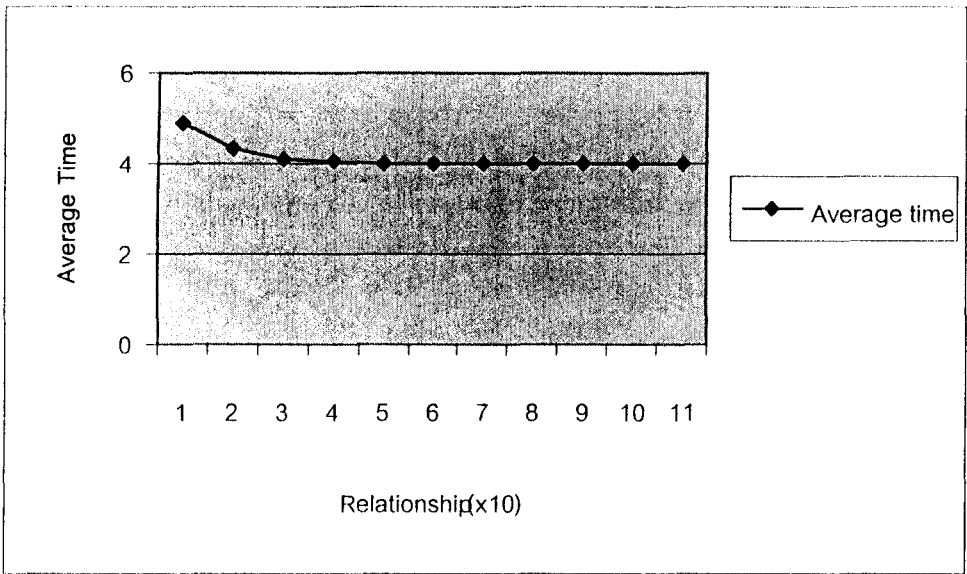


Figure 4-1: Searching time from theoretical formula

#### 4.2 Comparison time for searching between the Proposed Search Engine and a Search Engine that uses only URL database for searching.

The proposed system of search engine is the system that uses the Keyword table, Related table and URL table for searching by using the relationship between the Keyword and URL. The proposed system will use time and quantity of URL in Related table to compare with the Search Engine that uses only a URL database for searching.

For this experiment, we use 2,476 URL's records for searching and use “ช้าง” as a keyword for searching. We can find 161 URL's records from this keyword. We search 10 times for each rank level in the Proposed Search Engine.

#### Result of the Search Engine that uses only URL database for searching

Search	1	2	3	4	5	6	7	8	9	10	Average Time
Time (Sec)	5	4	4	5	5	5	5	5	5	4	4.7

Table 4-2: Searching time of the Search Engine that uses only URL database for searching

For these results of the Search Engine that uses only URL database for searching, we make a single experiment because it will go to search from the URL table directly. From this experiment, we use the same keyword and search this keyword 10 times, so we can get the average time of the Search Engine that uses only URL database for searching. It is 4.7 seconds.

Result of the Proposed Search Engine

Search	1	2	3	4	5	6	7	8	9	10	Average Time
0 Relationship	5	4	5	5	5	5	4	5	5	4	4.7
10 Relationships	4	5	4	4	5	5	4	5	5	4	4.5
20 Relationships	5	4	4	5	5	5	4	4	5	4	4.5
30 Relationships	4	5	5	4	4	5	4	5	5	4	4.5
40 Relationships	5	4	5	4	4	5	5	5	4	4	4.5
50 Relationships	4	4	5	4	5	5	5	5	4	4	4.5
60 Relationships	5	5	5	5	4	5	4	4	4	4	4.5
70 Relationships	4	4	5	5	4	5	4	5	4	4	4.4
80 Relationships	5	4	4	5	5	4	4	5	4	4	4.4
90 Relationships	5	4	5	5	4	5	4	4	4	4	4.4
100 Relationships	4	5	4	5	5	4	5	4	4	4	4.4

Table 4-3: Searching time of the Proposed Search Engine

For this result of the Proposed Search Engine, we make 10 experiments from 0 relationship to 100 relationships in a related table. The searching method will first check the keyword table first. If it is found, it will match the relationship of this keyword in the Related table and retrieve URL data in the URL table. From this experiment, we use the same keyword and search this keyword 10 times for each level of relationship, so we can get the average time of this Proposed Search Engine. It means that if we have a lot of relationships in the Related table, it will produce a lower average time.

From the results of the Search Engine that uses only URL database for searching with the Proposed System, we can produce a graph. The time of the Search Engine that uses only URL database for searching will be constant, but doesn't have the factor to make change. But the time of the Proposed Search Engine will reduce if the relationship between keyword and URL increase in related table.

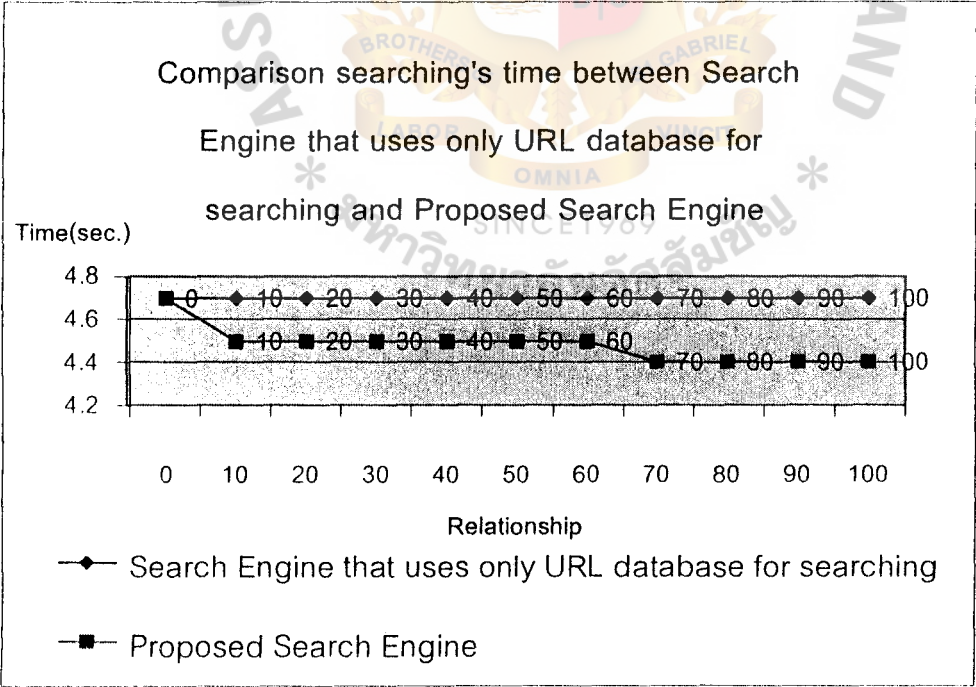


Figure 4-2: Comparison searching's time between Search Engine that uses only URL database for searching and Proposed Search Engine



4.3 Comparison the position of searching’s result between Proposed and Search Engine that uses only URL database for searching.

In this comparison, we will use rank field in the URL table of the Proposed Search Engine to compare with Search Engine that uses only URL database for searching that doesn’t have the rank field to help for searching.

In this experiment, we set every value of rank fields in URL table to be “0” and use the same keyword for searching. We use 3 cases for testing:

- 1. Select URL in the first page. In this case, if you still want information on this URL for the next time, both of Search Engines will give you the result in the first page.

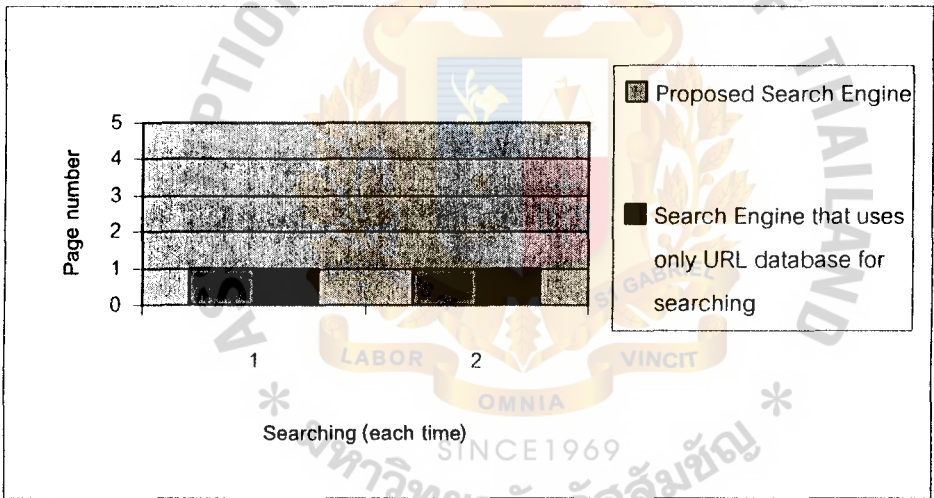


Figure 4-3: Comparison of the position of searching result for each time (in the case of the first page)

- 2. Select URL in the Last page. In this case, if the next time you still want information on this URL, the Proposed Search Engine will provide this URL in the first page because searching every time will be ordered. But for Search Engine that uses only URL database for searching, you have to go to the last page to find this URL.

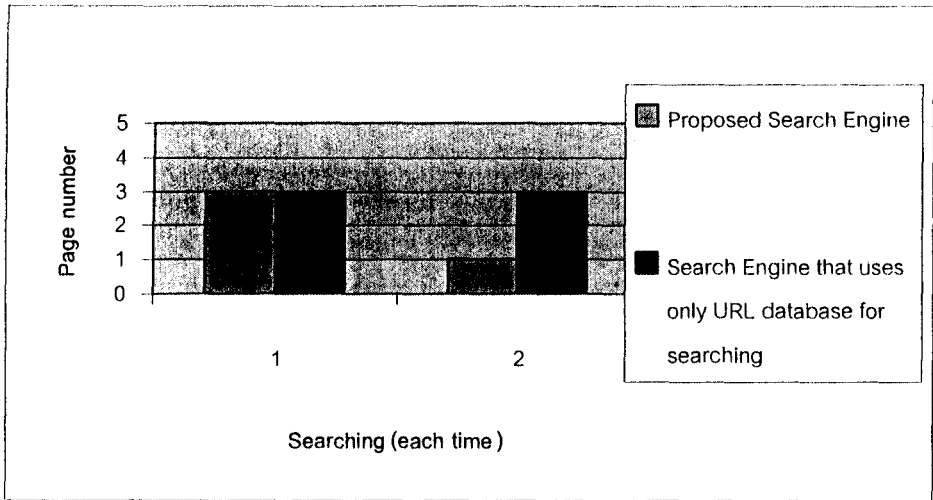


Figure 4-5: Comparison of the position of searching result for each time (in the case of unrecognized page)

From this experiment, the Proposed Search Engine will show the variety of results that depend on the rank of each web page. It will provide the popular web page for the user in the first page. But Search Engine that uses only URL database for searching will show the constant of searching results.

4.4 Experiment with the Proposed Search Engine in the organization.

From this experiment, I setup this proposed search engine for users at Planet Communications Asia Co., Ltd. to test it and make the questionnaire. In this questionnaire, every user will find the best product information for Chiang Mai by using keyword “เชียงใหม่” and answering 2 questions:

- 1. How many web pages does the user select on the whole results?
- 2. How long does the user spend before user select each web page?

For question no.2, we have 4 ranges to measure time

- 1) Less than 1 Min.
- 2) Between 1 Min. and 2 Min.

24	3	4
25	4	4
26	6	10
27	4	5
28	6	10
29	5	11
30	4	10
31	3	5
32	4	8
33	3	4

Table 4-4: Number of accesses and total average access time for each user

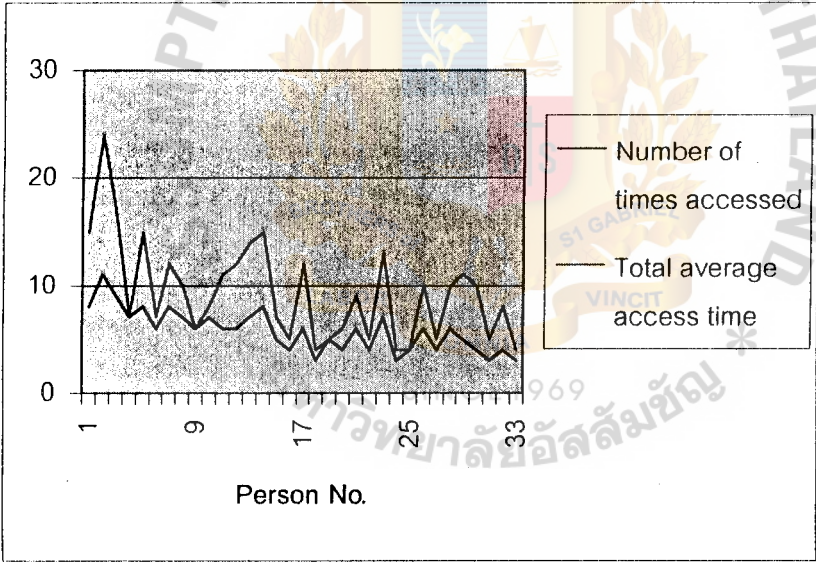


Figure 4-6: Graph showing the number of time accessed and total average access time

From this test data, it will have high a number of time accessed and high total average time in accessing at the beginning. The number of accessing and total average access time will decrease after this final test because this search engine will use the rank of each web page for arranging the web page's result. The highest rank of web

page will show first. Increasing rank will come from each selected web page result.

So this search engine will provide a good list for the user and save time in selecting.

**4.5 Comparison main features between the Proposed Search Engine and other Intranet Search Engines.**

Name	Spider	Index by	Rank by	Boolean	Using for
Inktomi Search Engine/Enterprise [16]	Yes	Match word in DB on each web page	No	Yes	Web Site
Intranet Resource Search Engine [17]	No	No	No	Yes	File Type
Orangevalley Intranet Search Engine [18]	Yes	Match word in DB on each web page	No	Yes	Web Site
Mno Search Engine software [19]	Yes	Match word in DB on each web page (Limit)	No	Yes	Web Site
ASTAware Search Key Pro V3.1 [20]	Yes	Match word in DB on each web page	No	Yes	Web Site
Proposed Search Engine	Yes	Input keyword from user	Selected URL	Yes	Web Site

Table 4-5: Comparison main features between the Proposed Search Engine and other Intranet Search Engines

This comparison, shows that most search engine software uses cut text software for creating an index. The cut text software will have a database that contains words for cutting keywords. This software will cut only the longest word and

all intranet search engines will not use rank for returning the result to user. It, therefore, could not return the good result to user.

The Proposed Search Engine will use input the keyword from the user to create an index and a selected URL to create a rank for ordering the results. From this the user can get good results and vary according to this method. The most popular web sites will show on the first page.

#### **4.6 Conclusion from these experiments**

1. The results of each search will vary depend on the rank of each web page.
2. This search engine will provide the best results on the first page (Popular web page), depending on the selected URL of user.
3. Reduce the process of cutting text, by using the user's inputted keyword of user.
4. This search engine will learn the relationship between the keyword and URL, and process the user's search results.
5. The keyword and related table will provide fast searching.

## **CHAPTER 5**

### **CONCLUSION AND FUTURE WORK**

#### **5.1 Conclusion**

The huge amount of information of the Internet, requires an effective search engine to locate and search information from web pages. The gathering process and management database of most search engines are still using manpower in processing and also don't provide the good result to the user. So we propose the Indexing of Intranet Thai Search Engines to improve the speed in searching, reduce manpower for gathering the information from each web site, creating an index and storing the keyword, support Thai and English language keyword, provide a good result list for the user, and reduce the traffic in the network of an organization because users don't need to traverse page by page of web site in searching the information. Users can search the Search Engine database in the organization's Intranet directly.

#### **5.2 Future work**

From this proposed idea, we developed this Search Engine that uses the organization's Intranet because of the limitation of hardware and network. For hardware, a search engine needs a high speed CPU for increasing speed in searching and a big storage device such as hard drive and backup. For the network, a search engine needs a lot of bandwidth and a high-speed connection for gathering the information for each web page. If without these, we cannot support a lot of users searching either. So if we have a budget for



investment with hardware and network in the future, we have a plan to upgrade this version of Search Engine that supports the Internet and develops the new searching methodology in increasing the speed of searching. We have to improve security for protecting search engine's database from viruses and hackers.



## BIBLIOGRAPHY

- [1] S. Lawrence and C.L. Giles, "Searching the World Wide Web", Science magazine, v.280, April 3, 1998 pp. 98-100
- [2],[3] Jyoti Kapoor, Web Search Engines, <http://yallara.cs.rmit.edu.au/~achattar/search-engines/definition.htm>, August 1999.
- [4] F.J. Devasason, Sheet of Deep Structure Indexing System (DSIS) Course at Asian Institute of Technology, 1997.
- [5],[6] Wes Sonnenreich and Tim Macinta, Web Developer.com, Guide to Search Engines, 1998.
- [7] Michael Bergman, Search Tutorial: Guide to Effective Searching of the Internet, The WebTools Company, May 1998.
- [8] Sullivan, D., The Major Search Engine, <http://searchengineswatch.com/facts/major.html>, October 1999.
- [9] Peggy Lee, July 28: Catcha, first Asian Portal to Offer Ask Jeeves' Popularity-Based Web Search, <http://www.catcha.co.th/catcha/press.phtml?id=200007280000>, July 2000.
- [10] Mauldin, M., Lycos: Design Choices in the Internet Search Services, IEEE Export Vol.15 No. 5 September 1997.
- [11] Bohdan O. Szuprowicz, "Search Engine Technologies for the World Wide Web and Intranets", Computer Technology Research Corporation, June 1997
- [12] William J. Stanton, Michael J. Etzel, and Bruce J. Walker, "Fundamental of Marketing", McGRAW-HILL, 1994

[13], [14] Perry Edwards, “Systems Analysis & Design”, Mitchell McGRAW-HILL, 1993

[15] Ron Soukup, and Kalen Delaney, “Inside Microsoft SQL Server 7.0”, Microsoft Press, 1999

[16] Inktomi Corporation, Inktomi Search/Enterprise,  
<http://www.inktomi.com/products/search/enterprise.html>, March 2001.

[17] Smart Infotech Systems Pvt. Ltd., Intranet Applications,  
<http://www.intrack.com/intranet/search/search.htm>, November 20, 2000.

[18] Orangevalley Web Design, ORANGEVALLEY INTRANET SEARCH ENGINE,  
[http://www.orangevalley.co.uk/search\\_engine/search.html](http://www.orangevalley.co.uk/search_engine/search.html), January 2, 2000.

[19] Lavtech.Com Corp., “mnoSearch Search Engine Software”, <http://search.mnogo.ru/>,  
March 12, 2001.

[20] ASTAware Technologies INC., Intranet Search Engine Software,  
<http://www.searchkey.com/html/products.html>, 2001.

## APPENDIX A

### QUESTIONNAIRE

This is a copy of the questionnaire. We use HTML and ASP to link with Excel file in creating this form. Each user can use a web browser to complete this questionnaire:

Questionnaire: Measurement of user's satisfaction towards the proposed search engine in organization.

Number: \_\_\_\_\_

Name: \_\_\_\_\_

Position: \_\_\_\_\_

Find the best result for you by using this keyword “สิ่ง” and answer these questions.

1. How many web pages that you select on the whole results?
2. How long do you spend before you select each web page?

Use the range of time below:

- |                                 |                                 |
|---------------------------------|---------------------------------|
| 1) Less than 30 Sec.            | 2) Between 30 Sec. and 1 Min.   |
| 3) Between 1 Min. and 1.30 Min. | 4) Between 1.30 Min. and 2 Min. |
| 5) Between 2 Min. and 2.30 Min. | 6) Between 2.30 Min. and 3 Min. |
| 7) More than 3 Min.             |                                 |

Web page No.	Time
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	

Web page No.	Time
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	
41	
42	
43	
44	
45	
46	
47	
48	
49	
50	

This is the data from questionnaires.

ID	Name	Position	1	2	3	4	5	6	7	8	9	10	11	12
1	Aneak Pokboonreung	R&D	3	2	1	2	2	2	2	1				
2	Titma Rujopakarn	Personal Manager	2	3	2	2	2	2	2	3	2	2	1	1
3	Manas Thongpat	Engineer	1	2	2	2	2	2	2	2	2			
4	Kiitipong Warasub	Engineer	1	1	1	1	1	1	1					
5	Nathaphol Phoonmart	Sales	1	2	2	2	2	2	2	2				
6	Prapat Rathlertkarn	Director	2	1	1	1	1	1						
7	Mattanee Lertchaipanon	Administrator	1	2	1	1	2	2	1	2				
8	Petai Likitchuturus	Administrator	2	2	2	1	1	1	1					
9	Kanokporn Inyod	Customer Service	1	1	1	1	1	1						
10	Thanattha Sudrak	Accounting	1	1	1	2	1	1	1					
11	Patchara Voradech	Purchasing	2	2	2	2	2	1						
12	Worawut Bupasiri	Customer Service	4	2	3	1	1	1						
13	Chaiyaporn Pornchaisiri	Engineer	2	2	2	2	2	2	2					
14	Wannapha Weeracharoen	Engineer	1	2	2	2	2	2	2	2				
15	Prawich Rungdacharoen	Engineer	1	1	1	2	2							
16	Somchai Sakulwichitsintu	Engineer	1	2	1	1								
17	Thannaporn Sanitijai	Accounting	2	2	2	2	2	2						
18	Worapan Opapan	Marketing	2	1	1									
19	Somchai Supap	Engineer	1	1	1	1	1							
20	Satit Rathlertkarn	Sales	2	2	1	1								
21	Sukanya Monprasert	Secretary	2	2	2	1	1	1						
22	Monsawat Monsawatchai	Sales	2	1	1	1								
23	Siriporn Natenarisarat	Internal Sales	2	2	1	2	2	2	2					
24	Wirat Locharoen	Engineer	2	1	1									
25	Veerasak Athornchaikul	IT Administrator	1	1	1	1								
26	Chokchai Phooprong	R&D	2	2	2	1	2	1						



27	Pornphun Siriprasert	Finance	1	1	2	1													
28	Roongthip Suwanatab	Customer Service	2	3	2	1	1	1											
29	Somboonluk Chaikomol	Customer Service	3	3	2	1	2												
30	Chalermchai Chuensombat	Customer Service	3	3	2	2													
31	Montri Suksomjit	Customer Service	2	1	2														
32	Bualee Sornthurn	Customer Service	2	2	2	2													
33	Jaroon Sangwanlek	Engineer	2	1	1														

Table A-1: Information of Questionnaire



APPENDIX B

DATA DICTIONARY

Keyword Table

Name	Data type	Size	Format
Key_Index	Integer	4	None
Keyword	Varchar	256	None

Table B-1: Data Dictionary of Keyword Table

Key fields: Key\_Index

Order of file: Indexed by Key\_Index

Relationship table: Keyword table has related with Related Table by using Key\_Index

Related Table

Name	Data type	Size	Format
Key_Index	Integer	4	None
URL_Index	Integer	4	None
Date	Datetime	8	mm/dd/yy

Table B-2: Data Dictionary of Related Table

Key fields: Key\_Index, URL\_Index

Order of file: Indexed by Key\_Index

Relationship table: Related table has related with Keyword Table by using Key\_Index and URL table by using URL\_Index

URL Table

Name	Data type	Size	Format
URL_Index	Integer	4	None
URL	Integer	256	None
Body	Varchar	8000	None
Meta	Varchar	1000	None
Title	Varchar	500	None
Rank	Integer	4	None
Date	Datetime	8	mm/dd/yy

Table B-3: Data Dictionary of URL Table

Key fields: URL\_Index

Order of file: Indexed by URL\_Index

Relationship table: URL table has related with Related Table by using URL\_Index.

APPENDIX C

NETWORK DIAGRAM

This is the network diagram that using for development search engine

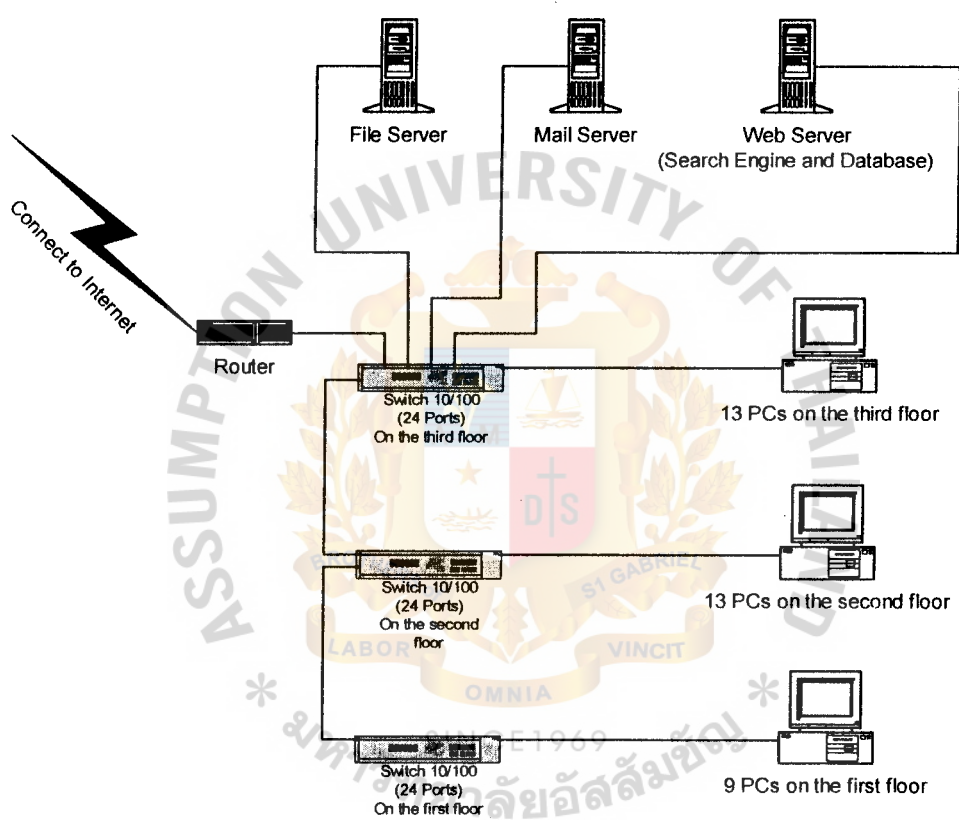


Figure C-1: Network diagram for testing Search Engine

## APPENDIX D

### SEARCH ENGINE AND SPIDER MANUAL

In the searching

1. Input keyword to text box and press “Search” Button, user can use operation “+” or “-” in searching. Example: Bangkok+Pub or Bangkok-Pub.
2. Unwanted keywords can be removed by pressing “Reset” Button.
3. After that the system will search web pages that contain the input keyword in the display screen

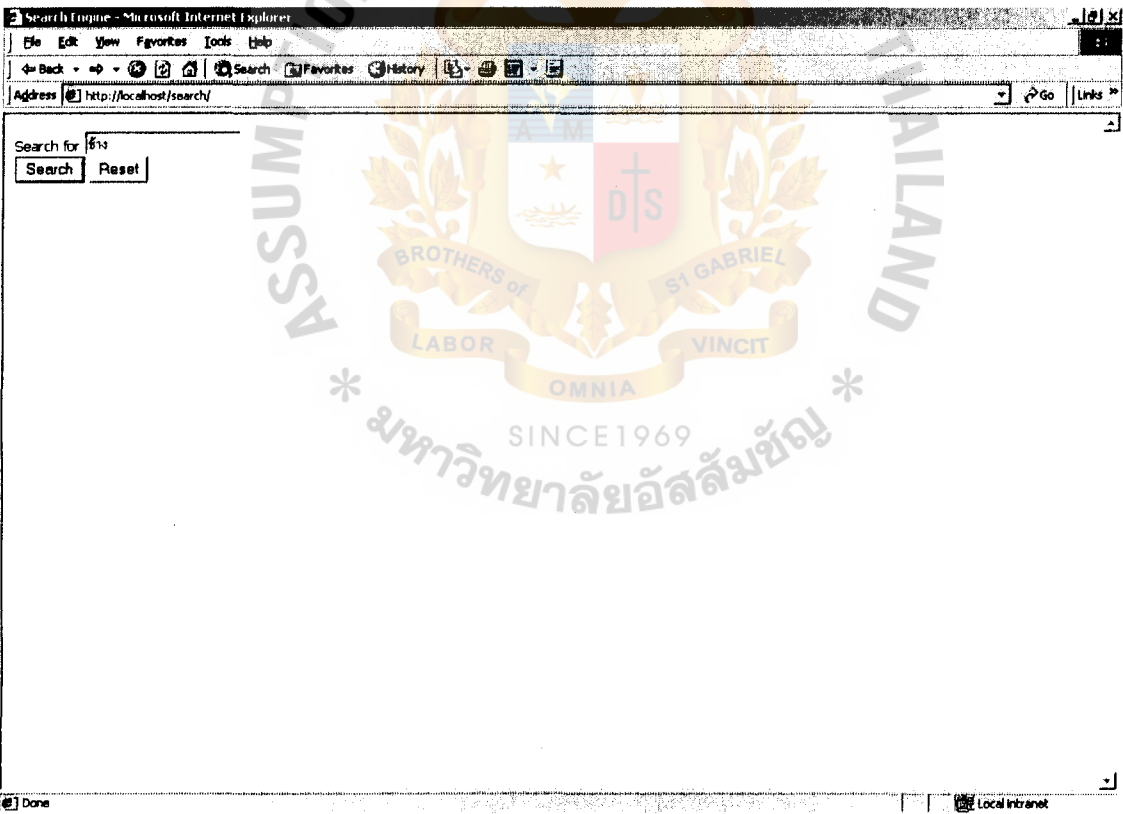


Figure D-1: Input Screen of The Proposed Search Engine

4. In display screen will show the results from searching, each page of display screen contains 10 web page's results. If the result has more than 10 web pages, it will have the next page number at the bottom of the display screen.

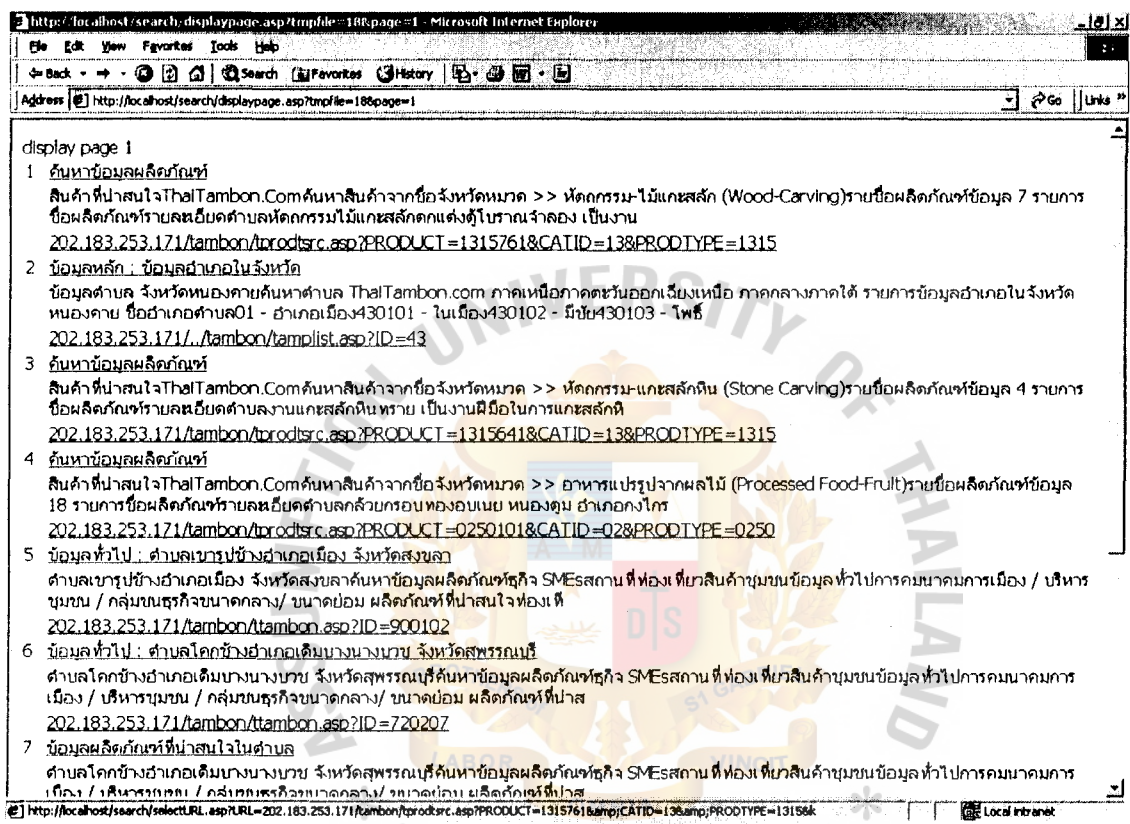


Figure D-2: Display Screen of The Proposed Search Engine



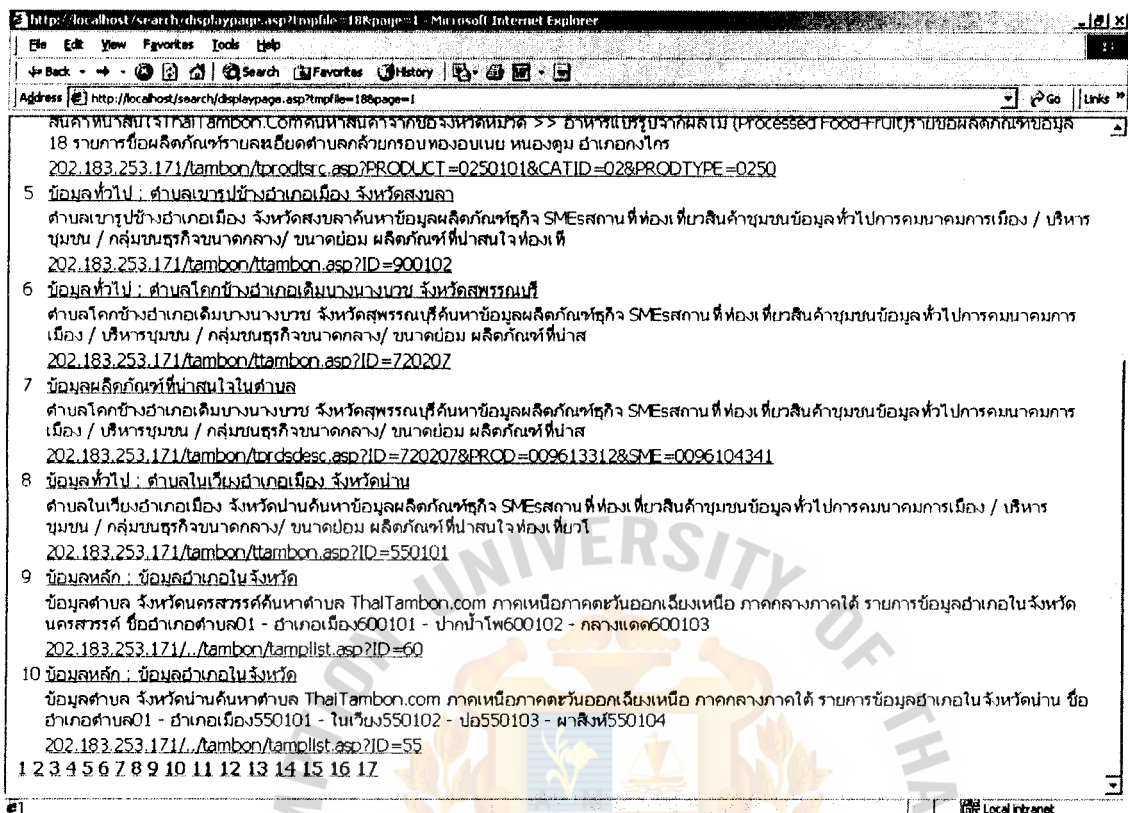


Figure D-3: Display Screen of The Proposed Search Engine (show page number)

In using spider for gather information for web site

1. Input IP Address or web site name into text box behind http:// word and press “Enter” for gathering information in 1 web site.
2. Administrator can gather information in the range of IP addresses by input first and last IP Address in text box.
3. For Delete function, Administrator has to input the date into delete date text box. The system will delete the record that has the previous input’s date.
4. For Name > IP, it is used for checking IP Address of each web site by input web site name and press “Name > IP” button, the system will show IP Address of this web site.

5. For IP > Name, it use for checking web site name of each IP Address by input IP Address and press “IP > Name” button, the system will show web site name of this IP Address.

Spider

http://localhost

From2091261631End209126163253

Go

Title

HTML

RawData

Date mm/dd/yy

Delete Old URL

Name->IP

IP->Name

BODY

Result

Header

Header

Figure D-4: Spider Interface

## APPENDIX E

### SPIDER'S SOURCE CODE

#### 1. Main Form

This is the form for input web address or IP address.

Public currentURL As String, Running As Boolean

Dim src As String

Dim arrLinks() As String

Dim arrGoneLinks() As Boolean

Dim currentSite As String

Dim lngLink As Long

Dim skipURL() As String

Dim UpToDate As Boolean

Dim curTime As Date

Private Sub btnName\_Click()

Dim strIP As String

strIP = txtIP.Text

strIP = DNS1.NameToAddress(strIP)

MsgBox strIP

txtIP.Text = DNS1.NameToAddress(txtIP.Text)

End Sub

```
Private Sub cbGo_KeyPress(KeyAscii As Integer)
```

```
Dim oldtime As Date
```

```
Initial
```

```
If KeyAscii = vbKeyReturn Then
```

```
    oldtime = Now
```

```
    A = Format(oldtime, "HH,MM,SS")
```

```
    Debug.Print A
```

```
    If cbGo.Text = "" Then
```

```
        Exit Sub
```

```
    End If
```

```
    If InStr(cbGo.Text, "/") Then
```

```
        currentSite = Left$(cbGo.Text, InStr(cbGo.Text, "/") - 1)
```

```
    Else
```

```
        currentSite = cbGo.Text
```

```
    End If
```

```
    cbGo.AddItem (cbGo.Text)
```

```
    GetLinks (name2ipURL(cbGo.Text))
```

```
    A = Format(oldtime - Now, "HH:MM:SS")
```

```
    Debug.Print A
```

```
    Beep
```

```
    Running = False
```

```
    btnGo.Caption = "Go"
```

```

ccurTime = curTime - Now

MsgBox A

End If

End Sub

Private Sub cmdAddDB_Click()

Dim RS As New ADODB.Recordset

With RS

.ActiveConnection = "Provider=SQLOLEDB.1;User ID=sa;" & _
    "Initial Catalog=searchdb;" & _
    "Data Source=THESIS;"

.CursorLocation = adUseServer

.CursorType = adOpenKeyset

.LockType = adLockPessimistic

.Source = "SELECT * FROM URL"

.Open

If .State = adStateOpen Then

.AddNew

.Fields("URL") = "www.chula.ac.th"

.Fields("Date") = "12/30/01"

.Update

MsgBox "success"

Else

```

```

        MsgBox "not success"

    End If

    .Close

End With

Set RS = Nothing

End Sub

Private Sub cmdCreateDB_Click()

    Dim MyWs As Workspace

    Dim MyDB As Database

    Dim MyTable As TableDef

    Dim MyField(4) As Field

    Dim MyIndex As Index

    Dim MyIndexField As Field

    Set MyWs = DBEngine.Workspaces(0)

    Set MyDB = MyWs.CreateDatabase(App.Path & "\MyDb.mdb", dbLangGeneral,
    dbVersion25)

    Set MyTable = MyDB.CreateTableDef("Customers")

    Set MyField(0) = MyTable.CreateField("CUSTID", dbLong)

    MyField(0).Attributes = dbAutoIncrField

    Set MyField(1) = MyTable.CreateField("Name", dbText)

    MyField(1).Size = 50

    MyField(1).AllowZeroLength = True

```



```

Set MyField(2) = MyTable.CreateField("Address", dbText)

MyField(2).Size = 50

MyField(2).AllowZeroLength = True

Set MyField(3) = MyTable.CreateField("Credit", dbCurrency)

MyField(3).DefaultValue = 0

Set MyField(4) = MyTable.CreateField("RunningNo", dbLong)

MyField(4).DefaultValue = 0

MyTable.Fields.Append MyField(0)

MyTable.Fields.Append MyField(1)

MyTable.Fields.Append MyField(2)

MyTable.Fields.Append MyField(3)

MyTable.Fields.Append MyField(4)

MyDB.TableDefs.Append MyTable

Set MyIndex = MyTable.CreateIndex("CUSTID")

MyIndex.Primary = True

MyIndex.Unique = True

Set MyIndexField = MyIndex.CreateField("CUSTID")

MyIndex.Fields.Append MyIndexField

MyTable.Indexes.Append MyIndex

MyDB.Close

MsgBox ("Database Created")

End Sub

```

Private Sub cmdLook\_Click()

txtIP.Text = DNS1.AddressToName(txtIP.Text)

End Sub

Private Sub btnGo\_Click()

Dim oldtime As Date

If Running Then

Running = False

btnGo.Caption = "Go"

Exit Sub

End If

Dim F1, F2, F3, F4, E1, E2, E3, E4 As Integer

oldtime = Now

A = Format(oldtime, "HH,MM,SS")

Debug.Print A

F1 = Val(txtF1.Text)

F2 = Val(txtF2.Text)

F3 = Val(txtF3.Text)

F4 = Val(txtF4.Text)

E1 = Val(txtE1.Text)

E2 = Val(txtE2.Text)

E3 = Val(txtE3.Text)

E4 = Val(txtE4.Text)

Do

strURL\$ = F1 & "." & F2 & "." & F3 & "." & F4

currentSite = strURL\$

Initial

GetLinks (strURL\$)

If (F1 = E1) And (F2 = E2) And (F3 = E3) And (F4 = E4) Then

Exit Do

End If

If (F4 = 253) Then

F4 = 1

If (F3 = 255) Then

F3 = 1

If (F2 = 255) Then

F2 = 1

If (F1 = 253) Then

Exit Do

Else

F1 = F1 + 1

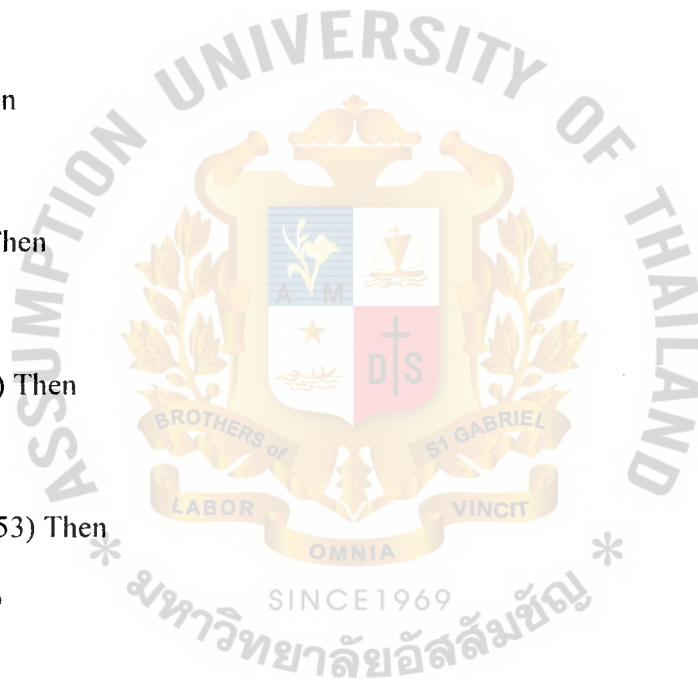
End If

Else

F2 = F2 + 1

End If

Else



F3 = F3 + 1

End If

Else

F4 = F4 + 1

End If

Loop While Running

A = Format(oldtime - Now, "HH:MM:SS")

Debug.Print A

Beep

Running = False

btnGo.Caption = "Go"

curTime = curTime - Now

MsgBox A

End Sub

Private Sub Navigator1(strURL As String)

Dim LastModified As Date, lastModifiedOld As Date

Dim haveOldVersion As Boolean, DBsuccess As Boolean

UptoDate = False

HtmlText.Text = "Waiting for " & strURL\$

WebText.Text = "Waiting for " & strURL\$

HeadText.Text = "Waiting for " & strURL\$

UptoDate = False

Err.Clear

Inet1.Protocol = icHTTP

On Error GoTo myErrorHandler

HtmlText.Text = Inet1.OpenURL(strURL)

If Not Running Then

clearOutput

Exit Sub

End If

If (HtmlText.Text <> "") Then

If documentOK(Inet1.GetHeader) Then

LastModified = FormatDateTime(Inet1.GetHeader)

If IsURLExist(strURL, lastModifiedOld) Then

haveOldVersion = True

If LastModified = lastModifiedOld Then

UptoDate = True

Else

End If

End If

WebText.Text = HTML2Text(HtmlText.Text)

WebText.Text = HTMLtoText(WebText.Text)

WebText.Text = Convert2Space(WebText.Text)

txtTitle.Text = GetTitle(HtmlText.Text)

HeadText.Text = GetHead(HtmlText.Text)

HeadText.Text = GetMeta(HeadText.Text)

WebText.Text = Left\$(WebText.Text, 8000)

txtTitle.Text = Left\$(txtTitle.Text, 500)

HeadText.Text = Left\$(HeadText.Text, 1000)

If Not UpToDate Then

If haveOldVersion Then

UpdateDB strURL, LastModified, WebText.Text, txtTitle.Text, HeadText.Text

Else

DBsuccess = addNewDBfnc(strURL, LastModified, WebText.Text,  
txtTitle.Text, HeadText.Text)

End If

End If

End If

Else

End If

Exit Sub

myErrorHandle:

clearOutput

StatusBar1.SimpleText = "error handler for " & strURL

End Sub

Private Sub DelOldURL\_Click()

Dim oldDate As Date, oldDateStr As String

Dim Confirm As VbMsgBoxResult

Dim warningStr As String

If IsDate(txtOldDate.Text) Then

oldDate = CDate(txtOldDate.Text)

oldDateStr = CmmddyyDate(str(oldDate))

If oldDateStr <> "" Then

warningStr = "All URL that older than " & oldDateStr & " will be deleted" & \_

Chr(13) & Chr(13) & "Are you sure you want to delete them?"

Confirm = MsgBox(warningStr, vbQuestion + vbYesNo)

If Confirm = vbYes Then

StatusBar1.SimpleText = "Deletion is in process..."

DelOldData (oldDateStr)

StatusBar1.SimpleText = ""

MsgBox "deletion completed"

Else

MsgBox "deletion canceled"

End If

Else

MsgBox "Please enter date format as mm/dd/yy"

End If

Else

MsgBox "Please enter date format as mm/dd/yy"

End If



End Sub

Private Sub Form\_Load()

ReDim arrLinks(0)

ReDim arrGoneLinks(0)

Running = False

ReDim skipURL(19)

skipURL(0) = ".gif"

skipURL(1) = ".jpg"

skipURL(2) = ".ps"

skipURL(3) = ".pdf"

skipURL(4) = "mailto:"

skipURL(5) = ".zip"

skipURL(6) = ".gz"

skipURL(7) = ".exe"

skipURL(8) = ".ra"

skipURL(9) = ".ram"

skipURL(10) = ".bmp"

skipURL(11) = ".all"

skipURL(12) = ".mpg"

skipURL(13) = ".mpeg"

skipURL(14) = ".qt"

skipURL(15) = ".mov"



skipURL(16) = ".mp3"

skipURL(17) = ".avi"

skipURL(18) = ".jpeg"

skipURL(19) = ".midi"

End Sub

Public Sub Inet1\_StateChanged(ByVal State As Integer)

Select Case State

Case 0

StatusBar1.SimpleText = " Idle..."

Case 1

StatusBar1.SimpleText = " Resolving host..."

Case 2

StatusBar1.SimpleText = " Host resolved."

Case 3

StatusBar1.SimpleText = " Connecting..."

Case 4

StatusBar1.SimpleText = " Connected."

Case 5

StatusBar1.SimpleText = " Requesting..."

Case 6

StatusBar1.SimpleText = " Request sent."

Case 7

```

        strURL$ = currentURL & "/" & strURL$

    End If

End If

Else

    strURL$ = Mid$(strURL$, Len("http://") + 1)

End If

strURL$ = name2ipURL(strURL$)

If Not Known(strURL$) Then 'check repeated URL
    If SameSite(strURL$, currentURL) Then
        lstLinks.AddItem (strURL$)

        ReDim Preserve arrLinks(UBound(arrLinks) + 1) As String
        ReDim Preserve arrGoneLinks(UBound(arrGoneLinks) + 1) As Boolean
        arrLinks(UBound(arrLinks)) = strURL$
    End If
End If

End If

nextLoop:

    Loop While Running

    LinksFromHTML = UBound(arrLinks) + 1

End Function

Private Sub lstLinks_DblClick()

    cbGo.Text = lstLinks.Text

```

```
cbGo_KeyPress (vbKeyReturn)
```

```
End Sub
```

```
Private Function Known(ByVal strURL As String) As Boolean
```

```
Dim i As Integer, j As Integer
```

```
For j = 0 To UBound(skipURL)
```

```
    If InStr(1, strURL, skipURL(j), vbTextCompare) Then
```

```
        Known = True
```

```
        Exit Function
```

```
    End If
```

```
Next j
```

```
For i = 0 To UBound(arrLinks)
```

```
    If arrLinks(i) = strURL Then
```

```
        Known = True
```

```
        Exit Function
```

```
    End If
```

```
Next i
```

```
Known = False
```

```
End Function
```

```
Private Function SameSite(ByVal URL1 As String, ByVal URL2 As String) As Boolean
```

```
Dim Name1 As String, Name2 As String
```

```
Dim IP1 As String, IP2 As String
```

```

If InStr(URL1, "/") Then

    Name1 = Left$(URL1, InStr(URL1, "/") - 1)

Else

    Name1 = URL1

End If

If InStr(URL2, "/") Then

    Name2 = Left$(URL2, InStr(URL2, "/") - 1)

Else

    Name2 = URL2

End If

If Name1 = Name2 Then

    SameSite = True

    Exit Function

End If

IP1 = DNS1.NameToAddress(Name1)

IP2 = DNS1.NameToAddress(Name2)

If IP1 = IP2 Then

    SameSite = True

Else

    SameSite = False

End If

End Function

```

Private Function IPbound(strIP As String, lowerBound As Integer, upperBound As Integer)

As String

Dim subIP As Integer

subIP = Val(strIP)

If subIP < lowerBound Then

IPbound = Trim(str\$(lowerBound))

ElseIf subIP > upperBound Then

IPbound = Trim(str\$(upperBound))

Else

IPbound = strIP

End If

End Function

Private Sub txtE1\_LostFocus()

txtE1.Text = IPbound(txtE1.Text, 1, 253)

End Sub

Private Sub txtE2\_LostFocus()

txtE2.Text = IPbound(txtE2.Text, 0, 255)

End Sub

Private Sub txtE3\_LostFocus()

txtE3.Text = IPbound(txtE3.Text, 0, 255)

End Sub

Private Sub txtE4\_LostFocus()

txtE4.Text = IPbound(txtE4.Text, 1, 253)

End Sub

Private Sub txtF1\_LostFocus()

txtF1.Text = IPbound(txtF1.Text, 1, 253)

End Sub

Private Sub txtF2\_LostFocus()

txtF2.Text = IPbound(txtF2.Text, 0, 255)

End Sub

Private Sub txtF3\_LostFocus()

txtF3.Text = IPbound(txtF3.Text, 0, 255)

End Sub

Private Sub txtF4\_LostFocus()

txtF4.Text = IPbound(txtF4.Text, 1, 253)

End Sub

Public Function name2ipURL(ByVal URLname As String) As String



```

Dim slash As Long

Dim URLip As String

slash = InStr(URLname, "/")

If slash Then

    URLip = Left$(URLname, slash - 1)

Else

    URLip = URLname

End If

If URLip <> "" Then

    URLip = DNS1.NameToAddress(URLip)

End If

If URLip = "" Then

    URLip = URLname

ElseIf slash Then

    URLip = URLip & Mid$(URLname, slash)

End If

name2ipURL = URLip

End Function

Private Sub Initial()

    curTime = Now

    ReDim arrLinks(0)

    ReDim arrGoneLinks(0)

```

```

Running = True

btnGo.Caption = "Stop"

lstLinks.Clear

End Sub

```

## 2. Navigator Form

This is the form for requesting html and check URL in database.

```

Private Sub Form_Load()

    Dim LastModified As Date, lastModifiedOld As Date

    Dim haveOldVersion As Boolean, DBsuccess As Boolean

    Dim UptoDate As Boolean

    Dim strURL As String

    strURL = Form1.currentURL

    With Form1

        UptoDate = False

        .HtmlText.Text = "Waiting for " & strURL

        .WebText.Text = "Waiting for " & strURL

        .HeadText.Text = "Waiting for " & strURL

        .StatusBar1.SimpleText = ""

        On Error GoTo myErrorHandle

        Inet1.Protocol = icHTTP

        .HtmlText.Text = Inet1.OpenURL(strURL)

```

If Not .Running Then

clearOutput

Exit Sub

End If

If (.HtmlText.Text <> "") Then

If documentOK(Inet1.GetHeader) Then

LastModified = FormatDateTime(Inet1.GetHeader)

If IsURLExist(strURL, lastModifiedOld) Then

haveOldVersion = True

If LastModified = lastModifiedOld Then

UptoDate = True

End If

End If

.WebText.Text = HTML2Text(.HtmlText.Text)

.WebText.Text = HTMLtoText(.WebText.Text)

.WebText.Text = Convert2Space(.WebText.Text)

.txtTitle.Text = GetTitle(.HtmlText.Text)

.HeadText.Text = GetHead(.HtmlText.Text)

.HeadText.Text = GetMeta(.HeadText.Text)

.WebText.Text = Left\$(.WebText.Text, 8000)

.txtTitle.Text = Left\$(.txtTitle.Text, 500)

.HeadText.Text = Left\$(.HeadText.Text, 1000)

```

If Not UptoDate Then
    If haveOldVersion Then
        UpdateDB strURL, LastModified, Form1.WebText.Text, Form1.txtTitle.Text,
Form1.HeadText.Text
    Else
        DBsuccess = addNewDBfnc(strURL, LastModified, Form1.WebText.Text,
Form1.txtTitle.Text, Form1.HeadText.Text)
    End If
End If
End If
Else
End If
Exit Sub
myErrorHandler:
    clearOutput
    Form1.StatusBar1.SimpleText = "error handler for " & strURL
End Sub

Private Sub Inet1_StateChanged(ByVal State As Integer)

    MsgBox State

    With Form1

        Select Case State

```

Case 0

`.StatusBar1.SimpleText = " Idle..."`

Case 1

`.StatusBar1.SimpleText = " Resolving host..."`

Case 2

`.StatusBar1.SimpleText = " Host resolved."`

Case 3

`.StatusBar1.SimpleText = " Connecting..."`

Case 4

`.StatusBar1.SimpleText = " Connected."`

Case 5

`.StatusBar1.SimpleText = " Requesting..."`

Case 6

`.StatusBar1.SimpleText = " Request sent."`

Case 7

`.StatusBar1.SimpleText = " Receiving response..."`

Case 8

`.StatusBar1.SimpleText = " Response received..."`

Case 9

`.StatusBar1.SimpleText = " Disconnecting..."`

Case 10

`.StatusBar1.SimpleText = " Disconnected."`

Case 11

```
.StatusBar1.SimpleText = " Error receiving response."
```

```
Case 12
```

```
.StatusBar1.SimpleText = " Transfer complete."
```

```
Case Else
```

```
.StatusBar1.SimpleText = " Unknown state."
```

```
End Select
```

```
End With
```

```
End Sub
```

### 3. HtmlConvert Module

This is the module for converting the html symbol to real symbol and real character.

```
Public Function HTMLtoText(ByVal OrigHTML$) As String
```

```
On Error Resume Next
```

```
If InStr(LCase$(OrigHTML$), "<body") > 0 Then
```

```
OrigHTML$ = Mid$(OrigHTML$, InStr(LCase$(OrigHTML$), "<body"))
```

```
OrigHTML$ = Mid$(OrigHTML$, InStr(OrigHTML$, ">") + 1)
```

```
If InStr(LCase$(OrigHTML$), "</body>") > 0 Then _
```

```
OrigHTML$ = Left$(OrigHTML$, InStr(LCase$(OrigHTML$), "</body>") - 1)
```

```
End If
```

```
Do While Len(OrigHTML$)
```

```
CurrChar$ = Left$(OrigHTML$, 1)
```

```
OrigHTML$ = Mid$(OrigHTML$, 2)
```

Select Case CurrChar\$

Case " "

OrigHTML\$ = LTrim\$(OrigHTML\$)

Case vbCr, vbLf

CurrChar\$ = ""

If Left\$(OrigHTML\$, 1) = vbLf Then OrigHTML\$ = Mid\$(OrigHTML\$, 2)

OrigHTML\$ = LTrim\$(OrigHTML\$)

Case "<"

CurrChar\$ = ""

If InStr(OrigHTML\$, ">") > 0 Then

CurrChar\$ = Left\$(OrigHTML\$, InStr(OrigHTML\$, ">") - 1)

OrigHTML\$ = Mid\$(OrigHTML\$, InStr(OrigHTML\$, ">") + 1)

Select Case LCase\$(CurrChar\$)

Case "p", "/div"

CurrChar\$ = vbCrLf + vbCrLf

Case "br"

CurrChar\$ = vbCrLf

Case Else

CurrChar\$ = ""

End Select

End If

Case "&"



If InStr(OrigHTML\$, ";") > 0 And InStr(OrigHTML\$, ";") < InStr(OrigHTML\$, " ")

Then

CurrChar\$ = Left\$(OrigHTML\$, InStr(OrigHTML\$, ";") - 1)

OrigHTML\$ = Mid\$(OrigHTML\$, InStr(OrigHTML\$, ";") + 1)

Select Case CurrChar\$

Case "amp"

CurrChar\$ = "&"

Case "quot"

CurrChar\$ = ""

Case "lt"

CurrChar\$ = "<"

Case "gt"

CurrChar\$ = ">"

Case "nbsp"

CurrChar\$ = " "

Case "Auml"

CurrChar\$ = "ä"

Case "auml"

CurrChar\$ = "ä"

Case "iexcl"

CurrChar\$ = "í"

Case "cent"

CurrChar\$ = "¢"

Case "pound"

CurrChar\$ = "?"

Case "curren"

CurrChar\$ = "?"

Case "yen"

CurrChar\$ = "?"

Case "brybar"

CurrChar\$ = "|"

Case "sect"

CurrChar\$ = "?"

Case "uml"

CurrChar\$ = "?"

Case "copy"

CurrChar\$ = "?"

Case "ordf"

CurrChar\$ = "?"

Case "laquo"

CurrChar\$ = "?"

Case "not"

CurrChar\$ = "?"

Case "reg"

CurrChar\$ = "?"

Case "macr"



CurrChar\$ = "?"

Case "deg"

CurrChar\$ = "?"

Case "plusm"

CurrChar\$ = "?"

Case "sup2"

CurrChar\$ = "?"

Case "sup3"

CurrChar\$ = "?"

Case "acute"

CurrChar\$ = "?"

Case "micro"

CurrChar\$ = "?"

Case "para"

CurrChar\$ = "?"

Case "middot"

CurrChar\$ = "?"

Case "cedil"

CurrChar\$ = "?"

Case "sup1"

CurrChar\$ = "?"

Case "ordm"

CurrChar\$ = "?"



Case "raquo"

CurrChar\$ = "?"

Case "frac14"

CurrChar\$ = "?"

Case "frac12"

CurrChar\$ = "?"

Case "frac34"

CurrChar\$ = "?"

Case "iquest"

CurrChar\$ = "?"

Case "Agrave"

CurrChar\$ = "?"

Case "Aacute"

CurrChar\$ = "?"

Case "Acirc"

CurrChar\$ = "?"

Case "Atilde"

CurrChar\$ = "?"

Case "Aring"

CurrChar\$ = "?"

Case "AElig"

CurrChar\$ = "?"

Case "Ccedil"



CurrChar\$ = "?"

Case "Egrave"

CurrChar\$ = "?"

Case "Eacute"

CurrChar\$ = "?"

Case "Ecirc"

CurrChar\$ = "?"

Case "Euml"

CurrChar\$ = "?"

Case "Igrave"

CurrChar\$ = "?"

Case "Iacute"

CurrChar\$ = "?"

Case "Icirc"

CurrChar\$ = "?"

Case "Iuml"

CurrChar\$ = "?"

Case "ETH"

CurrChar\$ = "?"

Case "Ntilde"

CurrChar\$ = "?"

Case "Ograve"

CurrChar\$ = "?"



Case "Oacute"

CurrChar\$ = "?"

Case "Ocirc"

CurrChar\$ = "?"

Case "Otilde"

CurrChar\$ = "?"

Case "Ouml"

CurrChar\$ = "?"

Case "times"

CurrChar\$ = "?"

Case "Oslash"

CurrChar\$ = "?"

Case "Ugrave"

CurrChar\$ = "?"

Case "Uacute"

CurrChar\$ = "?"

Case "Ucirc"

CurrChar\$ = "?"

Case "Uuml"

CurrChar\$ = "?"

Case "Yacute"

CurrChar\$ = "?"

Case "THORN"



CurrChar\$ = "?"

Case "szlig"

CurrChar\$ = "?"

Case "agrave"

CurrChar\$ = "?"

Case "aacute"

CurrChar\$ = "?"

Case "acirc"

CurrChar\$ = "?"

Case "atilde"

CurrChar\$ = "?"

Case "aring"

CurrChar\$ = "?"

Case "aelig"

CurrChar\$ = "?"

Case "ccedil"

CurrChar\$ = "?"

Case "egrave"

CurrChar\$ = "?"

Case "eacute"

CurrChar\$ = "?"

Case "ecirc"

CurrChar\$ = "?"





Case "euml"

CurrChar\$ = "?"

Case "igrave"

CurrChar\$ = "?"

Case "iacute"

CurrChar\$ = "?"

Case "icirc"

CurrChar\$ = "?"

Case "iuml"

CurrChar\$ = "?"

Case "eth"

CurrChar\$ = "?"

Case "ntilde"

CurrChar\$ = "?"

Case "ograve"

CurrChar\$ = "?"

Case "oacute"

CurrChar\$ = "?"

Case "ocirc"

CurrChar\$ = "?"

Case "otilde"

CurrChar\$ = "?"

Case "ouml"



CurrChar\$ = "?"

Case "divide"

CurrChar\$ = "?"

Case "oslash"

CurrChar\$ = "?"

Case "ugrave"

CurrChar\$ = "?"

Case "uacute"

CurrChar\$ = "?"

Case "ucirc"

CurrChar\$ = "?"

Case "uuml"

CurrChar\$ = "?"

Case "yacute"

CurrChar\$ = "?"

Case "thorn"

CurrChar\$ = "?"

Case "yuml"

CurrChar\$ = "?"

Case Else

CurrChar\$ = "&" + CurrChar\$ + ";"

End Select

End If



End Select

NoHTML\$ = NoHTML\$ + CurrChar\$

Loop

HTMLtoText = NoHTML\$

End Function

#### 4. Text Filter Module

This is the module for removing html tag.

Option Explicit

Public Sub FindHTMLTag(ByVal MainStr As String, ByRef s, ByRef l, Optional ByVal ss

As Long = 0)

Dim sl, ll As Long

If IsNull(MainStr) Or Len(MainStr) < 3 Then

s = 0

l = -1

Exit Sub

End If

sl = InStr(ss + 1, MainStr, "<")

ll = InStr(sl + 1, MainStr, ">")

If sl = 0 Or ll = 0 Then

s = 0

```

    l = -2

    Exit Sub

End If

s = sl - 1

l = ll - s

End Sub

Public Sub FindSHTMLTag(ByVal MainStr As String, ByVal TagStr As String, _
    ByRef s, ByRef l, Optional ByVal ss As Long = 0)
    Dim sl, s2, l2 As Long
    Dim str1, str2 As String
    If IsNull(MainStr) Or Len(MainStr) < 3 Or _
        IsNull(TagStr) Or Len(TagStr) = 0 Then
        s = 0
        l = -1
        Exit Sub
    End If
    str1 = "<" + TagStr
    str2 = "</" + TagStr
    s1 = InStr(ss + 1, MainStr, str1, vbTextCompare)
    s2 = InStr(s1 + 1, MainStr, str2, vbTextCompare)
    If s1 = 0 Or s2 = 0 Then
        s = 0

```

```
l = -2

Exit Sub

End If

l2 = InStr(s2, MainStr, ">", vbTextCompare)

If l2 = 0 Or IsNull(l2) Then

    s = 0

    l = -2

    Exit Sub

End If

s = s1 - 1

l = l2 - s

End Sub

Public Function NormalizeTags(ByVal str As String) As String

    str = CleanDoubleSpaces(str)

    Do While InStr(1, str, "< ") > 0

        str = Replace(str, "< ", "<")

    Loop

    NormalizeTags = str

End Function

Public Function Convert2Space(ByVal str As String) As String

    Dim cspace, ctab, clinefeed, creturn, cc As String
```

```

cspace = Chr(32)

ctab = Chr(9)

clinefeed = Chr(10)

creturn = Chr(13)

str = ReplaceAscii(str, ctab, cspace)

str = ReplaceAscii(str, creturn, cspace)

str = CleanDoubleSpaces(str)

cc = cspace + clinefeed

str = ReplaceAscii(str, cc, clinefeed)

cc = clinefeed + cspace

str = ReplaceAscii(str, cc, clinefeed)

str = CleanRepeatedAscii(str, 10)

Convert2Space = str

```

End Function

Public Function Convert2SingleLine(ByVal str As String) As String

```

Dim cspace, ctab, clinefeed, creturn, cc As String

```

```

cspace = Chr(32)

ctab = Chr(9)

clinefeed = Chr(10)

creturn = Chr(13)

str = ReplaceAscii(str, ctab, cspace)

str = ReplaceAscii(str, clinefeed, cspace)

```

```

str = ReplaceAscii(str, creturn, cspace)

str = CleanDoubleSpaces(str)

If Left(str, 1) = " " Then

    str = Right(str, Len(str) - 1)

End If

Convert2SingleLine = str

End Function

Private Function ReplaceAscii(ByVal str As String, ByVal f As String, _
    ByVal r As String) As String
    Dim ss As Long
    If IsNull(str) Or IsEmpty(str) Then
        ReplaceAscii = str
        Exit Function
    End If
    Do While True
        ss = InStr(str, f)
        If ss > 0 Then
            str = Replace(str, f, r)
        Else
            Exit Do
        End If
    Loop

```



```
ReplaceAscii = str
End Function

Public Function DeleteString(ByVal str As String, ByVal s As Long, ByVal l As Long) As
String
    If l = 0 Then
        DeleteString = str
        Exit Function
    End If
    DeleteString = Left(str, s) + Right(str, Len(str) - s - l)
End Function

Public Function CleanHTMLTags(ByVal str As String) As String
    Dim s, l As Long
    str = NormalizeTags(str)
    If IsNull(str) Then
        str = "no text to clean from HTML tags"
        Exit Function
    End If
    Do While l >= 0
        str = DeleteString(str, s, l)
        FindHTMLTag str, s, l
    Loop
```

str = CleanDoubleSpaces(str)

CleanHTMLTags = str

End Function

Public Function CleanSHTMLTags(ByVal MainStr As String, ByVal TagStr As String) As

String

Dim s, l As Long

If IsNull(MainStr) Then

MainStr = "no text to clean from HTML tags"

Exit Function

End If

MainStr = NormalizeTags(MainStr)

Do While l >= 0

MainStr = DeleteString(MainStr, s, l)

FindSHTMLTag MainStr, TagStr, s, l

Loop

MainStr = CleanDoubleSpaces(MainStr)

CleanSHTMLTags = MainStr

End Function

Public Function CleanRepeatedAscii(ByVal str As String, ByVal ac As Long) As String

Dim s, l, ss, ls As Long

Dim ch, ch2 As String

ch = Chr(ac)

ch2 = ch + ch

If IsNull(str) Or IsEmpty(str) Then

str = "no text to clean from specified Ascii code"

Exit Function

End If

Do While True

ss = InStr(str, ch2)

If ss > 0 Then

str = Replace(str, ch2, ch)

Else

Exit Do

End If

Loop

CleanRepeatedAscii = str

End Function

Public Function CleanDoubleSpaces(ByVal str As String) As String

str = CleanRepeatedAscii(str, "32")

CleanDoubleSpaces = str

End Function

Public Function HTML2Text(ByVal str As String) As String



```

str = NormalizeTags(str)

str = CleanSHTMLTags(str, "head")

str = CleanSHTMLTags(str, "script")

str = CleanHTMLTags(str)

str = CleanRepeatedAscii(str, 13)

str = CleanRepeatedAscii(str, 10)

str = CleanRepeatedAscii(str, 9)

str = CleanDoubleSpaces(str)

HTML2Text = str

End Function

Public Function GetStringBetween(ByVal str As String, ByVal str1 As String, ByVal str2 As
String, _
Optional ByVal st As Long = 0) As String

Dim s1, s2, s, l As Long
Dim foundstr As String
s1 = InStr(st + 1, str, str1, vbTextCompare)
s2 = InStr(s1 + 1, str, str2, vbTextCompare)
If s1 = 0 Or s2 = 0 Or IsNull(s1) Or IsNull(s2) Then

    foundstr = str

Else

    s = s1 + Len(str1)

    l = s2 - s

```

```
foundstr = Mid(str, s, l)
```

```
End If
```

```
GetStringBetween = foundstr
```

```
End Function
```

```
Public Function GetTitle(ByVal str As String) As String
```

```
Dim title As String
```

```
str = NormalizeTags(str)
```

```
title = GetStringBetween(str, "<title>", "</title>")
```

```
If title = str Then
```

```
GetTitle = "No Title Detected"
```

```
Else
```

```
GetTitle = title
```

```
End If
```

```
End Function
```

```
Public Function GetHead(ByVal str As String) As String
```

```
Dim head As String
```

```
str = NormalizeTags(str)
```

```
head = GetStringBetween(str, "<head>", "</head>")
```

```
If head = str Then
```

```
GetHead = "No Header Detected"
```

```
Else
```

GetHead = head

End If

End Function

Public Function GetMeta(ByVal HeadStr As String) As String

Dim metaSpot As Long, endSpot As Long

Dim result As String

Do

metaSpot = InStr(metaSpot + 1, HeadStr, "<")

If metaSpot Then

HeadStr = LTrim(Mid(HeadStr, metaSpot + 1))

If 1 = InStr(1, HeadStr, "meta", vbTextCompare) Then

If (InStr(1, HeadStr, "description", vbTextCompare) Or

InStr(1, HeadStr, "keyword", vbTextCompare)) Then

endSpot = InStr(HeadStr, ">")

If endSpot Then

result = result & Left(HeadStr, endSpot - 1)

Else

result = result & HeadStr

End If

End If

End If

End If

```

Loop While metaSpot

GetMeta = result

End Function

```

```

Public Sub clearOutput()

With Form1

    .HtmlText.Text = ""

    .WebText.Text = ""

    .HeadText.Text = ""

End With

End Sub

```

## 5. Timinganddatabase Module

This is the module for adding and removing data on the database by using date factor.

```

Public Function FormatDateTime(ByVal strHeader As String) As Date

Dim mystrTime As String

Dim foundIndex As Double

foundIndex = InStr(1, strHeader, "Last-Modified", vbTextCompare)

If foundIndex Then

    foundIndex = foundIndex + Len("Last-Modified: ") + 5

    mystrTime = Mid$(strHeader, foundIndex)

    mystrTime = Left$(mystrTime, InStr(mystrTime, " GMT"))

```



Else

foundIndex = InStr(1, strHeader, "Date: ", vbTextCompare)

If foundIndex Then

foundIndex = foundIndex + Len("Date: ") + 5

mystrTime = Mid\$(strHeader, foundIndex)

mystrTime = Left\$(mystrTime, InStr(mystrTime, " GMT"))

Else

End If

End If

If IsDate(mystrTime) Then

FormatDateTime = CDate(mystrTime)

Else

FormatDateTime = Now - CDate("7:00")

End If

End Function

Public Function documentOK(ByVal strHeader As String) As Boolean

Dim msgCode As String

Dim startCode As Long

startCode = InStr(1, strHeader, "HTTP/", vbTextCompare)

If startCode Then

startCode = InStr(startCode, strHeader, " ")

msgCode = Mid\$(strHeader, startCode + 1)

```
msgCode = Left$(msgCode, InStr(msgCode, " "))
```

```
If msgCode = "200 " Then
```

```
    documentOK = True
```

```
Else
```

```
    documentOK = False
```

```
End If
```

```
Else
```

```
    documentOK = False 'assume error if no HTTP header
```

```
End If
```

```
End Function
```

```
Public Function IsURLExist(ByVal strURL As String, lastUpdate As Date) As Boolean
```

```
    Dim RS As New ADODB.Recordset
```

```
    With RS
```

```
        .ActiveConnection = "Provider=SQLOLEDB.1;User ID=sa;" &  
            "Initial Catalog=searchdb;" &  
            "Data Source=THESIS;"
```

```
        .CursorLocation = adUseServer
```

```
        .CursorType = adOpenKeyset
```

```
        .LockType = adLockPessimistic
```

```
        .Source = "SELECT URL,Date FROM URL WHERE URL='" & strURL & "'"
```

```
        .Open
```

```
        If .State = adStateOpen Then
```

```

If (.RecordCount = 1) Then

    IsURLExist = True

    lastUpdate = .Fields("Date")

Else

    IsURLExist = False

End If

Else

    MsgBox "Can't connect to SQL server"

    IsURLExist = False

End If

End With

RS.Close

Set RS = Nothing

End Function

Public Function addNewDBfnc(ByVal strURL As String, ByVal lastUpdate As Date, strWeb
As String, _
    strTitle As String, strMeta As String) As Boolean

Dim RS As New ADODB.Recordset

Dim mystr As String

On Error GoTo FoundConflict

With RS

.ActiveConnection = "Provider=SQLOLEDB.1;User ID=sa;" & _

```

"Initial Catalog=searchdb;" & \_

"Data Source=THESIS;"

.CursorLocation = adUseServer

.CursorType = adOpenKeyset

.LockType = adLockOptimistic

.Source = "SELECT \* FROM URL"

.Open

If .State = adStateOpen Then

.AddNew

.Fields("URL") = strURL

.Fields("Date") = lastUpdate

.Fields("Body") = strWeb

.Fields("Title") = strTitle

.Fields("Meta") = strMeta

.Update

addNewDBfnc = True

Else

addNewDBfnc = False

End If

.Close

End With

Set RS = Nothing

Exit Function

FoundConflict:

addNewDBfnc = False

End Function

Public Sub UpdateDB(ByVal strURL As String, ByVal lastUpdate As Date, \_

strBody As String, strTitle As String, strMeta As String)

Dim RS As New ADODB.Recordset

On Error GoTo FoundConflict

With RS

.ActiveConnection = "Provider=SQLOLEDB.1;User ID=sa;" & \_

"Initial Catalog=searchdb;" & \_

"Data Source=THESIS;"

.CursorLocation = adUseClient

.CursorType = adOpenStatic

.LockType = adLockBatchOptimistic

.Source = "SELECT \* FROM URL WHERE URL='" & strURL & "'"

.Open

If .State = adStateOpen Then

If (RS.RecordCount = 1) Then

.Fields("URL") = strURL

.Fields("Date") = lastUpdate

.Fields("Body") = strBody

.Fields("Title") = strTitle

```

.Fields("Meta") = strMeta

.UpdateBatch

End If

End If

.Close

End With

Set RS = Nothing

Exit Sub

FoundConflict:

MsgBox Err.Number & ": " & Err.Description

End Sub

Public Function CmmddyDate(ByVal dateStr As String) As String

Dim Wslash As Long, spaceSpot As Long

Dim yearStr As String, year As Integer

yearStr = dateStr

Wslash = InStr(4, dateStr, "/")

If Wslash Then

    yearStr = Mid(yearStr, Wslash + 1)

    spaceSpot = InStr(yearStr, " ")

    If spaceSpot Then

        yearStr = Left(yearStr, spaceSpot - 1)

    End If


```

```
year = CInt(yearStr)

If year > 80 Then
    year = year + 1900
Else
    year = year + 2000
End If

dateStr = Left(dateStr, Wslash) & year

CmmdyyDate = dateStr

Else
    CmmdyyDate = ""
End If

End Function

Public Sub DelOldData(ByVal oldDate As String)
    Dim RS As New ADODB.Recordset
    Dim ConnectStr As String

    ConnectStr = "Provider=SQLOLEDB.1;" & _
        "Data Source=THESIS;" & _
        "Initial Catalog=searchdb;" & _
        "UID=sa;"

    With RS
        .ActiveConnection = ConnectStr
```



```

.CursorLocation = adUseClient

.CursorType = adOpenStatic

.LockType = adLockBatchOptimistic

.Source = "SELECT * From Related " & _

        "WHERE (Date < CONVERT(DATETIME, "" & oldDate & "", 101))"

.Open

While Not .EOF

    .Delete

    .MoveNext

Wend

.UpdateBatch

.Close

End With

With RS

.ActiveConnection = ConnectStr

.CursorLocation = adUseClient

.CursorType = adOpenStatic

.LockType = adLockBatchOptimistic

.Source = "SELECT URL_Index From URL " & _

        "WHERE (Date < CONVERT(DATETIME, "" & oldDate & "", 101))"

.Open

While Not .EOF

    .Delete

```

.MoveNext

Wend

.UpdateBatch

.Close

End With

With RS

.ActiveConnection = ConnectStr

.CursorLocation = adUseClient

.CursorType = adOpenStatic

.LockType = adLockBatchOptimistic

.Source = "SELECT \* From Keyword " & \_

"WHERE (NOT (Key\_Index IN (SELECT Key\_Index " & \_

"From Related GROUP BY Key\_Index)))"

.Open

While Not .EOF

.Delete

.MoveNext

Wend

.UpdateBatch

.Close

End With

Set RS = Nothing

End Sub

## APPENDIX F

### SEARCH ENGINE MECHANISM'S SOURCE CODE

#### 1. Default.htm

This is the form for inputting keyword.

```
<HTML>

<HEAD>

<TITLE>Search Engine </TITLE>

</HEAD>

<BODY BGCOLOR="#FFFFFF">

<FORM METHOD=POST ACTION="result.asp">

Search for <INPUT TYPE="text" NAME="keyword"><BR>

<INPUT TYPE="submit" VALUE="Search"> <INPUT TYPE="reset">

</FORM>

</BODY>

</HTML>
```

#### 2. Displaypage.asp

This is the form for displaying the web page result.

```
<%
```

```

Response.Write "display page " & Request.QueryString("page") '& " from tmp" &
Request.QueryString("tmpfile") & ".txt"

Dim FileObj, InStream, tmpFile, allLink

Dim pageNum, pageDetail

Dim beginspot, endspot, pagelinkspot

pageNum = Request.QueryString("page")

Set FileObj = Server.CreateObject("Scripting.FileSystemObject")

tmpFile = Server.MapPath("/search/temp/tmp" & Request.QueryString("tmpfile") &
".txt")

Set InStream = FileObj.OpenTextFile(tmpFile, 1, False)

allLink = InStream.ReadAll

Set InStream = Nothing

Response.Write "<table>"

beginspot = InStr(allLink, "page_" & pageNum)

endspot = InStr(allLink, "page_" & pageNum + 1)

pagelinkspot = InStr(allLink, "<!--Page_link-->")

If beginspot Then

    beginspot = InStr(beginspot, allLink, "<TR>")

    If beginspot Then

        If endspot Then

            pageDetail = Mid(allLink, beginspot, endspot - beginspot)

        Else

            pageDetail = Mid(allLink, beginspot, pagelinkspot - beginspot)

```

End If

End If

Else

Response.Write "don't have page " & pageNum & " detail"

End If

Response.Write pageDetail

Response.Write "</table>"

If InStr(allLink, "page\_2") Then

Response.Write Mid(allLink, pagelinkspot)

End If

%>

### 3. Result.asp

This is the form for searching the user's keyword

```
<% UserKey = Request.Form("keyword")
```

```
MaxUser = "99"
```

```
i = 0
```

```
page = 0
```

```
If Trim(UserKey) = "" Then
```

```
Response.Redirect "InvalidKeyword.html"
```

```
Response.End
```

Else

```
Set DBObj = Server.CreateObject("ADODB.Connection")

DBObj.Open "Provider=SQLOLEDB.1;User ID=sa;Initial Catalog=searchdb;Data
Source=THESIS;"

Set FileObj = Server.CreateObject("Scripting.FileSystemObject")

NumFile = Server.MapPath("/search/number.txt")

Set InStream = FileObj.OpenTextFile(NumFile, 1, False)

FileIndex = InStream.ReadLine

Set InStream = Nothing

If (FileIndex = MaxUser) Then FileIndex = 0

Set OutStream = FileObj.OpenTextFile(NumFile, 2, True)

OutStream.Write FileIndex + 1

Set OutStream = Nothing

OutFile = Server.MapPath("/search/temp/tmp" & FileIndex & ".txt")

Set OutStream = FileObj.CreateTextFile(OutFile, True)

SQLQuery = "SELECT URL.URL, URL.Rank, URL.URL_Index, URL.Body, URL.Title "
SQLQuery = SQLQuery & "FROM Keyword INNER JOIN "
SQLQuery = SQLQuery & "Related ON Keyword.Key_Index = Related.Key_Index INNER
JOIN "
SQLQuery = SQLQuery & "URL ON Related.URL_Index = URL.URL_Index "
SQLQuery = SQLQuery & "WHERE (Keyword.Keyword = " & UserKey & ") "
SQLQuery = SQLQuery & "Order By - URL.Rank "

Set RSkeyword = DBObj.Execute(SQLQuery)
```

```

Do While Not RSkeyword.EOF

    i = i + 1

    If ((i Mod 10) = 1) Then

        page = page + 1

        OutStream.Writeline "page_" & page

        LinkEachPage = LinkEachPage & "<A HREF="""

        LinkEachPage = LinkEachPage & "displaypage.asp?tmpfile=" & FileIndex &

        "&page=" & page & """">"

        LinkEachPage = LinkEachPage & "<B>" & page & "</B></A> "

    End If

    OutStream.Writeline "<TR>"

    OutStream.Writeline " <TD>" & i & "</TD>"

    OutStream.Writeline " <TD>"

    OutStream.Writeline " <a href="""selectURL.asp?URL=" &

    Server.URLEncode(RSkeyword.Fields("URL")) & "&keyword=" &

    Server.URLEncode(UserKey) & """">"

    OutStream.Writeline " " & Server.HTMLencode(RSkeyword.Fields("Title")) &

    "</a>"

    OutStream.Writeline " </TD>"

    OutStream.Writeline "</TR>"

    OutStream.Writeline "<TR>"

    OutStream.Writeline " <TD></TD><TD>"

    OutStream.Writeline Server.HTMLencode(Left(RSkeyword.Fields("Body"), 200))

```



```

OutputStream.WriteLine "</TD></TR>"

OutputStream.WriteLine "<TR>"

OutputStream.WriteLine "  <TD></TD>"

OutputStream.WriteLine "  <TD>"

OutputStream.WriteLine "    <a href=\""selectURL.asp?URL=" &
Server.URLEncode(RSkeyword.Fields("URL")) & "&keyword=" &
Server.URLEncode(UserKey) & "\">"

OutputStream.WriteLine "      " & RSkeyword.Fields("URL") & "</a>"

OutputStream.WriteLine "    </TD>"

OutputStream.WriteLine "</TR>"

RSkeyword.MoveNext

Loop

RSkeyword.Close

Set RSkeyword = Nothing

OutputStream.WriteLine "<!--End of Query from keyword table-->"

If (InStr(UserKey, "+") Or InStr(UserKey, "-")) Then
    remainKey = LTrim(UserKey)

    Do

        plus = InStr(remainKey, "+")

        minus = InStr(remainKey, "-")

        spot = 0

        If (plus > 0) Then spot = plus

        If (minus > 0) Then

```

```

If (spot > 0) Then
    If (minus < spot) Then spot = minus
Else
    spot = minus
End If
End If
If spot Then
    curWord = Trim(Left(remainKey, spot - 1))
    Ch = Mid(remainKey, spot, 1)
    remainKey = LTrim(Mid(remainKey, spot + 1))
    If (QBody <> "") And (curWord <> "") Then
        QMeta = QMeta & Condition
        QTitle = QTitle & Condition
        QBody = QBody & Condition
    End If
    If (Ch = "+") Then
        Condition = " AND "
    Else
        Condition = " OR "
    End If
    If curWord <> "" Then
        QBody = QBody & "*****" & curWord & "*****"
    End If

```

Else

If remainKey <> "" Then

curWord = Trim(remainKey)

remainKey = ""

If (QBody <> "") And (curWord <> "") Then

QMeta = QMeta & Condition

QTitle = QTitle & Condition

QBody = QBody & Condition

End If

QBody = QBody & "" & curWord & ""

End If

End If

Loop While spot

mystr = QBody

Else

UserKey = Trim(UserKey)

mystr = "" & UserKey & ""

End If

SQLQuery = "select URL.\* "

SQLQuery = SQLQuery & " from URL "

SQLQuery = SQLQuery & " INNER JOIN "

SQLQuery = SQLQuery & " containstable (URL, \*, ' " & mystr & " ') as K "

SQLQuery = SQLQuery & " ON URL.URL\_Index = K.[KEY] "

```
SQLQuery = SQLQuery & "WHERE URL.URL_Index NOT IN (SELECT URL.URL_Index  
"
```

```
SQLQuery = SQLQuery & "FROM Keyword INNER JOIN "
```

```
SQLQuery = SQLQuery & "  Related ON Keyword.Key_Index = Related.Key_Index  
INNER JOIN "
```

```
SQLQuery = SQLQuery & "  URL ON Related.URL_Index = URL.URL_Index "
```

```
SQLQuery = SQLQuery & "WHERE (Keyword.Keyword = "" & UserKey & "") "
```

```
SQLQuery = SQLQuery & " order by URL.Rank DESC,K.Rank DESC"
```

```
Set RSurl = DBObj.Execute(SQLQuery)
```

```
Do While Not RSurl.EOF
```

```
  i = i + 1
```

```
  If ((i Mod 10) = 1) Then
```

```
    page = page + 1
```

```
    OutStream.WriteLine "page_" & page
```

```
    LinkEachPage = LinkEachPage & "<A HREF=""
```

```
    LinkEachPage = LinkEachPage & "displaypage.asp?tmpfile=" & FileIndex &  
"&page=" & page & "">"
```

```
    LinkEachPage = LinkEachPage & "<B>" & page & "</B></A> "
```

```
  End If
```

```
  OutStream.WriteLine "<TR>"
```

```
  OutStream.WriteLine "  <TD>" & i & "</TD>"
```

```
  OutStream.WriteLine "  <TD>"
```

```

    OutStream.WriteLine "      <a href=\""selectURL.asp?URL=" &
Server.URLEncode(RSurl.Fields("URL")) & "&keyword=" & Server.URLEncode(UserKey)
& "\">"

    OutStream.WriteLine "      " & Server.HTMLencode(RSurl.Fields("Title")) & "</a>"
    OutStream.WriteLine "    </TD>"

    OutStream.WriteLine "</TR>"

    OutStream.WriteLine "<TR>"

    OutStream.WriteLine "    <TD></TD><TD>"
    OutStream.WriteLine Server.HTMLencode(Left(RSurl.Fields("Body"), 200))
    OutStream.WriteLine "</TD></TR>"

    OutStream.WriteLine "<TR>"

    OutStream.WriteLine "    <TD></TD>"
    OutStream.WriteLine "    <TD>"
    OutStream.WriteLine "      <a href=\""selectURL.asp?URL=" &
Server.URLEncode(RSurl.Fields("URL")) & "&keyword=" & Server.URLEncode(UserKey)
& "\">"

    OutStream.WriteLine "      " & RSurl.Fields("URL") & "</a>"

    OutStream.WriteLine "    </TD>"

    OutStream.WriteLine "</TR>"

    RSurl.MoveNext

Loop

OutStream.WriteLine

OutStream.WriteLine "<!--Page_link-->"

```

```

OutStream.WriteLine LinkEachPage

Set OutStream = Nothing

    Response.Redirect "displaypage.asp?tmpfile=" & FileIndex & "&page=1"

End If

%>

```

#### 4. SelectURL.asp

This is the form for updating, and creating in index, related, and URL table.

```

<%
If Request.QueryString("URL") <> "" Then
    Dim RS ' As New ADODB.Recordset
    Dim ConnectionStr, SQLQuery, increment
Dim URLIdx
Dim URLdate

    Set RS = Server.CreateObject("ADODB.Recordset")
    ConnectionStr = "Provider=SQLOLEDB.1;Data Source=THESIS;"
    ConnectionStr = ConnectionStr & "Initial Catalog=searchdb;User ID=sa;"

adUseServer = 2
adOpenDynamic = 2
adLockOptimistic = 3
adStateOpen = 1

```

```

SQLQuery = "SELECT * FROM URL WHERE URL='" &
Request.QueryString("URL") & "'"

'Response.Write "<br>" & SQLQuery & "<br>"

RS.CursorLocation = adUseServer

RS.Open SQLQuery, ConnectionStr, adOpenDynamic, adLockOptimistic

If Not RS.EOF Then

    URLIdx = RS.Fields("URL_Index")

    URLdate = RS.Fields("Date")

    RS.Fields("Rank").Value = RS.Fields("Rank").Value + 1

    RS.UpdateBatch

Else

    End If

RS.Close

Dim keywordIdx

SQLQuery = "SELECT * FROM Keyword WHERE Keyword='" &
Request.QueryString("keyword") & "'"

RS.Open SQLQuery, ConnectionStr, adOpenDynamic, adLockOptimistic

If RS.EOF Then

    RS.AddNew

    RS.Fields("Keyword") = Request.QueryString("keyword")

    RS.Update

    RS.Close

    RS.Open SQLQuery, ConnectionStr, adOpenDynamic, adLockOptimistic

```

```

If Not RS.EOF Then

    keywordIdx = RS.Fields("Key_Index")

    'Response.Write "keyword table updated : key_index = " & keywordIdx

End If

Else

    keywordIdx = RS.Fields("Key_Index")

End If

RS.Close

If URLIdx <> 0 And keywordIdx <> 0 Then

    SQLQuery = "SELECT * From Related Where (key_index = " & keywordIdx & ") And
(URL_Index = " & URLIdx & ")"

    RS.Open SQLQuery, ConnectionStr, adOpenDynamic, adLockOptimistic

    If RS.EOF Then

        RS.AddNew

        RS.Fields("Key_Index") = keywordIdx

        RS.Fields("URL_Index") = URLIdx

        RS.Fields("Date") = URLdate

        RS.Update

    End If

Else

End If

Response.Redirect ("http://" & Request.QueryString("URL"))

Response.End

```



Else

End If

%>



