A Customized Web Search Engine Using a Tiny WebSQL Query Language

A Project Report

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirement for

the Degree of

Master of Science and Computer Engineering

By

Attasith Patamarjarnkul

Auburn, Alabama
May, 20000

Acknowledgement

The author would like to express his sincere gratitude to his major professor Dr. Wen Chen Hu for his advice, understanding, and encouragement throughout the long course of the project implementation. He also would like to thank his committee members, Dr. Kai H. Chang and Dr. Gerry Dozier, for their valuable time and for all the knowledge they have given him during the course of his graduate studies. The special help in the report grammar editing that has been done by Jeff Cunningham is also appreciated by the author.

Abstract

The World Wide Web (the Web) contains a huge collection of documents, which are connected by hyperlinks. The most common technology currently used for searching the Web depends on sending information-retrieval requests (search inputs) to "search engines", also known as "index servers", that index as many documents as they can find from the Web [5]. Each search engine uses software called "spider" or "robot" to collect information from the Web in order to create Web site entries. Those entries are then stored into the database of the search engine. When receiving search queries from users, search engines search based on Web site entries. This provides a search service at only the general level, where most spiders do not exploit the structure and topology of the document network. Based on this technique, users may find a lot of Web sites that are not related to the search input. Furthermore, in many search engines, users do not have any means to restrict the range of Web sites for the spiders to collect Web information.

This project proposes: (i) the Tiny WebSQL search engine; and (ii) the Tiny WebSQL query as the restrictions for a spider to collect Web information. The Tiny WebSQL search engine uses a spider to exploit the topology of the Web document network. The syntax of the Tiny WebSQL Query language is similar to SQL language syntax. This approach benefits system users since SQL is a well-known and widely-used language. By using the Tiny WebSQL query to instruct the spider, users can specify the scope of the Web sites to collect information.

Contents

1. Introduction

Nowadays, there are a number of services provided by search engines. One of the services from those search engines is an extra search service called advanced search. The advanced search service can search by using more than one keyword with Boolean operators, or will allow a user to specify the type of Web document to be included in the search results. It may also allow the user to assign weight to each keyword for the search engine to rank search results before returning them to the user. However, there is no common interface or method for the advanced search service. The advanced search services on different search engines provide different features and different interfaces. Users have to be aware of the different features and interfaces when using advanced search services on different search engines. Furthermore, most of the search engines do not exploit the structure and topology of the Web document network. Therefore, the user cannot specify the range of the Web sites that the Web information should be collected.

This project proposes: (i) the Tiny WebSQL search engine; and (ii) the Tiny WebSQL query as the restrictions for a spider to collect Web information. The Tiny WebSQL search engine uses a spider to exploit the topology of the Web document network. The syntax of the Tiny WebSQL Query language is similar to SQL language syntax. This approach benefits system users since SQL is a well-known and widely-used language. By using the Tiny WebSQL query to instruct the spider, users can specify the scope of the Web sites to collect information.

The objectives and the project outline are given in this chapter. Chapter 2 explains how search engines work. The analysis of some popular search engines is in chapter 3. The Tiny WebSQL Query language and the Tiny WebSQL search engine are introduced in chapter 4. Chapter 5 shows the methods used in this project. The Web interfaces and some experimental results are given in Chapter 6. Chapter 7 concludes this project and gives some future research directions.

1.1 Objectives

The following list gives the objectives of this project:

- Study how search engines work.
- Examine simple and advanced services from some popular search engines.
- Define an advanced search method by using an SQL-like language, the Tiny WebSQL Query language.
- Apply compilation techniques to the Tiny WebSQL Query language.
- Experience the Web programming based on Common Gateway Interface (CGI).
- Integrate the database system to the Web.
- Practice database programming in Java with database connectivity called JDBC.

1.2 Project Outline

This project includes four major research areas: the Tiny WebSQL Query language, a customized Web spider, Web-database connectivity, and relevancy ranking.

- **The Tiny WebSQL Query Language**

This language is used to set the restrictions for the spider to collect information from the Web. For example, a query can instruct the spider to collect information from Web sites whose title includes the word "holiday".

- **A Customized Web Spider (The Tiny Spider)**

The spider navigates the Web via the hyperlinks within the Web pages. For each Web site being crawled, the spider analyzes the Web document and checks whether it satisfies the restrictions specified in a Tiny WebSQL query.

- **Web-Database Connectivity**

The Web-database connectivity is to pass three types of queries from the Web to the proposed system. The first query type is in the Tiny WebSQL Query language format. It instructs the spider how and where to collect the Web information. The second query type is the search query, which instructs the search software the keywords to search. The last query type is purposed to query the database of the search engine to check the information downloaded from the Web. Figure 1.1 shows the Web-database connectivity used in this project.

- **Relevancy Ranking**

The search results are ranked by using a method called relevancy ranking. The idea is to rank the search results according to the similarity between the search results and the query keywords. The details of the relevancy ranking technique used in this project are given in chapter 5.

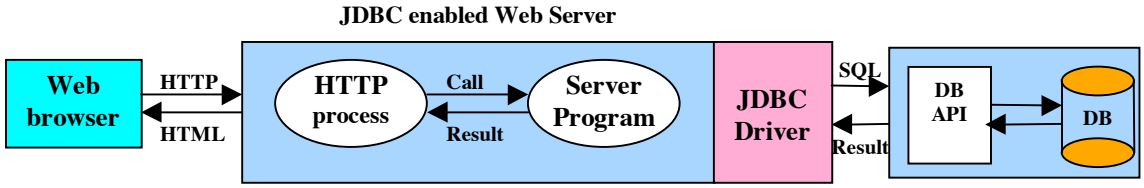


Figure 1.1: *Web-Database Connectivity used in the project*

2. Search Engines

Searching for Web sites is one of the common tasks performed on the Web. Nowadays, about 71% of Internet users utilize search services to find Web sites, according to Nielsen Media research, a company that measures computer and Internet usage [3]. There are two major search services used in the Internet: Search engines and Web directories. Traditional search technology is based on users typing in keywords for the information they want to receive, then a search service scans Web pages for those keywords. As the project is about search engines, the details of Web directories are briefly covered in this report. Details of search engines are given in this chapter and the analysis of some popular search engines will be in chapter 3.

2.1 Search Engine and Web Directory

The Web search services, which include search engines and Web directories, have very different implementation methods and search strategies from each other. Basic concepts for search engines are given in section 2.1.1, and basic concepts for Web directories are given in section 2.1.2.

2.1.1 Search Engine Basics

Search engines consist of three main components: the spider, the index, and the search software [3]. The first component, the spider, also known as crawler or robot, is software that automatically scans various Web sites and creates indices of URLs,

keywords, links, text, etc. for those Web sites. The spider also follows the links on the site to find other Web sites.

When a user submits a search query with keywords, the search software goes through the index in the search engine's database to find Web sites with the desired keywords, and ranks the results in term of relevance of the results and the keywords.

2.1.2 Web Directory Basics

Web directories, such as Yahoo, work with descriptions of Web pages submitted by either Web masters or Web editors who have reviewed the Web pages. Web directories respond to a search query by searching through these descriptions. Some search engines, such as Microsoft's MSN and Netscape Search, take a hybrid approach by also using indices and directories [3].

Since Web directories do not use spiders to collect Web information, they cannot automatically find changes in Web pages. The reason Web directories do not use spiders is that they believe that using the human-generated descriptions can produce more relevant responses to some search queries [4].

2.2 How Do Search Engines Work?

As mentioned in section 2.1.1, search engines use spiders to survey the Web and build their databases. A spider is software that automatically follows hyperlinks from one Web site to the next. When a spider navigates a new Web site, the Web information is retrieved and indexed to be stored in the search engine's database. When a user enters a

search query, it will be checked against Web sites in the database. The best matches are then returned to the user as hits.

There are a number of features in search engines to consider, and the following are important features that will be mentioned in this chapter:

- Search strategies: A search engine may perform keyword-based search (also known as keyword search) and/or concept-based search.
- Advanced search: A user has to provide some search restrictions to refine search results.
- Relevancy ranking: A search engine provides search results ranked by the technique called relevancy ranking, which is varied from one search engine to another.

2.3 Search Strategies

After the user enters the keywords as the search query, the search engine will perform the search. There are two primary methods for searching:

- Keyword search, and
- Concept-based search

2.3.1 Keyword Search

Keyword search is the most common form of text search on the Web. Most search engines process their queries and retrievals using keywords. In fact, the search engine determines a Web document from keywords specified in the Meta tag in the header part of the HTML file of that Web document [4]. However, it is up to the search engine to

determine if a Web document does not specify its keywords. (Essentially, this means that search engines pull out and index words that are believed to be significant.)

The Problems with Keyword Search

1. It cannot distinguish between words that are spelled the same way, but have different meanings; e.g. **Jordan** map, and Michael **Jordan**, Computer **Network** and Neural **Network**. This often results in hits that are completely irrelevant to an input query.
2. Some search engines also have trouble with the stemming, which is the grammatical variation of search terms such as "eat" and "eating".
3. It cannot return hits on keywords that have the same concept, but are spelled differently. A query on the word "friend" will not return the document about "partner", "team", "pal", or "dude" instead of the exact word "friend" itself.

2.3.2 Concept-Based Search

Unlike keyword search systems, concept-based search systems try to determine what a user means, not just what is entered as the input query. In the best circumstances, a concept-based search returns hits on Web documents that are about the subject the user is exploring, even if the words in the Web documents do not precisely match the input query.

There are various methods of building concept-based search engines, some of which are highly complex, relying on sophisticated linguistic and artificial intelligence theory (that this project does not cover). One example of a concept-based search engine is in the Excite search engine. It is a general-purpose search engine that also performs

concept-based searches. Excite sticks to a numerical approach. Its software determines meaning by calculating the frequency with which certain important words appear in a Web document [4]. 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 [4].

However, this often works better in theory than in practice. The concept-based search is a good idea, but it is far from perfect.

2.4 Search Refinement

Most search engines offer two different types of search: "basic search" and "advanced search". Sometimes, advanced search is referred to as search refinement. In a basic search, a user just enters keywords without sifting through any pull-down menus of additional options, then the search engine performs the search using the user-entered keywords.

Advanced search options differ from one search engine to another, but some of the possibilities include the ability to search on more than one keyword, to give more weight to one keyword than other keywords, and to exclude some words. A user can choose to search on proper names, on phrases, or on words that are found within a certain proximity to other search terms.

Some search engines allow users to specify the form in which the results will appear and whether the user wishes to restrict the search to certain domains on the Internet or to specific parts of Web documents; e.g. the title, URL, etc.

Many search engines allow a user to use the Boolean operators to refine search results. These include logical terms AND, OR, NOT, and proximal locators NEAR and FOLLOWED BY.

Boolean operator **AND** means that all the search terms must appear in the documents; e.g. "protocol" AND "analyzer." This Boolean operator might be used if a user wants to exclude common hits that would be irrelevant to the query.

Boolean operator **OR** means that at least one of the search terms must appear in the documents; e.g. Kane OR Mick OR Vince. This Boolean operator might be used if a user does not want to be too strict in a search.

Boolean operator **NOT** means that the search terms must not appear in the documents.

Some search engines use the characters + and - instead of Boolean operators to include and exclude search terms.

Locator **NEAR** means that the terms should be within a certain number of words of each other.

Locator **FOLLOWED BY** means that one term must directly follow the other.

Locator **ADJ**, for adjacent, serves the same function as FOLLOWED BY.

Phrases: The ability to query on phrases is very important in a search engine. Those that allow this capability usually require a user to enclose the phrase in quotation marks; e.g. "walking on the moon" .

Capitalization: This is essential for searching on proper names of people, companies, or products.

2.5 Relevancy Ranking

Most search engines return search results with relevancy rankings. In other words, they list the hits according to the similarity of the results and the keywords. However, the search results usually seem irrelevant to information that the user is looking for since search engine technologies have not reached the point where humans and computers understand each other clearly.

Most search engines use the frequency of keyword appearance as a primary way to determine the relevancy score of search results. If one is researching wrestling and the word "wrestling" appears multiple times in a Web document, it is reasonable for a search engine to assume that the document will contain useful information. Therefore, a document that repeats the word "wrestling" over and over is likely to turn up near the top of the list of the search results.

If a keyword is a common one, or if it has multiple meanings, a user could receive a lot of irrelevant hits as the search results. And if the keyword is a subject about user-desired information, the user does not need to see it repeatedly because it is the information *about* that word that the user is interested in, not the word itself.

Some search engines consider both the frequency and the position of keywords to determine relevancy, reasoning that if the keywords appear early in the document, this increases the likelihood that the document is on target. For instance, Lycos ranks results according to the frequency of keywords appearing in their indices of the Web document and which fields they appear; e.g. title, description [4].

There are three major methods of relevancy ranking:

- Ranking by frequency and/or positions that the keywords appear in a Web document.
- Ranking by how many Web sites link to that particular Web document.
- Ranking by assigning weight to each keyword term.

This project does not work in depth in the relevancy ranking. The methods used to retrieve hits in the project are exact match and ordered character matching, which will be explained in chapter 5.

2.6 Information from Meta Tags

Some search engines are now indexing Web documents according to Meta tags in an HTML document. Meta tags are defined within the beginning part of a Web document, which are known as "head" tags. This means that a Web page author can have some influence over which keywords should be used to index a Web document, and even in the description of the Web document.

Several major search engines collect information from “title”, “keywords”, and “description” Meta tags from Web pages to use as Web information to perform searches. Using relevant keywords in the title and varying titles on different pages that make up a Web site can target as many keywords as possible. As for the "description" Meta tag, some search engines use it as a short summary of the Web site’s URL for search results.

The following is an example of the use of Meta tags within the “head” tag.

```
<head>
<title>This is title</title>
<meta name="keywords" content="keyword1, keyword2, keyword3, keyword4">
<meta name="description" content="This is the description of the Web document">
</head>
```

The above HTML code contains information about the TITLE, KEYWORDS, and DESCRIPTION of a Web document. Many search engines collect information about the Web document from these parts. The Tiny WebSQL search engine, the search engine in the project, also mainly refers to and stores information from these parts of the Web document instead of storing the entire Web document into its database.

3. Analysis of Some Popular Search Engines

Search policies vary from search engine to search engine. This chapter analyzes search policies, which include search type, search option, search domain, search refinement, methods of ranking, advantages, and disadvantages.

3.1 Alta Vista

Alta Vista is a powerful keyword-based search engine that provides two search options: Simple search and advanced search, with a language option. The simple search uses keywords and selected search domain as the search query. The search domain also performs searches based on keyword matching in a specific domain (e.g. "Web", or "News"). The user interface of the simple search with search domain refinement is shown in Figure 3.1.

While Alta Vista's main search is used for general searching, advanced search is used to conduct a very specific search [2]. Advanced search allows users to tailor the search to locate what they are looking for by using Boolean operators to create relationships among keywords in the search query. The user interface for the advanced search of the Alta Vista search engine is shown in Figure 3.2.

To do a simple search, the "Search" tab must be selected and the keyword must be specified in the "Text Entry Box". The search domain can be reached by selecting the search domain from "Find result on". The language option is provided for some languages in the "Language drop-down menu".

To do an advanced search, the "Advanced Search" tab must be selected and the keywords with the Boolean operators must be entered into the "Text Entry Box". By default, the search results from the advanced search will not be sorted. The user has to specify the constant string into the "Sort by" box to define a string for the search engine to sort and rank the search results based upon the comparison to that string. The date entry field is used to restrict search results from only Web sites published in a specific period of time. The language option is also provided in the advanced search.

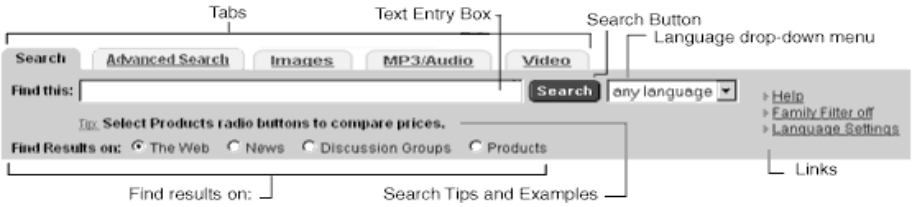


Figure 3.1: *User interface for the simple search of the Alta Vista search engine*

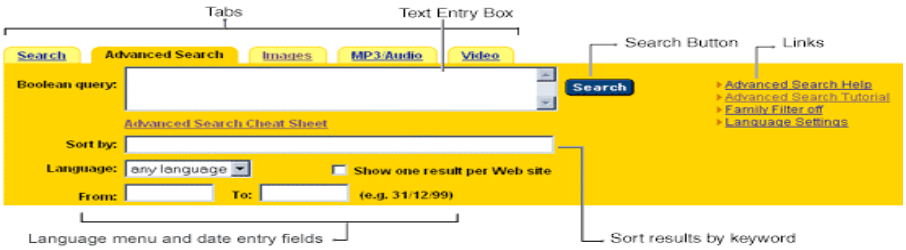


Figure 3.2: *User interface for the advanced search of the Alta Vista search engine*

Summary of Alta Vista Search Engine

Type of search: Keyword

Search options: Simple or Advanced search, search refinement

Domains searched: Web, Usenet

Search refinement: Boolean operators "**AND**", "**OR**", "**NOT**," and the proximal locator "**NEAR**." Allows wildcard and backward searching (i.e., users can find all the Web sites that link to a specific page). A user can decide how search terms should be weighed, and where in the document to look for them.

Relevancy ranking [1]:

In the main search, the ranking of results is determined by a few simple rules. Pages move up the list if they have:

- More instances of the search terms.
- Search terms that appear in relatively few other pages.
- Search terms closer together in the text.

Results to be presented as: First several lines of the document.

Advantages: Fast search, capitalization, and large databases. Alta Vista searches both the Web and the Usenet for words and phrases. A user can even search to discover how many people have linked their sites to his own Web sites. Users can also have the resulting pages of their searches translated into several other languages.

Disadvantages [4]: Multiple pages from the same site show up too frequently; some inconsistent relevancy rankings, especially on the simple search.

3.2 Excite

Excite is currently the best-known general-purposed search engine site that uses concept-based search. It provides search features similar to the advanced search in the Alta Vista search engine. However, methods used to specify the advanced search are slightly different from Alta Vista. Figure 3.3 shows the user interface of the advanced search for the Excite search engine. As a concept-based search engine, search results from Excite are not only the Web sites that contain the keywords, but they also include Web sites that are related to the keywords.

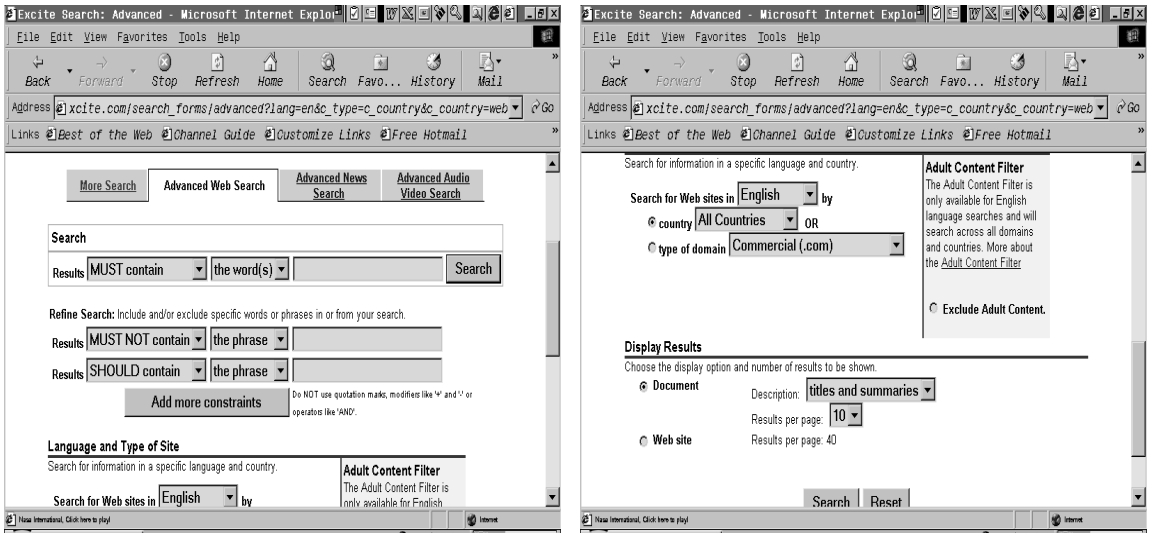


Figure 3.3: *User interface for the advanced search of the Excite search engine*

Summary of Excite Search Engine

Type of search: Both concept and keyword

Search options: Simple and Advanced search, search refinement

Domains searched: Web, Usenet

Search refinement: Boolean operators **AND**, **OR**.

Relevancy ranking: Not clear

Results to be presented as: Summary of each search result consisting of title (as hypertext), description, and URL.

Advantages: Large index and excellent summaries, which also shows the top few most important sentences in the document. Users can view the search results in various ways; e.g. grouped by confidence, grouped by Web site, etc.

Disadvantages: Does not specify the format or the size of the search results, nor does it tell users the exact frequency that the keywords appear in each particular Web site.

3.3 Infoseek

Infoseek uses a full-text indexing system. It picks up every word in the text except commonly occurring stop words such as "a", "an", "the", "is", "and", "or", and "www". It also provides the advanced search with more sub-categories of search domains to help users specify more consistent ranges of the search. The search domain can be Web, White pages, Usenet, Companies, News, Stocks, etc. Figure 3.4 shows the user interface for the advanced search of the Infoseek search engine.

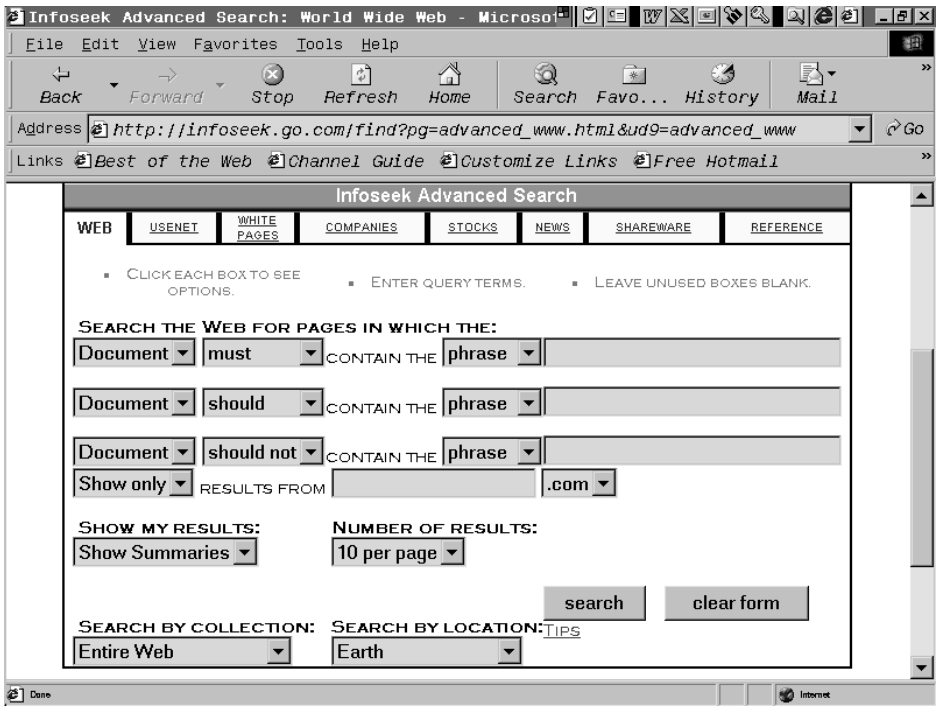


Figure 3.4: *User interface for the advanced search of the Infoseek search engine*

One benefit of the Infoseek search engine is that it categorizes information in Web documents. It allows users to specify the category within Web documents that they want to search such as: URL or title.

Summary of Infoseek Search Engine

Type of search: Keyword

Search options: Simple and Advanced search

Domains searched: Web, Usenet, Usenet FAQ's, Reviews, Topics

Search refinement [4]: Phrases, capitalization, no Boolean operators, but uses + and - instead.

Relevancy ranking [4]: Based on the number of times the keywords appear in a Web document.

Results presented as [4]: First 30-100 words of the page

Advantages: Fast, flexible, reliable searching. Good output, which gives the URL, the size of the document, and the relevancy score. Allows users to see similar pages (based on topic information about the pages). Full-text indexing, allows capital letters and phrases.

Disadvantages: Smaller index.

3.4 Lycos

Lycos is a keyword-based search engine. Advanced search in Lycos consists of four main options: CONTENT, PAGE FIELD, LANGUAGE, and LINK REFERALS (these four options are claimed as the search refinements in the Lycos search engine). The user interface of Lycos is shown in Figure 3.5 (a-d).

The "CONTENT" categorizes search domains as in other search engines, but it provides more groups of search domains for users to select. The "PAGE FIELD" performs the search by some fields in the Web documents, as in Figure 3.5b. The "LANGUAGE" option allows a user to select a language, shown in Figure 3.5c. The "LINK REFERALS" is the idea from Lycos that is similar to a Web directory to collect Web sites about a specific topic into the same group. When a user searches for information, Lycos will find results from Web sites within a particular topic specified in the LINK REFFERALS option. The user interface of the Link REFERALS is shown in Figure 3.5d

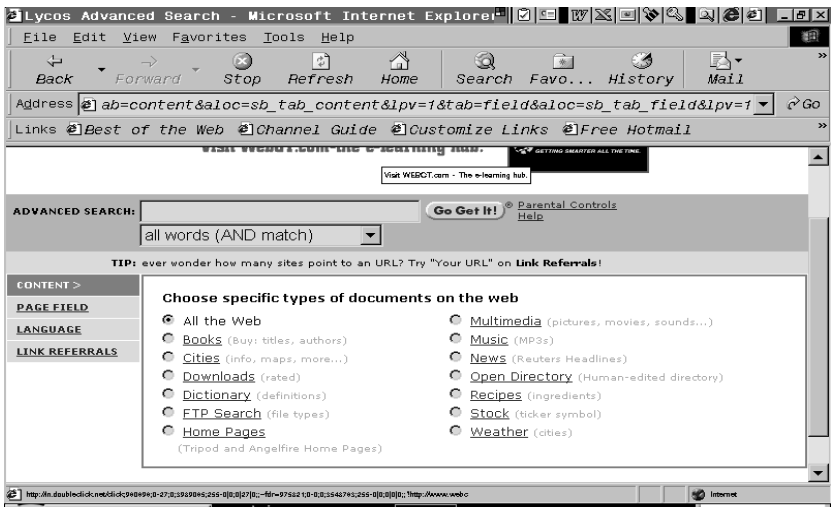


Figure 3.5a: *User interface for the Lycos search engine with the CONTENT option.*

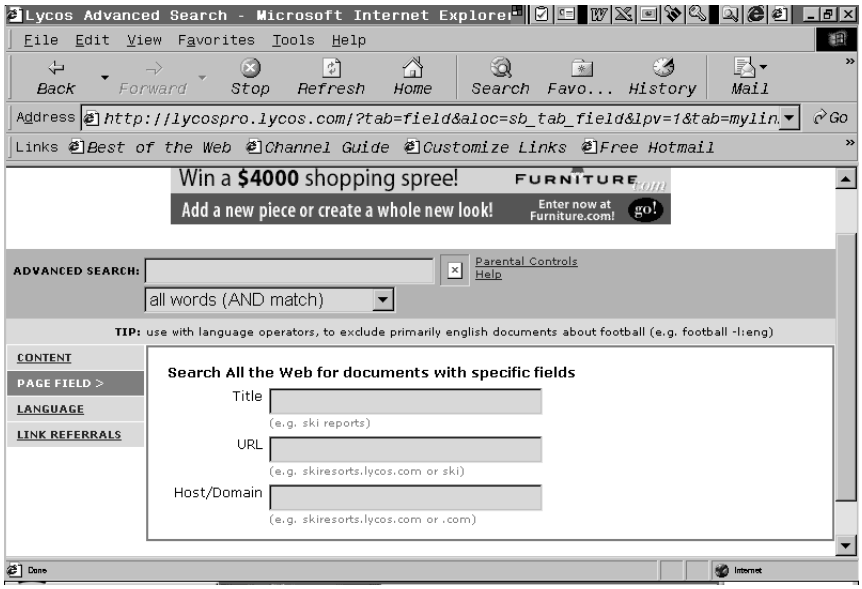


Figure 3.5b: *User interface for the Lycos search engine with* the *PAGE FIELD option*.

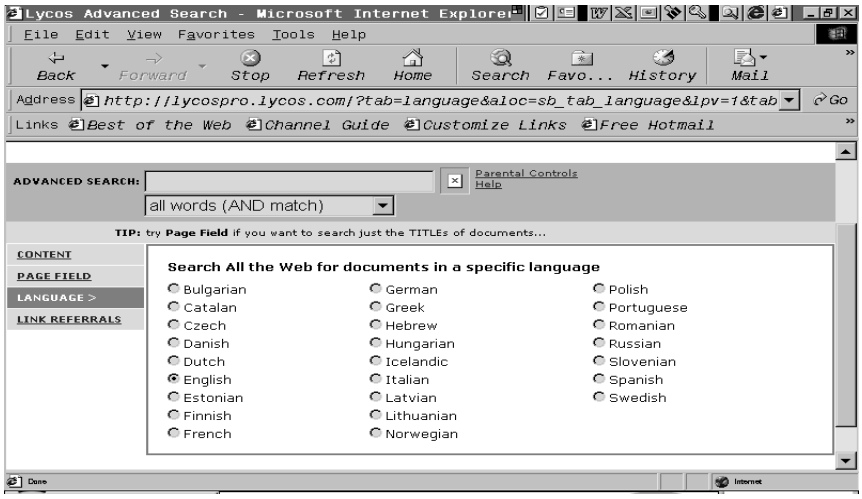


Figure 3.5c: *User interface for the Lycos search engine with the LANGUAGE option*.

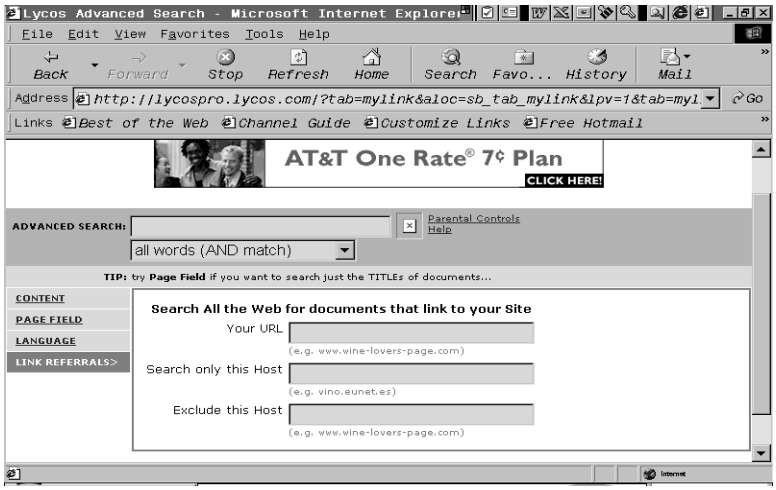


Figure 3.5d: *User interface for the Lycos search engine with the LINK REFERALS option.*

Summary of Lycos Search Engine

Type of search: Keyword, but Lycos is gradually becoming less of a search engine, and more of a Web directory.

Search options: Basic and Advanced search.

Domains searched: Web, Usenet, News, Stocks, Weather, Multi-media.

Search refinement: Full Boolean capabilities.

Relevancy ranking [4]: Frequency and position that the keywords appear in a Web document, as well as the frequency with which Web documents are linked to.

Results to be presented as [4]: First 100 words, in simple search. User defined with advanced search.

Advantages: Large database. Comprehensive results given; i.e. the date of the document, its size, etc. Ability to search images and audio.

3.5 HotBot

HotBot is a keyword-based search engine that also ignores stop words such as "a", "an", "the", "is", "and", "or", and "www". Search refinements in HotBot include language options, periods that Web documents were published, and Web document types. HotBot allows a user to define physical location of Web sites. A unique feature from HotBot is word stemming, the grammatical variation of search terms such as "eat" and "eating". Search results from HotBot can be chosen from URL, some part of the description, or full description. The user interface is divided into four parts as shown in Figure 3.6 (a - d).

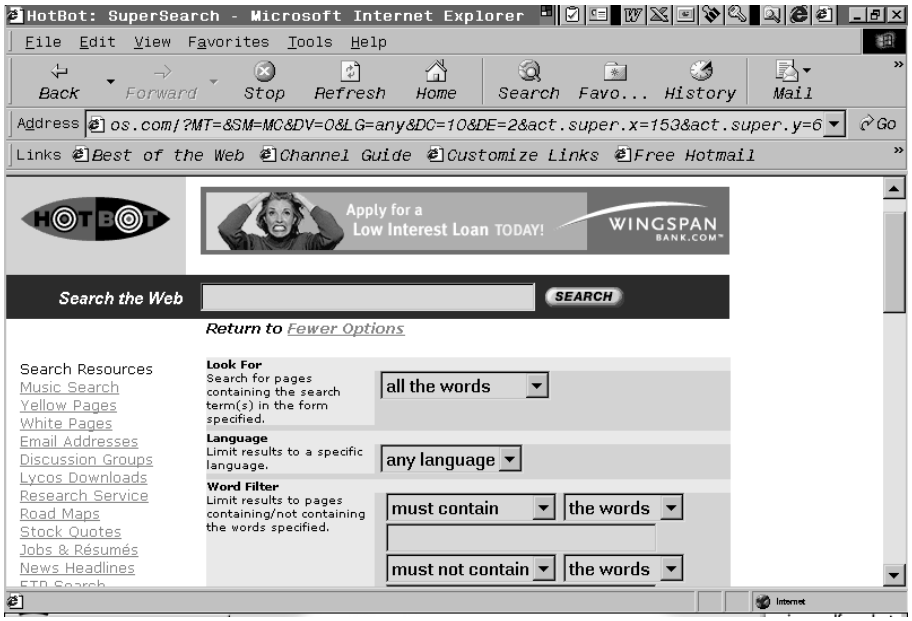


Figure 3.6a: *User interface for the advanced search of the HotBot search engine (part1).*

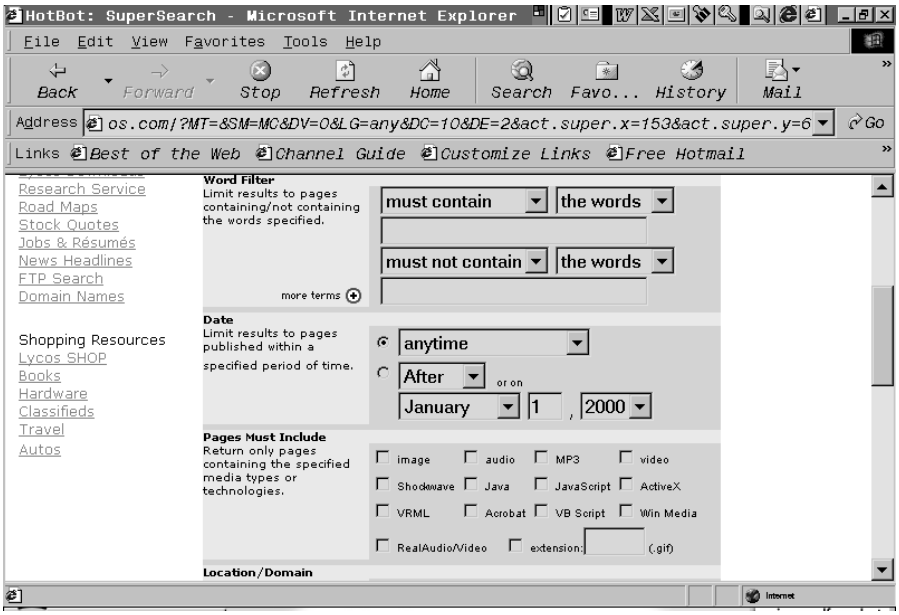


Figure 3.6b: *User interface for the advanced search of the HotBot search engine (part 2).*

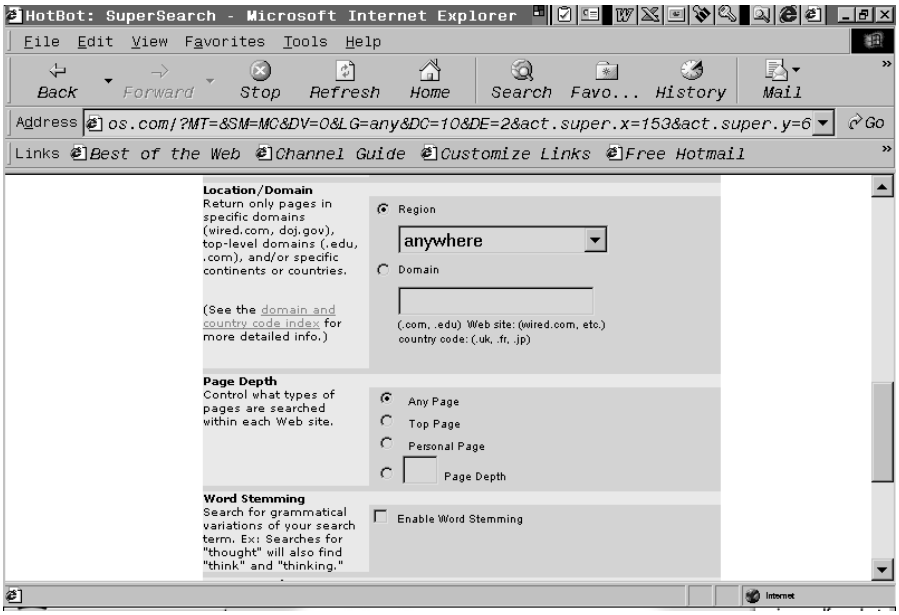


Figure 3.6c: *User interface for the advanced search of the HotBot search engine (part 3).*

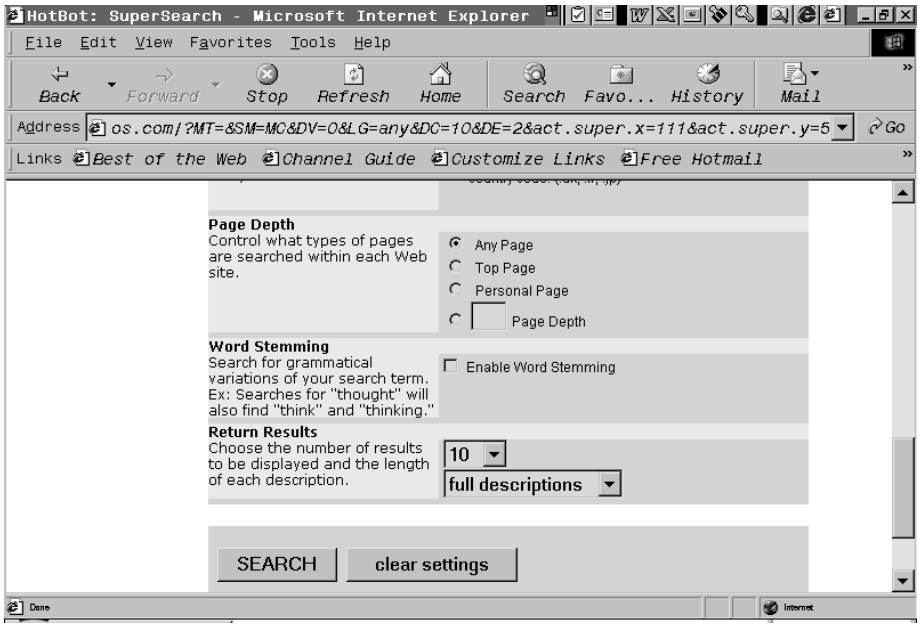


Figure 3.6d: *User interface for the advanced search of the HotBot search engine (part 4).*

Summary of HotBot Search Engine

Type of search: Keyword

Search options: Simple, Advanced (also known as Modified or Expert)

Domains searched: Web

Search refinement: By phrase, person, and Boolean-like choices in pull-down boxes. In Expert search mode users can search by date and even by different file or media types such as Java, JavaScript, or VRML.

Relevancy ranking: Keywords appearing in the title will be ranked higher than keywords appearing in another part of the document. Higher frequency of appearance of keywords also results in a higher rank.

Results to be presented as: Relevancy score and URL

Advantages: Fast because of the use of parallel processing, which distributes the load of queries as well as the database over several work stations.

Disadvantages [4]: Some limitations on Boolean operators, and the help files still are not very good.

3.6 Yahoo

Although Yahoo is not precisely a search engine, it is an important Web search service site. Yahoo works as a hierarchical subject index, which allows users to find information from general topics to specific topics. Yahoo is an attempt to organize and catalogue the Web with the concept of Web directory.

Summary of Yahoo Web Directory

Type of search: Keyword

Search options: Simple, Advanced

Domains searched: Yahoo's index, Usenet, E-mail addresses. Titles, URLs and the brief comments or descriptions of Web sites.

Search refinement: Boolean AND, OR. Yahoo is case insensitive.

Relevancy ranking: Not clear

Results to be presented as: A two-line description of the site, which the Web master has submitted when adding the Web site to Yahoo.

Advantages: Easy-to-navigate subject catalogue.

Disadvantages: Only small portion of the Web has actually been catalogued by Yahoo.

4. A Tiny WebSQL Search Engine

The Tiny WebSQL search engine is a customized search engine, which uses restrictions for its spider to collect information from the Web. The restrictions are specified in Tiny WebSQL Query language. The reasons to use Tiny WebSQL search engine are given in section 4.1. In section 4.2, the structure of the Tiny WebSQL search engine is explained. The ideas behind the Tiny WebSQL Query language, syntax, semantics, and some examples of the Tiny WebSQL queries are in section 4.3.

4.1 Why Tiny WebSQL Search Engine?

Search policies vary on different search engines. Most search engines are general-purpose search engines that do not exploit the topology in a document network. Some search engines allow users to customize the search restrictions. However, the users have to be aware of different methods and features on those different search engines. Furthermore, the users may not be able to specify the restrictions on the range of Web sites to collect information.

The Tiny WebSQL search engine uses the Tiny WebSQL Query language as the restriction for the spider to collect Web information. With the syntax similar to SQL language, the Tiny WebSQL is familiar to most database users. It also allows users to exploit the network topology, which is crucial for limiting the range of the Web sites to be searched.

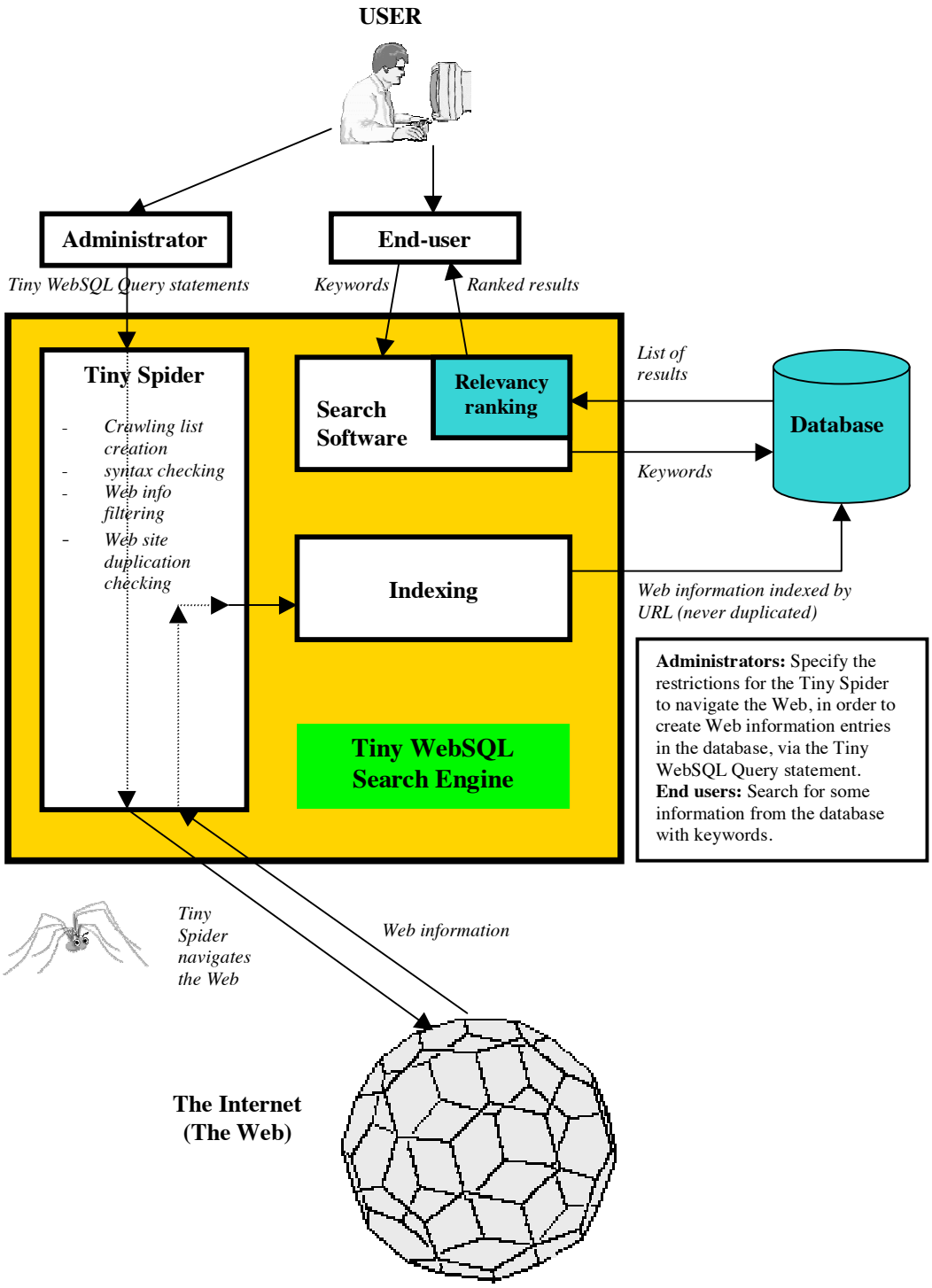
By using the Tiny WebSQL Query language, users can restrict both Web information collection, and the range of the Web sites that the information will be retrieved from. Furthermore, the Tiny WebSQL search engine is easy to use since the syntax is close to SQL language syntax.

4.2 System Structure

The Tiny WebSQL search engine consists of three major components: the spider, the indexing, and the search software. To create the database, a user has to provide the restrictions for the spider to collect Web information. The restrictions must be in the Tiny WebSQL Query language. While the spider is crawling the Web, only Web sites with information that satisfy the restrictions are returned to the search engine. The information from those Web sites are indexed and stored in the search engine's database. The database now contains only Web sites that satisfy the restrictions, and the customized search engine is ready to provide the search service.

The search software uses the keywords entered from a user to search for information in the previously created database. The search software then ranks the search results using a relevancy ranking method. The overall structure of the Tiny WebSQL search engine is shown in Figure 4.1.

Figure 4.1: *Overall structure of the Tiny WebSQL search engine*



4.3 The Tiny WebSQL Query Language

The Tiny WebSQL Query language is used as the restriction for the Tiny WebSQL search engine to collect Web information. Syntax of the language is based on the structure of SQL language syntax. However, the Tiny WebSQL Query language is designed to work with the Web. Therefore, Web features have been introduced to the language.

4.3.1 Model Document Structure (MDS)

There is difficulty in using the Tiny WebSQL Query language due to the lack of database schema on the Web. To use the Tiny WebSQL query as the restrictions to collect Web information, model document structures (MDSs) must be created. MDSs are created by a spider when it is visiting each Web site. They are seen as tables that consist of a number of fields. The spider retrieves necessary information from each Web site visited and puts this information into corresponding fields in each MDS. By creating the MDSs, the spider can compare information from each Web site to the restrictions before making the decision of whether to collect the Web site or not.

MDSs are discovered and seen by the spider as two tables: Table Document and table Anchor. Each Web site will be seen as an entry of table Document, which consists of 6 fields [URL, TITLE, TYPE, DESCRIPTION, KEYWORDS, TEXT], and as an entry of table Anchor, which consists of 3 fields [BASE, HREF, LABEL]. Again, both tables, Document and Anchor, are not real tables in the database.

Table Document is discovered from Meta tags in the header part of an HTML document. The following shows how those attributes fit in the HTML document, from "http://www.eng.auburn.edu":

<HEAD>

*<META NAME="**Content-Type**" CONTENT="**text/html**; charset=windows-1252">*

*<META NAME="**Description**" CONTENT="**Auburn University, College of Engineering, Network Services - Home Page**">*

*<META NAME="**Keywords**" CONTENT="**Auburn University, College of Engineering, Network Services, Newsletter, Departments, Centers, Organizations, Research, Policies, Class Pages, Computer Help, Personal Pages, Directory, Catalog, Libraries, Calendar, OASIS, Shareware Sites, Maps, Weather, News sites, Hot Links**,">*

*<TITLE>**College of Engineering Auburn University**</TITLE>*

</HEAD>

The information from above HTML code:

TITLE: ***College of Engineering Auburn University***

TYPE: ***text/html***

DESCRIPTION: ***Auburn University, College of Engineering, Network Services - Home Page***

KEYWORDS: ***Auburn University, College of Engineering, Network Services, Newsletter, Departments, etc.***

The TEXT field in table document contains the whole Web document.

Table Anchor consists of three fields: BASE, HREF, and LABEL. The field BASE is discovered from the URL of the Web document. The hyperlinks from the Web document are considered as entries in the field HREF. Entries for the field LABEL are from the hypertexts in the Web document.

The following is a part of an HTML document (from http://www.eng.aburn.edu) that the spider discovers with HREF and LABEL:

```
<A HREF="/" TARGET="_TOP">[ Eng Home ]</A>
<A HREF="/info/search.html" TARGET="_TOP">[ Search ]</A>
<A HREF="http://www.auburn.edu/main/au_campus_directory.html">[ AU Directory ]</A>
<A HREF="http://www.auburn.edu/hotline">[ AU Technology Hotline ]</A>
<A HREF="http://search.auburn.edu/">[ AU Search ]</A>
```

HREF and LABEL from the above HTML code are:

- HREF: ***http://www.eng.auburn.edu,*** LABEL: ***[Eng Home]***
- HREF: ***http://www.eng.auburn.edu/info/search.html,*** LABEL: ***[Search]***
- HREF: ***http://www.auburn.edu/main/au_campus_directory.html,*** LABEL: ***[AU Directory]***
- HREF: ***http://www.auburn.edu/hotline,*** LABEL: ***[AU Technology Hotline]***
- HREF: ***http://search.auburn.edu,*** LABEL: ***[AU Search]***

4.3.2 Path Regulation

The Tiny WebSQL query allows a user to specify the restrictions on the range of the Web site to retrieve Web information. This feature is called "path regulation". There are three ranges of path regulations: Interior, Internal, and External.

The user has to provide a reference Web site in which the spider compares the reference Web site with the currently navigated Web site to find if the Web site is in the specified range level. Figure 4.2 shows three ranges of path regulations.

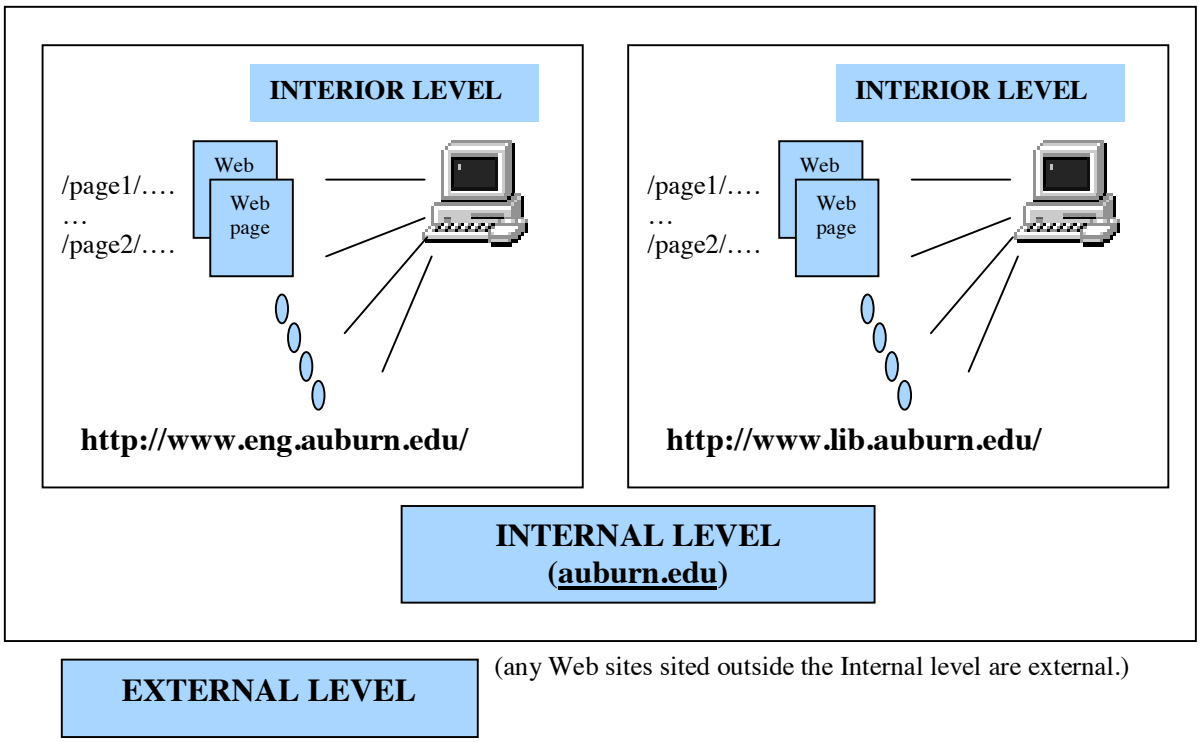


Figure 4.2: Three *ranges of the path regulation: Interior level, Internal level, and External level*

Interior Level

If the range is the Interior level, Web sites must be sited within the same server as the reference URL. Examples of the Interior level Web sites are:

- *http://**www.eng.auburn.edu*** *(reference Web site)*
- *http://**www.eng.auburn.edu**/info/search.html*
- *http://**www.eng.auburn.edu**/info/toc.html*
- *http://**www.eng.auburn.edu**/info/feedback.html*
- *http://**www.eng.auburn.edu**/info/mail.html*

Those Web sites are all sited on the same server "***www.eng.auburn.edu***."

Internal Level

If the range is the Internal level, Web sites must be sited within the same enterprise domain (the two right-most domains) of the reference. Examples of the Internal level Web sites are:

- *http://www.**auburn.edu*** *(reference Web site)*
- *http://www.eng.**auburn.edu***
- *http://www.eng.**auburn.edu**/info/search.html*
- *http://www.lib.**auburn.edu***
- *http://oasis.**auburn.edu***

Those Web sites are all on the same enterprise domain "**auburn.edu**."

External Level

If the range is the External level, Web sites must be sited in different domains. Examples of the External level Web sites are:

- *http://www.**auburn.edu*** *(reference Web site)*
- *http://www.**cisco.com***

Those Web sites are all on the different enterprise domain "**auburn.edu**" and "**cisco.com**."

4.3.3 Syntax

The syntax of the Tiny WebSQL Query language is based on the standard SQL Select statement. The syntax of the language is shown in Figure 4.3. All queries in the Tiny WebSQL Query language refer to model document structures that are explained in section 4.3.1. Therefore, tables in the language can only be Document and Anchor and fields can only be the attributes from those tables.

```
Query          := SELECT AttrList FROM DomainSpec
                  [WHERE Condition];
AttrList       := Attr {, Attr}
Attr           := Field | TableVar.Field
Field          := 'url' | 'title' | 'type' | 'description' | 'text' |
                  'keywords' | 'base' | 'href' | 'label'
TableVar       := Id
DomainSpec     := DomainTerm
DomainTerm     := Table TableVar SUCH THAT DomainCond
DomainCond     := Node PathRegExp TableVar |
                  TableVar MENTIONS StringConstant |
                  Attr = Node
Node           := StringConstant | TableVar
Condition      := BoolTerm {OR BoolTerm}
BoolTerm       := BoolTerm {AND BoolTerm}
BoolTerm       := Attr = StringConstant | Attr CONTAINS  StringConstant
PathRegExp     := -> | => | ~>
```

Figure 4.3: *The BNF specification of the Tiny WebSQL Query Language*

4.3.4 Semantics

The Tiny WebSQL query can be a basic SELECT statement, or a basic SELECT statement with the CONDITION portion.

Basic SELECT Statement

The basic structure of the Tiny WebSQL Query statement is shown as follows:

SELECT *xxxxx* ***FROM*** *yyyyy;*

The Tiny Spider collects Web document attributes xxxxx, which can be URL, TITLE, TYPE, DESCRIPTION, KEYWORDS, BASE, HREF, and LABEL. The meaning of each attribute is mentioned in section 4.3.1.

Table Instance Declaration

The yyyyy portion contains the table instance declaration and the basic condition. As mentioned in section 4.3.1, the two tables are:

- **Document** [URL, TITLE, TYPE, DESCRIPTION, KEYWORDS, TEXT], and
- **Anchor** [BASE, HREF, LABEL]

For instance,

SELECT *d.title, d.description* ***FROM*** *Document d* ***SUCH THAT*** *zzzzz;*

From the above statement, an instance, d, of table Document is declared. The language allows only one table instance in a Tiny WebSQL query, but fields from the

other table can be included in the SELECT statement, as shown in the following Select statement:

SELECT *d.type, d.description,* ***href, label*** *FROM Document d SUCH THAT zzzzz;*

Those two fields, **href** and **label**, which are not from the table Document, are allowed in the SELECT statement without the instance of the table Document ("**d**" in this case) in front of them. However, from the example above, the SELECT statement would obviously cause an error if the fields were mentioned as **d.href** and/or **d.label** because they are not from the table Document, but from Anchor.

Basic Condition (DomainCond)

In the portion of yyyyy, in addition to the table instance declaration, users have to include the basic condition, which is in the portion of zzzzz shown in the above SELECT statement. (The basic condition is also mentioned as "DomainCond" as in Figure 4.3). The following is an example of the basic condition in a Tiny WebSQL query:

SELECT d.title, d.description, href, label

FROM Document d

SUCH THAT ***d MENTIONS "Internet"****;*

The above SELECT statement shows that the basic requirement is to collect Web sites that mention the word "**Internet**". A semicolon must be included at the end of the statement.

Path Regulation

The basic condition can also be the path regulation, as shown in the following statement.

SELECT d.title, d.description, href, label

FROM Document d

SUCH THAT "http://www.eng.auburn.edu" ***-> d;***

From the above SELECT statement, the basic condition is to collect only the Web sites that are sited in the same enterprise domain as the reference Web site, "http://www.eng.auburn.edu". (Information about the Path Regulation is mentioned in section 4.3.2). The symbols used to represent three levels of path regulation are: **~>** (Interior), **->** (Internal), **=>** (External).

The Condition Portion

The following statement is an example of the condition portion in a Tiny WebSQL SELECT statement:

SELECT d.type, d.description, href, label

FROM Document d

SUCH THAT d MENTIONS "Internet"

WHERE

type = "text/html" AND description = "e-business" or description = "e-commerce";

From the above statement, in addition to the basic condition that specifies the Web document must mention the word "**Internet**", the condition portion also specifies that the document has to be of the type "**text/html**", and the document's description must contain the word "**e-business**". Otherwise, the Web document must mention the word "**Internet**" and the description must contain the word "**e-commerce**".

4.3.5 Examples of Tiny WebSQL Queries

The following are some uses of the Tiny WebSQL queries as the restrictions for the spider to collect Web information. Due to the security concerns at the Engineering Network Services of Auburn University, any Web sites outside the Engineering Network cannot be accessed to retrieve Web information from the network. Therefore, it is not possible to check all these Tiny WebSQL Query statement examples through the project Web page. Some examples need to run from the Unix command line.

The following are examples of the SELECT statements used as the restrictions on Web information:

EXAMPLE 1

Find HTML documents about "Auburn University".

```
SELECT    d.url, d.title, d.type, d.keywords, d.description, href, label
FROM      Document d    SUCH THAT d MENTIONS "Auburn University"
WHERE     type = "text/html";
```

EXAMPLE 2

Find all links that refer to "Jini" from documents about "Java".

```
SELECT    href
FROM      Document d SUCH THAT d MENTIONS "Java"
WHERE     label CONTAINS "Jini";
```

The following are examples of the SELECT statements used to be the restrictions on Web information and path regulation:

EXAMPLE 3

Find all HTML documents within the "www.eng.auburn.edu" server that refer to "Computer".

```
SELECT    d.url, d.title, d.type, d.keywords, d.description, href, label
FROM      Document d SUCH THAT "http://www.eng.aburn.edu" ~> d
WHERE     text CONTAINS "Computer" AND type = "text/html";
```

EXAMPLE 4

Find all HTML documents outside Auburn University network that refer to "map".

```
SELECT    d.url, d.title, d.type, d.keywords, d.description, href, label
FROM      Document d SUCH THAT "http://www.auburn.edu" => d
WHERE     text CONTAINS "map" AND type = "text/html";
```

5. Methods

This project avoids the various means of advanced search that would confuse users, by employing the Tiny WebSQL Query language as the restriction for collecting Web information. The Tiny WebSQL Query language is familiar to the database users since the syntax of the language follows the SQL language syntax. (This might be inconvenient for some end users. However, the use of SQL-like input is able to cover most of the features that many popular search engines can provide in their own different advanced searches.)

The methods used by this project belong to three different research areas: Spider, indexing, and search software.

Spider: The spider is the focus of the Tiny WebSQL search engine. It collects Web information according to the Tiny WebSQL Query statement specified by the user. The spider is called the Tiny Spider, which ignores a non-textual Web document; e.g. audio, photo, movie, etc.

Indexing: The indexing part is implemented in a basic fashion since it is not the focus of this project. Each Web site that has been visited by the spider and is in the textual format (i.e. plain text or HTML file) will be stored in the database of the search engine. The Web site's URL is used to be the unique key (index) for each entry in the database. Therefore, the Tiny WebSQL search engine never duplicates Web sites in the database.

Search Software: A search engine lets the spider crawl the Web and collect Web information before storing the information into the database. When a user wants to search for some Web sites with keywords, the search software will receive the keywords and perform the search on the database.

5.1 The Tiny Spider

The Tiny Spider navigates the Web and collects Web information according to restrictions composed by a user via the Tiny WebSQL query. The functions of the spider are listed as the following:

- Performs the syntax check on the Tiny WebSQL query before navigating the Web, which is started from the seed URL provided by the user.
- Keeps track of the number of Web sites visited (the maximum number of Web sites the spider can visit must be specified by users - see figure 6.1a).
- Skips collecting information if the Web document is not in textual format, or the Web sites have been visited.
- Analyzes Web documents and creates a model document structure (table "document" and table "anchor").
- Compares the information retrieved in the form of the model document structure of the currently visited Web site and the restrictions from the Tiny WebSQL query to make a decision whether to store the Web site into the database or to ignore it.

5.1.1 Web Site Crawling List

The Tiny Spider navigates Web sites and stores the URLs into a crawling list. The spider starts crawling and navigating the Web from a seed URL given by the user in "Seed URL" box, as shown in figure 6.1a. This seed URL will also be the first URL in the crawling list.

The spider checks the URL to make sure that the Web document is in textual format before analyzing the Web document. Only hyperlinks accompanied with hypertexts that have not been visited will be inserted into the crawling list. The number of hyperlinks from a Web document that can be inserted into the crawling list can be optionally specified via the "Hyperlink Limit" box as shown in Figure 6.1a.

The spider repeats the same steps to navigate following URLs in the crawling list using First come, First serve technique. The size of the crawling list is given by the user in the "Webpage Limit" box as shown in figure 6.1a.

5.2 Indexing

The Tiny WebSQL search engine has two tables in the database: Document table and Keyword table. The index (key) for the Document table is the URL of each Web site. The index used in the Keyword table is from two fields of each entry in the table: Field URL and field Keywords.

5.2.1 Data Structure

The database in this project, which consists of the table Document and the table Keyword, is shown in table 5.1 and 5.2, respectively. Table Document includes four attributes:

- **URL** from the Web Site's URL. It is used as the key of the table
- **TITLE** from the "TITLE" tag of that Web site
- **TYPE** from the "TYPE" tag of that Web site
- **DESCRIPTION** from the "DESCRIPTION" tag from that Web site

Table Document:

URL	TITLE	TYPE	DESCRIPTION
http://www.auburn.edu	Auburn University	Text/html	Auburn University ….
http://www.lib.auburn.edu.	Library.	Text/html.	Library of Auburn Uni…
http://www.eng.auburn.edu	Engineering Network	Text/html	Engineering Network…..
…..	……	……..	……

Table 5.1: *Table Document*

The following is the SQL statement that is used to create the table Document.

```
Create table document (
        URL varchar2 (85) NOT NULL Primary Key,
        TITLE varchar2 (40),
        TYPE varchar2 (25),
        DESCRIPTION varchar2 (55) );
```

Table "Keyword" includes two attributes:

- **URL** from the Web site URL
- **KEYWORDS** from the "KEYWORDS" tag of that Web site

Table Keyword:

URL	KEYWORDS
http://www.eng.auburn.edu	KEYWORD1
http://www.eng.auburn.edu	KEYWORD2
http://www.eng.auburn.edu	KEYWORD3
…..	…..

Table 5.2: *Table Keyword*

The following is the SQL statement that is used to create the table Keyword

```
Create table keyword (
        URL varchar2 (85) NOT NULL,
        KEYWORDS varchar2 (45) );
```

5.3 Search Software

The search software in this project requires users to specify a field in a Web document to be searched. The field can be: URL, title, type, description, or keywords, as shown in Figure 6.1b. There are two matching techniques used to find search results in this project. The first technique determines the search results from exactly matching the keywords with the information from the specified field in the database. This technique is called "exact match". The other technique does not require the exact match. This technique is called "ordered character match", which is explained in the second part of section 5.3.1. When the search software has results for the user, it ranks the results before returning them to the user. The method used to rank the results is called relevancy ranking. The relevancy ranking used in the search software ranks the search results based on the similarity of the keywords and the information stored in the database.

5.3.1 Matching Methods

There are two purposes for the matching methods used in the project. One is to check Web information with the restrictions specified by the user when the spider navigates the Web. Another purpose is to find the search results from the database (after the spider has navigated and stored the search results into the database), which is performed by the search software.

There are two matching techniques used in the Tiny WebSQL search engine:

- Exact Match
- Ordered Character Match

Exact Match

The exact match is used when the spider is navigating the Web and when the search software is looking for information in the fields URL, TYPE, or KEYWORDS from the database. The hits from the exact match must be from the exact matching of the given keywords with the information from the Web site or the information stored in the database. Otherwise, the search will not consider it as a hit. The exact match is not case sensitive in the Tiny WebSQL search engine. There is no need to rank the results from the exact match.

Ordered Character Match

The ordered character match is used when searching for information in the fields TITLE or DESCRIPTION from the database. It is a proprietary matching technique performed by the search software in the Tiny WebSQL search engine.

The idea of the ordered character match is from the idea of LCS (Longest Common Substring), but it is less time consuming since it does not concern itself with finding the longest sub string, but the similarity of two strings.

The process of the ordered character match starts from setting the first character in the keyword to be the main character before moving to the next character within the

keyword when the same character in the other string is found. The check keeps going until the search software reaches one end of the string.

The percentage of the string similarity is calculated from the number of main characters and the length of the keyword. The relevancy ranking is assigned according to the percentage of the string similarity.

For instance:

Keyword**: Auburn University* ***(length of 16 - not include white spaces)

***String to compare** (from the database): Auburn Bus Station*

The main character** -> Auburn Un* ***(length of 8)

From

***The matched characters** -> **Auburn** Bus Station*

***The relevancy ranking** -> **50%** (from **8/16**)*

6. Interfaces and Experimental Results

Section 6.1 shows the interfaces of this project: My Tiny Spider, My Tiny Search Software, and Checking My Downloads. Some experimental results are shown in section 6.2.

6.1 Interfaces

There are three major interfaces for this project:

- My Tiny Spider: activates the Web Spider;
- My Tiny Search Software: searches the database populated by the Tiny Spider; and
- Checking My Downloads: validates the information downloaded.

6.1.1 My Tiny Spider

My Tiny Spider retrieves the Tiny WebSQL query from the "Tiny WebSQL Query Statement" box as the restrictions for the spider to collect Web information as well as the path regulation. The Tiny Spider also needs the beginning Web site to start crawling. Therefore, the user has to assign a URL in the "Seed URL" box, in Figure 6.1a. The user also has to specify the size of the crawling list in the "Webpage Limit" box. As optional, the project allows the user to specify the maximum number of links referred in a Web document that the spider will insert into the crawling list. This number can be assigned in the "Hyperlink Limit" box.

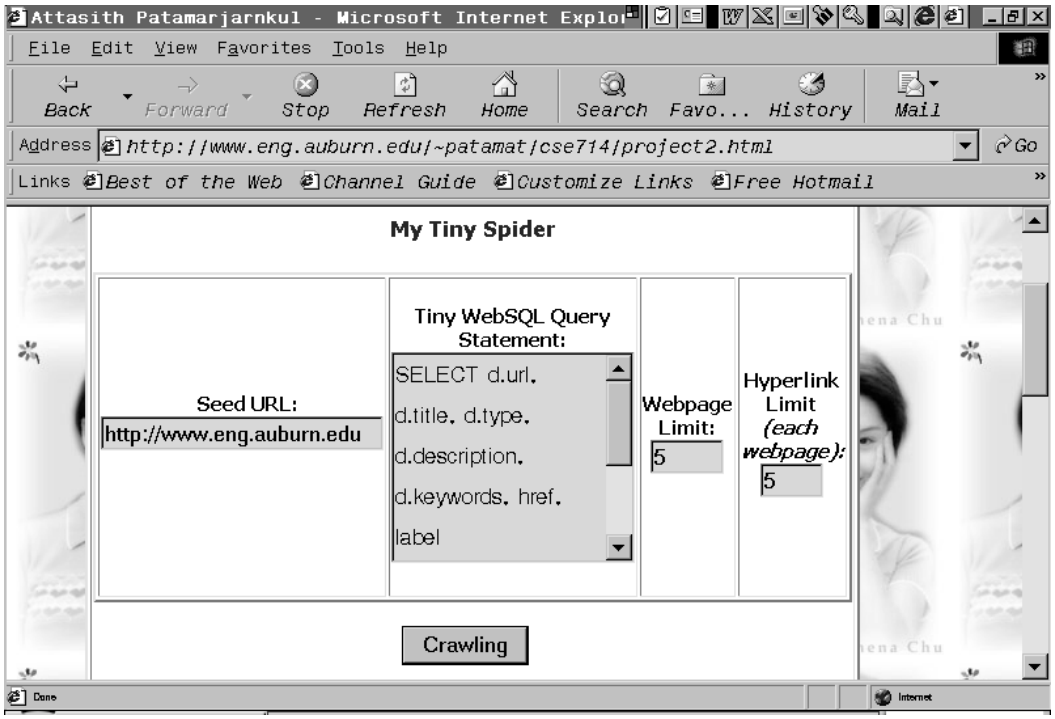


Figure 6.1a: *The user interface for My Tiny Spider*

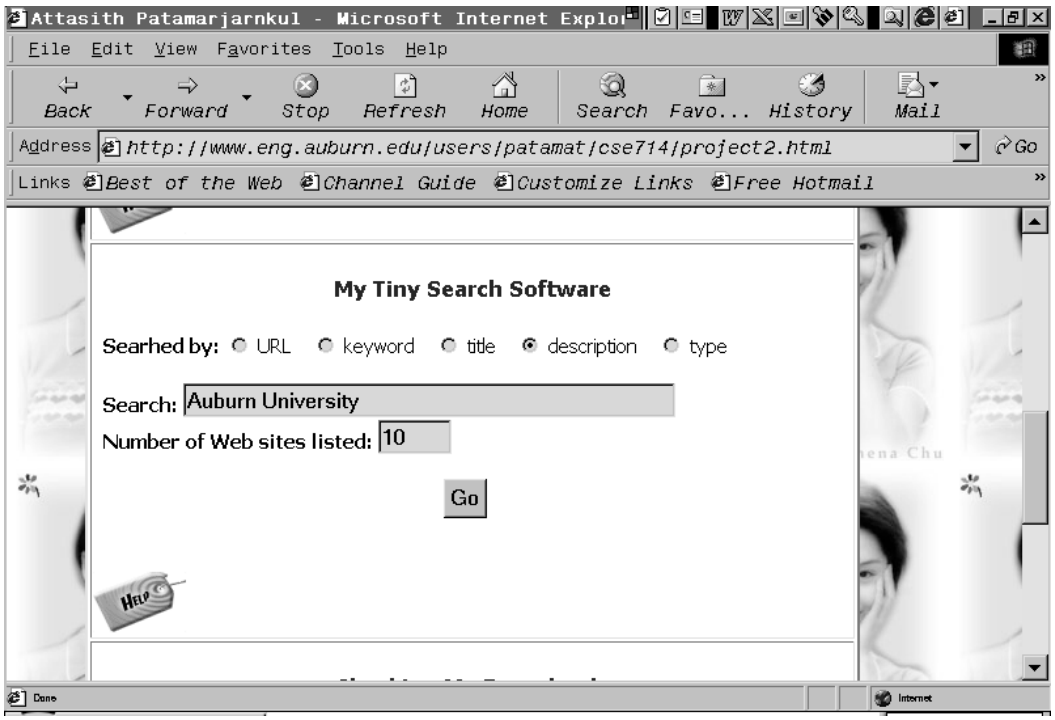


Figure 6.1b: *The user interface for My Tiny Search Software*

6.1.2 My Tiny Search Software

This part performs the tasks of the search engine that most people are familiar with. My Tiny Search Software receives keywords from a user and performs the search in the database. The user has to follow the following steps to search for information:

1: Select one Web document field with which the search engine will perform the search. The field to be selected may be: "URL", "keyword", "title", "description", or "type", as shown in Figure 6.1b.

2: Enter keywords into the "Search" box so that My Tiny Search Software will look up the keywords in the specified field (from STEP1) in the database.

3: Enter the number of maximum search results in the "Number of Web sites listed" box.

4: Click the "**Go**" button to start searching.

The user interface for My Tiny Search Software is shown in Figure 6.1b.

6.1.3 Checking My Downloads

This part is the tool to check all the information stored in the database in which the fields to be shown must be selected. As in Figure 6.1c, the fields may be the combination of the fields URL, keyword, title, description, or type. The "List_All" button is for the user to check all entries in the database. The "List" button is to check for information of a specific Web site that is specified in the "from site" box.

Checking My Downloads also provides an auxiliary tool to trace all Web sites that the spider has navigated. By clicking on the "Trace Logfile(web)" button, the attributes specified in the Tiny WebSQL query of each Web site will be shown.

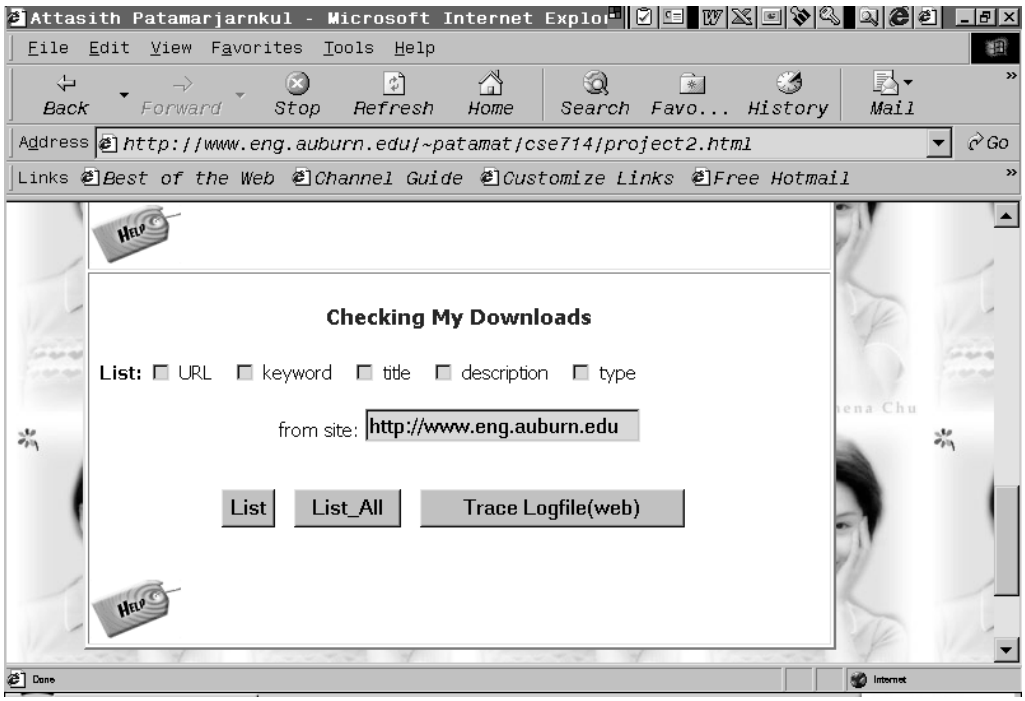


Figure 6.1c: *The user interface for Checking My Downloads*

6.2 Experimental Results

This section shows some experimental results from the Tiny WebSQL search engine.

Search Results

Figure 6.2a shows the search results for the keyword "Auburn University Map" from the "title" field. The search is performed by "My Tiny Search Software". The results include four types of information:

1. Percentage of similarity: This value is shown when searching in the fields: Title and description,
2. Document title (as hypertext),
3. Document description, and
4. Document type

Each document from the search results is ranked according to its similarity percentage, which is shown at the beginning of each search result. The similarity percentage is then followed by the document title as an underlined hypertext. In fact, the description and the type of a document are included as a part of a search result. However, if the document does not contain the description and the type, that particular search result will show only the similarity percentage and the document title.

Attasith Patamarjarnkul: Tiny WebSQL Seach Engine

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favo... History Mail

Address http://roadrunner.eng.auburn.edu/cgi-bin/patamat/websql/search.sh Go

Links Best of the Web Channel Guide Customize Links Free Hotmail

TINY WEBSQL SEACH ENGINE - SEARCHING RESULTS

SEARCHING FROM THE DATABASE (JDBC)
SIZE: 10

94% CHEMICAL ENGINEERING DEPARTMENT AUBURN UNIVERSITY ALABAMA

94% COLLEGE OF ENGINEERING AUBURN UNIVERSITY E-MAIL ADDRESS

84% COLLEGE OF ENGINEERING AUBURN UNIVERSITY
AUBURN UNIVERSITY COLLEGE OF ENGINEERING NETWORK SERVICES - HOME PAGE
TEXT/HTML

Attasith Patamarjarnkul: Tiny WebSQL Seach Engine

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favo... History Mail

Back to Attasith Patamarjarnkul

Address http://roadrunner.eng.auburn.edu/cgi-bin/patamat/websql/search.sh Go

Links Best of the Web Channel Guide Customize Links Free Hotmail

84% AUBURN UNIVERSITY COLLEGE OF ENGINEERING QUERYING

84% AUBURN UNIVERSITY COLLEGE OF ENGINEERING TABLE OF CONTENTS

84% COLLEGE OF ENGINEERING AUBURN UNIVERSITY
AUBURN UNIVERSITY COLLEGE OF ENGINEERING NETWORK SERVICES - HOME PAGE
TEXT/HTML

31% AUBURN AEROSPACE ENGINEERING

10% AGRICULTURAL ENGINEERING

Figure 6.2a: *Results from searching the title field with the keyword "Auburn University Map"*

There are 8 documents in the search results in Figure 6.2a. The similarity percentage of each document is calculated from the similarity between the keywords "Auburn University Map" and the document title. The first two results, as in Figure 6.2a, do not have description and type. The results of both documents have only the similarity percentage and the underlined hypertext title of the documents.

The third document does contain the document title, as well as description and type. Therefore, the third result contains all information: the similarity percentage, title, description, and type. The similarity percentage of each document is calculated according to the technique used in the "ordered character match", as in section 5.3.1. The number of results is limited by the number entered in the "Number of Web sites listed" box, as in Figure 6.1b.

Crawling List

The Tiny WebSQL search engine allows a user to examine the Web sites whose URLs stored and ordered in the crawling list. The information from an examined Web site consists of:

- The order number in the crawling list: this number shows the position of the Web site's URL in the crawling list. The first position in the crawling list is 0, which also implies that the URL is the seed URL.
- The Web site's URL: follows the order number in the first line (as in Figure 6.2b),
- The basic condition from the Tiny WebSQL query,

- The information from the Web site: the information is in a table, which has two columns. The first column contains the attributes from the Tiny WebSQL query and the second column contains the corresponding information from the Web site.

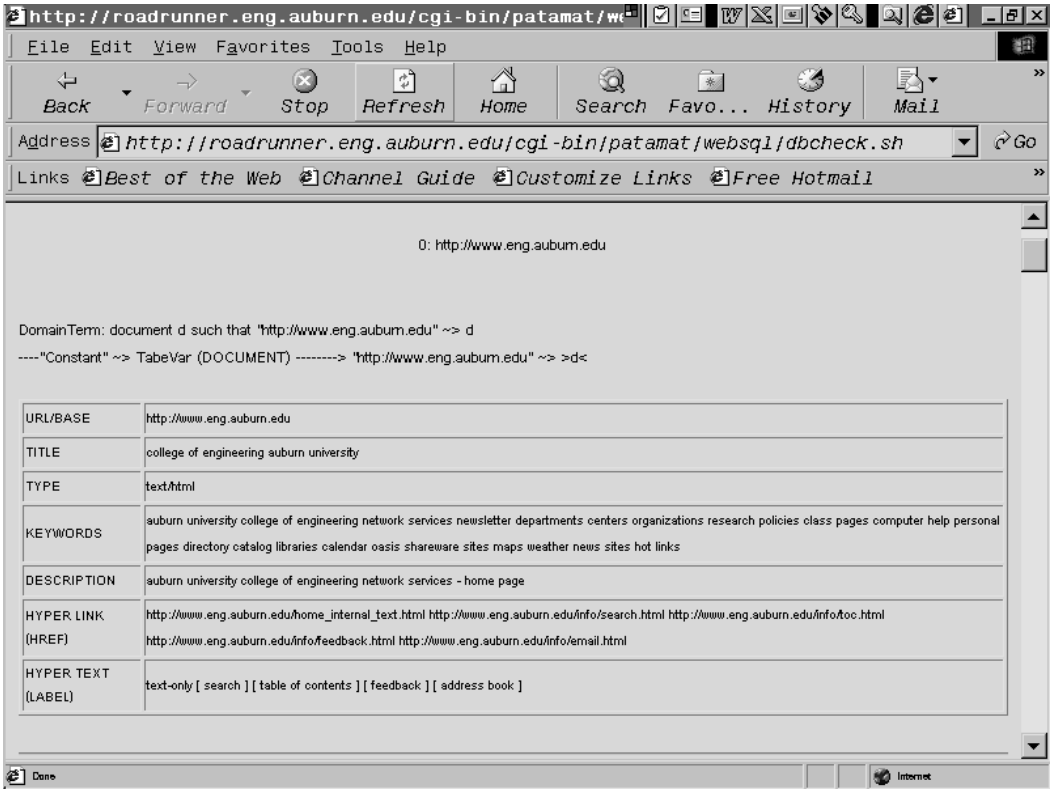


Figure 6.2b: *The tracing results from one of the Web sites in the crawling list.*

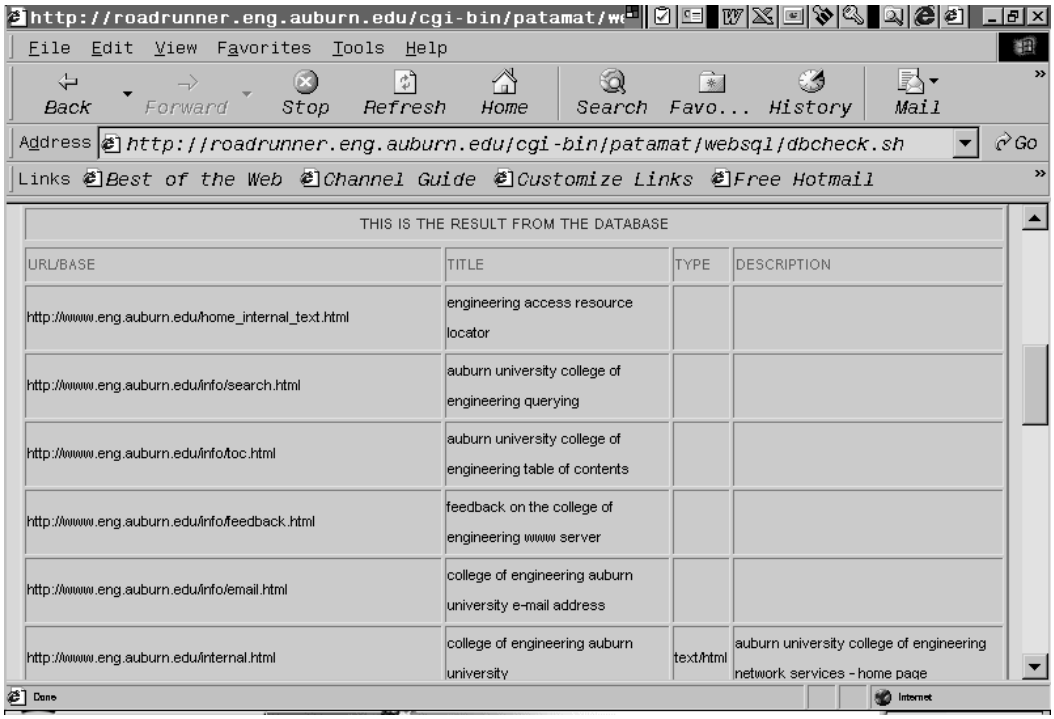
The examined Web site from Figure 6.2b is "http://www.eng.auburn.edu". Number "0" at the beginning of the first line implies that the Web site's URL is stored in the first position in the crawling list. The table in Figure 6.2b shows information of the attributes specified in the Tiny WebSQL query, including:

- URL/BASE: The URL of the Web site,
- TITLE: The title of the Web site
- TYPE: The type of the Web site

- KEYWORDS: The keywords specified in the Meta tag of the Web site
- DESCRIPTION: The description specified in the Meta tag of the Web site
- HYPERLINK (HREF): The links (in URL form) from the Web site
- HYPERTEXT (LABEL): The hypertext (label) of the links from the Web site

Checking My Downloads

The database contents can be displayed from the interface "Checking My Downloads". Figure 6.2c shows the result after clicking on the "List_All" button, as in Figure 6.1c. If there is no information for some particular field of a Web document, the corresponding field in the database will be blank.



THIS IS THE RESULT FROM THE DATABASE

URL/BASE	TITLE	TYPE	DESCRIPTION
http://www.eng.auburn.edu/home_internal_text.html	engineering access resource locator		
http://www.eng.auburn.edu/info/search.html	auburn university college of engineering querying		
http://www.eng.auburn.edu/info/toc.html	auburn university college of engineering table of contents		
http://www.eng.auburn.edu/info/feedback.html	feedback on the college of engineering www server		
http://www.eng.auburn.edu/info/email.html	college of engineering auburn university e-mail address		
http://www.eng.auburn.edu/internal.html	college of engineering auburn university	text/html	auburn university college of engineering network services - home page

Figure 6.2c: *The database contents from Checking My Downloads.*

7. Conclusions and Future Works

The conclusions for this project are given in section 7.1. Section 7.2 proposes some future works for improving the search and for extending the Tiny WebSQL Query language.

7.1 Conclusions

The major components of a search engine are spider, indexing, and search software. The focus of this project is on the spider and the Tiny WebSQL Query language. The relevancy ranking method used in the search software is based on a basic technique, which can be replaced by a more effective relevancy ranking module to improve the system since the proposed system is constructed modularly.

One of the advantages of the Tiny WebSQL search engine is using the Tiny WebSQL Query language as the restriction for the spider to collect Web information. The syntax used in the language also allows a user to specify the range of Web sites to be searched. This approach makes the search engine a customized search engine, which can be used as a special purpose search engine; e.g. customer-data collection.

The following gives a summary of the Tiny WebSQL search engine:

Type of search: Keyword,

Search options: Advanced,

Domains searched: World Wide Web,

Search refining: Boolean "AND", "OR", Path Regulation,

Relevancy ranking: Results ranked according to the similarity of strings (not including the number of times or the positions of the keywords appearing in the Web document),

Result to be presented: Title (as the hypertext), Description, and Type,

Advantages: Customized, no URL duplicate entries in the database, and able to be improved to a special-purposed search engine, and

Disadvantages: Search speed not optimal.

7.2 Future Works

There are a number of features in the Tiny WebSQL search engine that can be enhanced.

7.2.1 Path Regulation Enhancements

As with the path regulation referred to in section 4.3.2, the project supports only one step of Path Regulation in the sense that the reference Web site is directly compared to the currently navigated Web site. From the BNF specification in Figure 4.3:

DomainCond *:= Node ~>|->|=> TableVar*

The syntax of the Tiny WebSQL Query language used in this project does not allow users to regulate the path beyond the three predefined path regulations; ~>, ->, =>. To make the TinyWebSQL Query language more flexible, there are a number of functions that have to be added to the path regulation in the Tiny WebSQL Query language:

- Path Regulation Repetition (**PathRegExp** := PathRegExp*)
- Path Regulation Alternation (**PathRegExp** := PathRegExp"|" PathRegExp)
- User-defined Path Regulation Precedence (**PathRegExp** := (PathRegExp))

The following is an example using the enhanced Tiny WebSQL Query language.

*The Web site must be in the external level, **=>**, from Web sites that are sited within the same server, **~>**, as the reference Web site "http://www.auburn.edu".*

SELECT d.url

FROM Document d

*SUCH THAT "http://www.auburn.edu " **~> =>** d;*

7.2.2 More Instances of Model Document Structures

As mentioned about the model document structure in section 4.3.1, the capability of the spider in this project might be enhanced by allowing users to assign more than one instance of table (model document structure). As shown in figure 7.1:

DomainSpec *:= DomainTerm {, DomainTerm}*

DomainTerm *:= Table TableVar **SUCH THAT** DomainCond*

The DomainTerm (from the BNF specification of the enhanced Tiny WebSQL Query language, Figure 7.1) can be assigned into a Tiny WebSQL query for more than one term. Since each DomainTerm contains the declaration of a table instance, more than one instance of the tables can be declared.

Since restrictions can be from various table instances, the restrictions do not limit to only the comparison of a table instance with specified constant string. For instance:

Show Web sites that are outside the Web site "http://www.auburn.edu" and must have the same description as the title of Web sites outside the Web site "http://www.eng.auburn.edu".

SELECT *d.url*

From Document *d* ***SUCH THAT*** *"http://www.auburn.edu" => d,*

Document *e* ***SUCH THAT*** *"http://www.eng.auburn.edu" => e*

WHERE *e.title = d.description;*

From the above statement, there is a new comparison of two different fields from different table instances: *e.title = d.description*.

7.2.3 Condition Enhancements

The precedence of the Boolean terms for the conditions in the Tiny WebSQL Query language in this project only allows for the AND operator to have higher precedence than the OR operator. To make the language more flexible, an enhancement for the condition precedence, as shown below, should be introduced into the language.

BoolTerm := Attr **=** StringConstant | Attr **CONTAINS** StringConstant | <u>(Condition)</u>

7.2.4 Enhanced Tiny WebSQL Query Language Syntax

```
Query          := SELECT AttrList FROM DomainSpec
                  [WHERE Condition];
AttrList       := Attr {, Attr}
Attr           := Field | TableVar.Field
Field          := 'url' | 'title' | 'type' | 'description' | 'text' |
                  'keywords' | 'base' | 'href' | 'label'
TableVar       := Id
DomainSpec     := DomainTerm {, DomainTerm}
DomainTerm     := Table TableVar SUCH THAT DomainCond
DomainCond     := Node PathRegExp TableVar |
                  TableVar MENTIONS StringConstant |
                  Attr = Node
Node           := StringConstant | TableVar
Condition      := BoolTerm {OR BoolTerm}
BoolTerm       := BoolTerm {AND BoolTerm}
BoolTerm       := Attr = StringConstant | Attr CONTAINS  StringConstant | (Condition)
PathRegExp     := Link | PathRegExp* | PathRegExp "|" PathRegExp | (PathRegExp)
Link           := -> | => | ~>
```

Figure 7.1: *The BNF specification of the Enhanced Tiny WebSQL Query language*

7.2.5 Other Enhancements

There are some other features that could be developed further from the Tiny WebSQL Search Engine, but are referred to in the BNF specification in Figure 7.1.

Fields in MDS Document

Fields in the MDS Document can include: Length, modif, etc. These new fields are from the HTML document. The length field specifies the upper bound of the size of the document, and the modif field specifies modifications on the document; e.g. the last modified date, etc.

Relevancy Ranking

As specified in chapter 5, this project does not seriously consider the relevancy ranking technique. However, to enhance the performance of the search in the Tiny WebSQL search engine, some better relevancy ranking methods, such as LCS or LACS, should be introduced into this project.

References

[1] "**Alta Vista Advanced Search Tutorial**",

http://doc.altavista.com/adv_search/ast_as_sort.shtml

[2] "**Alta Vista - help - Advanced Search**" ,

http://doc.altavista.com/help/search/adv_help.shtml

[3] Ilan Greenberg and Lee Garber, "**Searching for new search technologies**",

Computer, pp. 4-11, August 19999

[4] "**A Helpful Guide to Web Search Engine**",

http://www.monash.com/spidap4.html

[5] Alberto O. Mendelzon, George A. Mihaila, Tova Milo, "**Querying the World Wide Web**", Proceedings of PDIS'96, Miami, Florida.

[6] Allison Woodruff, Paul M. Aoki, Eric Brewer, Paul Gauthier, Lawrence A. Rowe, "**An Investigation of Documents from the World Wide Web**", Fifth International World Wide Web Conference, May 6-10, 1996, Paris, France

[7] Wen Chen Hu, "**Introduction and Web Document Analysis (Class Slides)**",

http://www.eng.auburn.edu/~wenchen/course/comp714/week1/figure/slide1.gif to

http://www.eng.auburn.edu/~wenchen/course/comp714/week1/figure/slide13.gif

[8] Wen Chen Hu, "**Web Search Engines (Class Slides)**",

http://www.eng.auburn.edu/~wenchen/course/comp714/week2/figure/slide1.gif to

http://www.eng.auburn.edu/~wenchen/course/comp714/week2/figure/slide15.gif

[9] Wen Chen Hu, "**Spiders and System Interface (Class Slides)**",

http://www.eng.auburn.edu/~wenchen/course/comp714/week3/figure/slide1.gif to

http://www.eng.auburn.edu/~wenchen/course/comp714/week3/figure/slide7.gif