Full Text Indexing Technology Applying to the Information System of Supplementary Jr.

Chun-Liang Tung (董俊良), Wei-Hurng Yaur (姚威宏), Jung-Chung Hung (洪榮崇)
Chiu-Ping Lin (林秋萍), and Pei-Yi Leu (呂佩怡)

Department of Information Management, Department of Mechanical Engineering of National Chin-Yi Institute of Technology, Taichung, Taiwan, R.O.C.

Abstract

The purpose of this paper is to improve IBM *Storage And Information Retrieval System* (*STAIRS*) and use the modified theory (Full Text Indexing Technology) to design a full-text-index Search Engine on the Internet. The design and implementation of the Full Text Indexing Technology (FTIT) can help people to efficiently search for useful information on the huge Internet.

The FTIT Search Engine consists of the *Spider*, the *Database*, and The *Search Tool*. The Spider, which delves and retrieves the Web content from the Internet, stores all of the *valid tokens* to a database of FTIT Search Engine. The Database, which is a kind of knowledge base database, collects a great deal of information about valid tokens such as the name of valid tokens, the attributes of valid tokens, and the addresses of Home Pages. The Search Tool, which is a search utility, looks for information in the database of FTIT Search Engine and responds the result of the searching task to users via CGI.

In order to achieve the purpose of explaining the function of FTIT, the paper will use Visual C++, standard Common Gateway Interface, Hypertext Markup Language, and Windows NT network to demonstrate

Keywords: Storage And Information Retrieval System(STAIRS), Full Text Indexing Technology (FTIT), Spider, Database, Search Tool, valid tokens, Hypertext Markup Language(HTML)

1. Introduction

There is so much information on the Internet that it may be difficult for people to find what would be most useful to them. Consequently, search engines are developed to search particular information for people. Many perfect search engines are available on the Web search sites of the Internet, such as Excite, HotBot, and Yahoo. They usually do a good job in searching and finding much useful information. But too much valuable time, perhaps up to 70%, is spent seeking out unimportant information in the search process [1]. We need new ideas, theories, and techniques to develop new search engines which can handle linguistic meanings and processes.

Several different technologies have been developed to facilitate the search for information within a search engine database such as Boolean search, Fuzzy Boolean search, and Vector-based search [2]. We can improve IBM Storage And Information Retrieval System (STAIRS) [3] and use this modified

theory to create the spider of a search engine, the database of a search engine, and the search tool of a search engine (a search engine consists of spiders, database, and search tool) .

The theory of this paper is that we can successfully find the character relation between words which then can be developed to a meaningful search engine, and the specific way to search the particular data in a huge database. This paper may help foretell the lines of development of the indexing database. It is very important to know the space of a hard disk before an indexing database to occupy all of the hard disk [4].

2. Storage and Information Retrieval System

Actually, STAIRS is an application of inverted files, which are a kind of multilist. In index sequential files, we usually use a single primary key to retrieve data because each index entry identifies unique data in the main file. But now, we want to use multikey to retrieve data because a secondary index provides a faster way to locate all the records with a particular attribute value.

2.1 Threaded Files, Multilists, and Inverted Files

In a threaded file, a pointer field is associated with each index secondary key field. The value of each secondary index key identifies the next record with the same value of the secondary key. In Figure 1, the initial value of the $Attribute_1$ is $Value_1$ and its secondary key (next pointer) is pointed to $Value_3$. Thus, a number of threads run through the file. Using this technique, we can get complete information which has the same attribute.

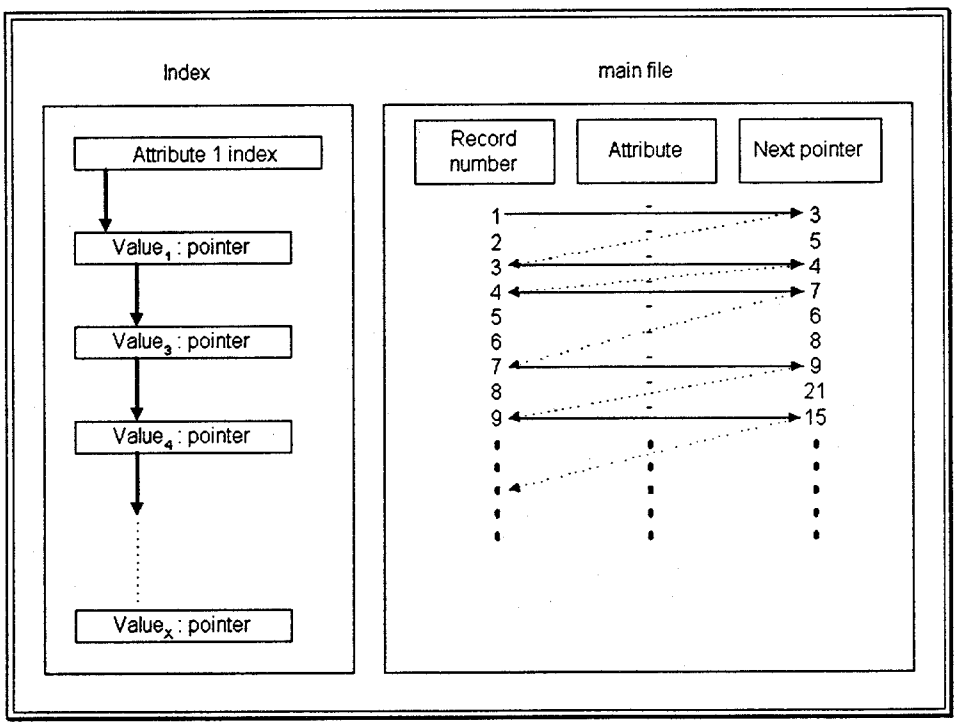


Figure 1. General Threaded File

A threaded file seems to use a single list to retrieve data which have the same attribute. The second technique uses a multilist to retrieve data. A multilist which has the same structure as in the threaded file organization, uses different index entries. In Figure 2, we can see that it uses a pointer list to indicate the same value of the attribute. Smith & Barnes[3, p. 189] describe the difference between a threaded file and a multilist: "Instead of an index entry pointing simply to the beginning of a thread, it now points to every Kth record on the thread (for some value of K). That is, the index entry for a particular attribute value identifies every K record with that value."

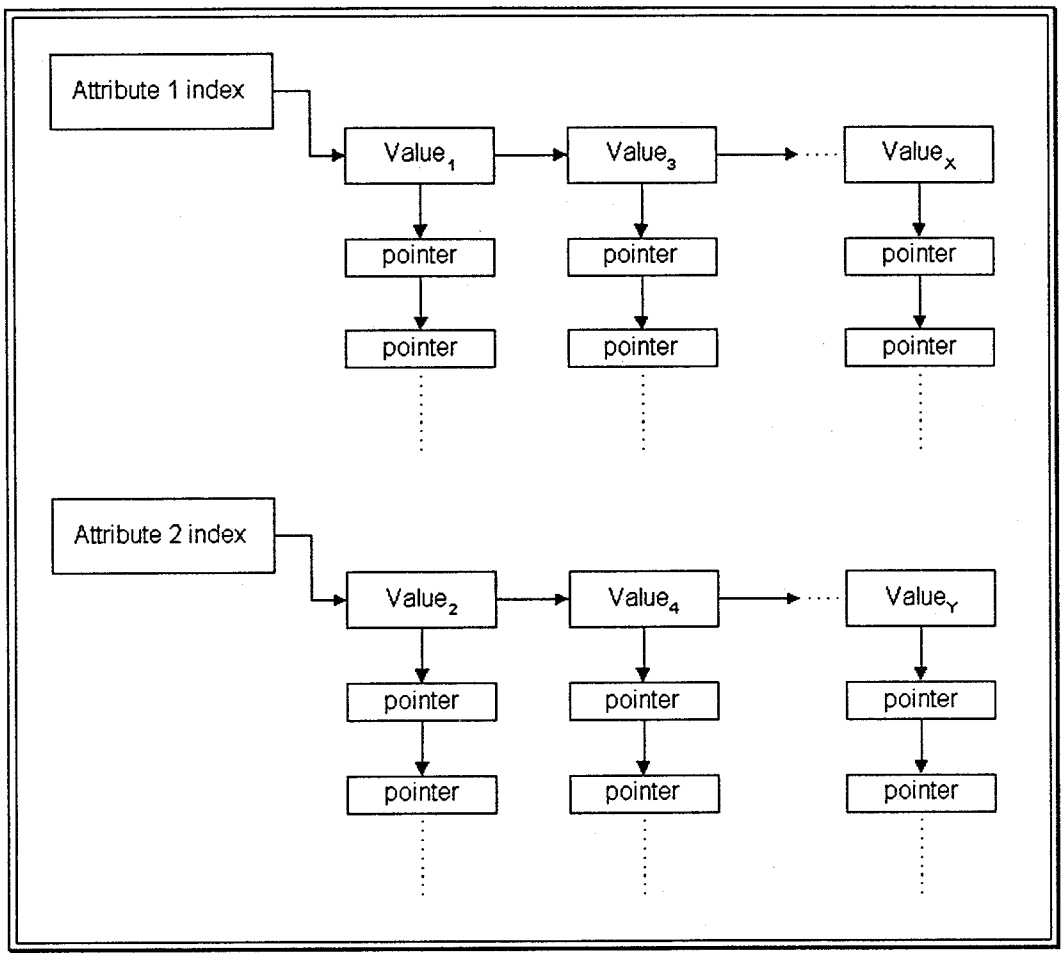
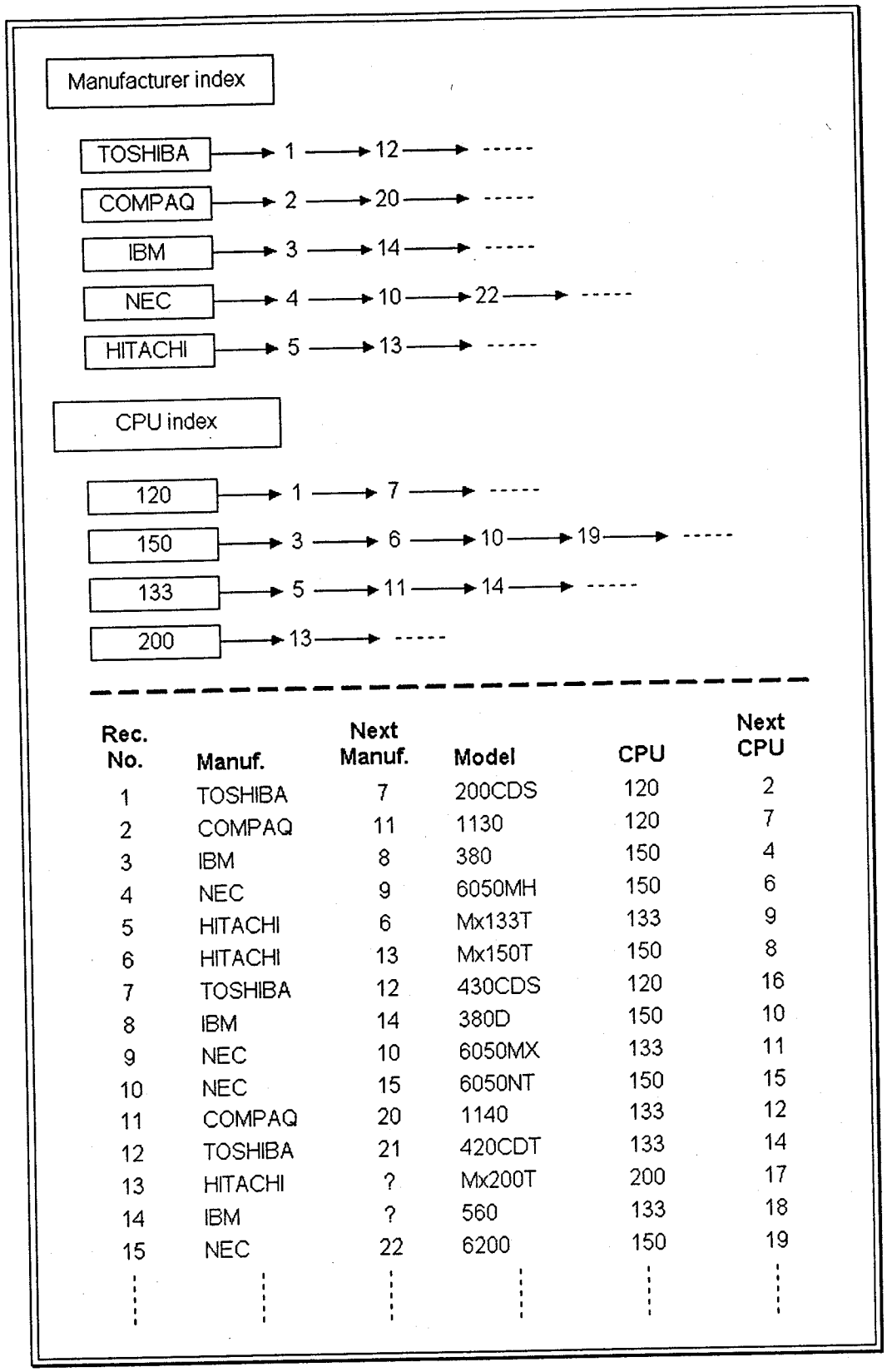


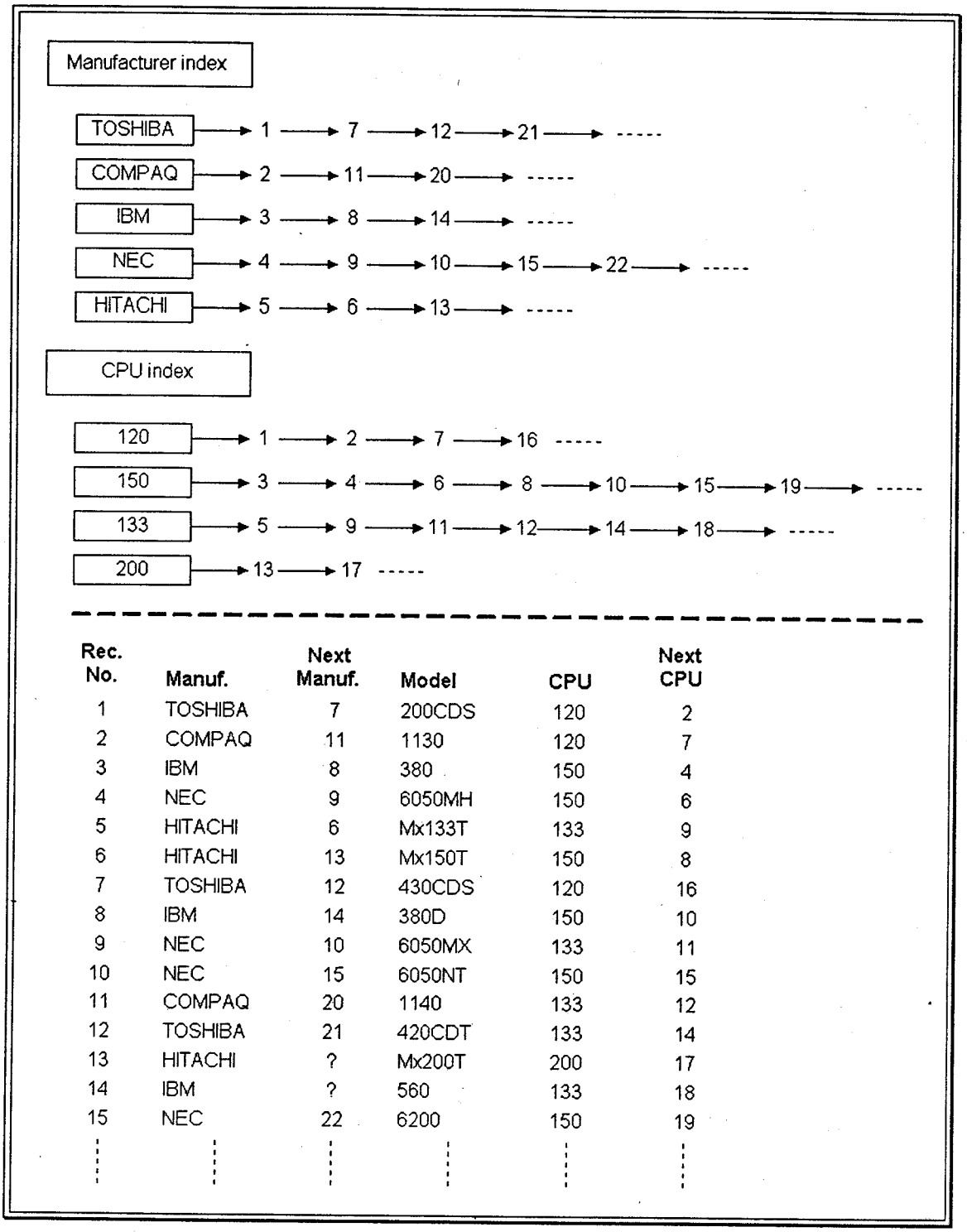
Figure 2. General Multilists

Figure 3 shows part of a file of computer records with two representative indexes and uses a value of 2 for K. We can use K=∞ to represent one extreme of the multilist organization, and we can use K=1 to represent the other extreme of the multilist. When K=1 is set in a multilist organization, we call it an Inverted File. Figure 4 shows part of a file of computer records with two representative indexes and uses a value of 1 for K.



Rec. No.	Manuf.	Next Manuf.	Model	CPU	Next CPU
1	TOSHIBA	7	200CDS	120	2
2	COMPAQ	11	1130	120	7
3	IBM	8	380	150	4
4	NEC	9	6050MH	150	6
5	HITACHI	6	Mx133T	133	9
6	HITACHI	13	Mx150T	150	8
7	TOSHIBA	12	430CDS	120	16
8	IBM	14	380D	150	10
9	NEC	10	6050MX	133	11
10	NEC	15	6050NT	150	15
11	COMPAQ	20	1140	133	12
12	TOSHIBA	21	420CDT	133	14
13	HITACHI	?	Mx200T	200	17
14	IBM	?	560	133	18
15	NEC	22	6200	150	19

Figure 3. Multilists with K=2



Rec. No.	Manuf.	Next Manuf.	Model	CPU	Next CPU
1	TOSHIBA	7	200CDS	120	2
2	COMPAQ	11	1130	120	7
3	IBM	8	380	150	4
4	NEC	9	6050MH	150	6
5	HITACHI	6	Mx133T	133	9
6	HITACHI	13	Mx150T	150	8
7	TOSHIBA	12	430CDS	120	16
8	IBM	14	380D	150	10
9	NEC	10	6050MX	133	11
10	NEC	15	6050NT	150	15
11	COMPAQ	20	1140	133	12
12	TOSHIBA	21	420CDT	133	14
13	HITACHI	?	Mx200T	200	17
14	IBM	?	560	133	18
15	NEC	22	6200	150	19

Figure 4. Multilists with K=1

2.2 The Principles of STAIRS

IBM STAIRS (Storage And Information Retrieval System) is a kind of application of inverted files. Using this system, One can retrieve documents containing an arbitrary word, or do complex queries. For example, consider retrieval of a document which contains a word "apple" simple query or

"apple and orange" complex query. By modifying the theory of STAIRS, we can create a full-text document retrieval system and this is the main point of this paper.

There are five levels of data structures or files in the STAIRS system, as depicted in Figure 5. The first level of STAIRS is called Matrix which has 26×27 entries. In the matrix level, each of the entries identifies the start of the second level, the dictionary level, for words beginning with a particular pair of letters. The twenty-seventh column of the matrix level is reserved for one-letter words.

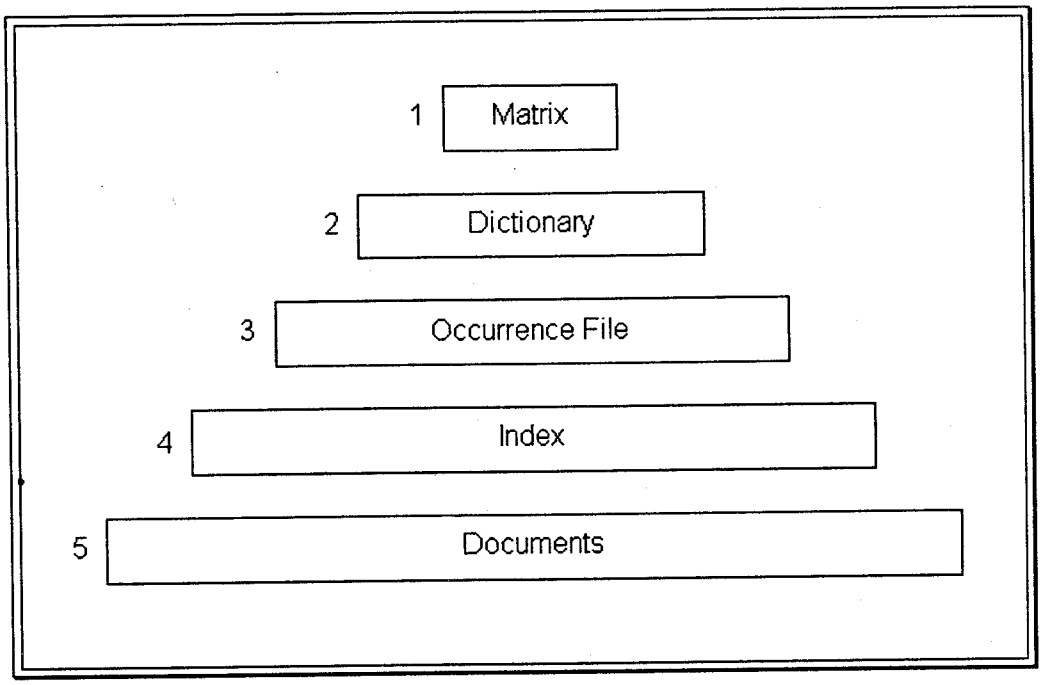


Figure 5. STAIRS Structure Hierarchy

The second level of STAIRS is called Dictionary, which contains an entry for each words collected in the matrix level. What information is stored in the dictionary level? It may contain, for example, the number of times the word occurs and the number of different document, where it occurs. The third level of STAIRS is called Occurrence File, which contains one record for each word occurrence in the dictionary level collection. According to Smith & Barnes[3, p.196], the information recorded for each word occurrence is "document number, paragraph code, sentence number, and position within sentence."

The fourth level of STAIRS is called Index which has one entry for each document. The entry contains much useful information such as the date of the entry into the system and a pointer to the document in the documents level. The fifth level of STAIRS is called Documents which contains machine-readable documents. The documents in the Documents level are a little bit different than conventional documents. They are tagged with a label such as TITLE, TEXT, ABSTRACT, and so on. It also puts an end-of-sentence code after each complete sentence [3, chp. 7].

Figure 6 presents a partial example of top three levels of STAIRS. Suppose we want to look

for a word "acaleph". First, we trace the relation from the field "ac" of the matrix level to the dictionary level and we find a pointer which points to a record "academic" in the dictionary level. Then, we start searching for the word "acaleph" from the position of "academic" in the dictionary level. We find the word "acaleph" exists in the dictionary level and it occurs a total of 7 different times and in 5 different documents.

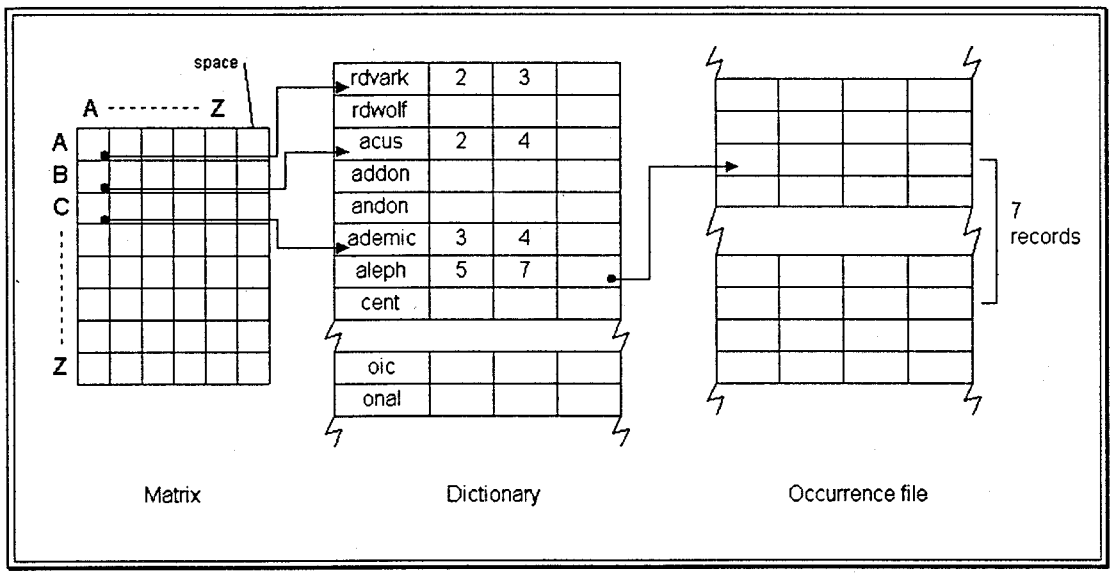


Figure 6. Top Three Levels of STAIRS

2.3 The Advantages and Disadvantages of STAIRS

One advantage of STAIRS is that users do not need to go through the whole system to search a word. All the work can be done in the top three levels of STAIRS. In fact, users can only use the matrix level and dictionary file level to find out how often a word occurs and in how many documents. The other advantage of STAIRS is that it is easy to design. Programmers, who design database applications, do not need to use either database tools or languages to design a database system. They can use common languages such as Visual BASIC, C, or C++ with the theory of STAIRS to design an information retrieval system.

One disadvantage of STAIRS is that it still has the drawback of a sequential search in the dictionary level and the occurrences file level, but this is not a big problem. We can use "random access" and "hash" to speed up the data retrieval. Another disadvantage of STAIRS is that the size of matrix level is too small. The original theory of STAIRS uses a two-dimension matrix level as a thumb index. If it can use a three- or four-dimension matrix level, it could increase the speed of the data retrieval of the dictionary level. The other disadvantage of STAIRS is that it is hard to maintain the indexing for each attribute value. According to Smith & Barnes[3, p.198], we can use bit vectors and a general graph structure to do a simple multilist.

3. Full Text Indexing Technology

FTIT (Full Text Indexing Technology), improved by IBM STAIRS, was developed for searching information efficiently on the Internet. The FTIT uses an express indexing technique in the matrix level of STAIRS and accelerates the retrieval speed of the dictionary file level by using random access. In order to gain high performance of program execution under Windows environment, Visual C++ was used to design the program of this project [5].

3.1 The Principles of Full Text Indexing Technology

The main purpose of FTIT is to develop high performance search engines on the Internet. It uses a modified theory of STAIRS as the kernel design of the search engine. The FTIT contains four levels of data structures/files, as depicted in Figure 7. The first level of FTIT is called matrix level which has 26×26 entries. There is difference between the matrix level of FTIT and the matrix level of STAIRS. The matrix level of STAIRS has 26×27 entries; the twenty-seventh column of the matrix level is reserved for one-letter words. But the matrix level of FTIT only has 26×26 entries because it does not consider one-letter words. The program of this project automatically omits one-letter words such as "a", "b", and "c," which are meaningless to people on the Internet.

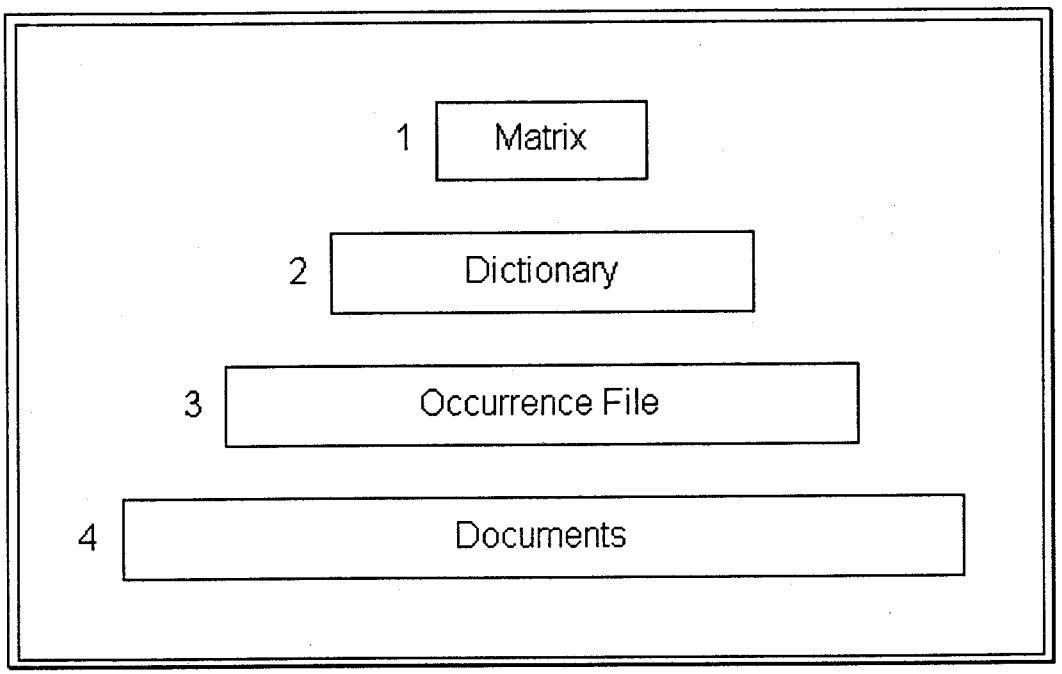


Figure 7. FTIT File Hierarchy

In the matrix level of FTIT, we use an express indexing to search the data in the dictionary level. For example, one assume that the value of "AA" is 5 and the value of "AB" is 12. The meaning of the first number "5" is that the dictionary file contains five records which start with the letter "aa" from the first record to the fifth record. The second number "12" indicates that the dictionary file contains seven

records which start with the letter "ab" from the sixth record to the twelfth record.

The second level of FTIT, called the dictionary level, contains four types of information: the letters of the word, the location of the word, the index of the dictionary level and how many times the word occurs. The purpose of the dictionary level of the FTIT is very similar to the dictionary level of STAIRS. The third level of FTIT is called the occurrence file level. It is comprised of four types of information for each word occurrence:

- the URL of the web page
- line number
- position within the line
- document number

A "document number" is a kind of index created by a program that directly points to a record in the document level of the FTIT. By using the four types of information, one can easily find the real location of the word on the Internet. The fourth level of the FTIT, the Documents level, is where the introduction of the web page is located. Figure 8 shows an example of the four levels of the FTIT file collection. For example, suppose we want to retrieve information for the word "abandon." First we find that the "AB" pointer with a value 5 points to the third record of the dictionary file. Since the "AA" pointer has a value of 2 and the "AB" pointer has a value of 5, we find that the third to the fifth records start with the letters "ab" in the dictionary level. The record "andon" contains information of two occurrences a total of 12 different times in three different web pages.

The pointer of "andon" points to the fourteenth record of the occurrence file. Because the value of the pointer "addon" is 13 and the value of the pointer "andon" is 25, we know that the records the fourteenth to twenty-five in the occurrence file contain some information of the word "abandon." The records in the occurrence file have a pointer pointed to a record in the documents file. For example, we assume that the value of the pointer is 201. We find the pointer pointed to a record of documents file whose number is 201.

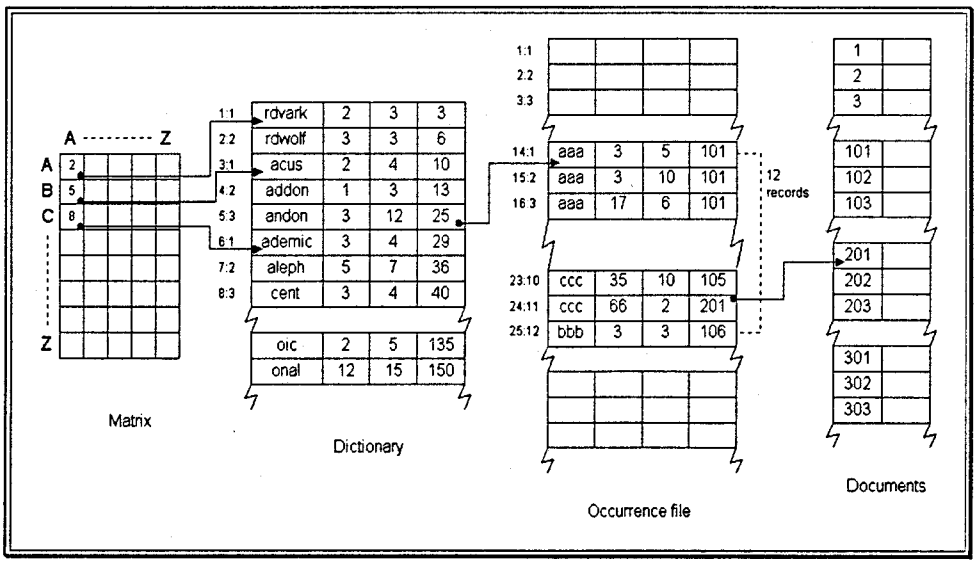


Figure 8. A Sample File Collection

3.2 The Advantage and Disadvantages of FTIT

Full Text Indexing Technology is designed for the Internet. By using this technology one can create a powerful search engine to look for needed information on the Internet. One advantage of FTIT is that it can pinpoint quickly the location of a valid token in a web document and retrieve the token information. In fact, a token is an arbitrary English word, but it does not include the one-letter words. Very few people would like to search for words on the Internet such as a, b, c, and d.

The other advantage of FTIT is that it uses an express indexing technique in the matrix level of STAIRS and accelerates the retrieval speed of the dictionary file level, occurrence file level, and documents level by using random access. The FTIT assists a search engine using the full-text index to search for information on the Internet and it is unlike other general search engines, which search only for the title of a web page. In other words, a search engine using the FTIT as the kernel device is more efficient than other search engines.

The major difference between the Full Text Indexing Technology and the Storage And Information Retrieval System is in the index part. The FTIT uses an express indexing technique to replace the index level of the STAIRS. The express indexing technique can directly calculate the location of the information of a valid token in another level.

The index part of the FTIT search engine is a little complex, so its major disadvantage is that the index is hard to update. In the author's view future modification of FTIT should focus on the index part.

3.3 The Analysis of Files Used in FTIT

In this paper, a search engine is built successfully by applying FTIT and the search engine works very well. But the size of the index files used in this project is worth tracing and discussing. Table 1 shows 50 samples of web sites which are sampled from the Internet. In the FTIT search engine, we use MATRIX.DAT, DICT.DAT, OCC.DAT, and DOC.DAT to save the information of the matrix level, the dictionary level, the occurrence file level, and the documents level.

TABLE 1. The Samples of Web Sites

	Web Sites	URL	File (bytes)
1	The National Anti-Vivisection	www.navs.org/	1,778
2	Audubon	www.audubon.org/audubon/	3,071
3	Those Wonderful Cat	www.eskimo.com/	6,156
4	The American Kennel Club	www.akc.org/	1,929
5	Cybersteed	www.cybersteed.com/	3,838
6	AEC infoCenter	www.aecinfo.com	10,673
7	Classic Short Stories	www.bnl.com/shorts/	10,386
8	Museum Online Resource Review	www.okc.com/morr/	2,208

9	Museums of Paris	www.paris.org/Musees/	4,407
10	Black Star	www.blackstar.com/	4,043
11	Warm Fuzzy	www.txnews.com/	2,498
12	Alabama Shakespeare Festival	www.wsnet.com/	5,819
13	Welcome to the Shakespeare Web	www.shakespeare.com/	3,110
14	Babel	www.babelny.com/	3,999
15	Internet GalleryWalk	artresources.com/guide/	3,712
16	Word	www.word.com/	11,053
17	American Airlines	www.americanair.com/	10,718
18	Continental Airlines	www.flycontinental.com/	7,838
19	CareerPath	www.careerpath.com	4,794
20	American Exoress	www.americanexpress.com	5,042
21	MetLife Online	www.metlife.com/	6,806
22	Fidelity Investments Online	www.fid-inv.com/	11,174
23	The Wall Street Journal	update.wsj.com/	9,132
24	NewsPage	www.newspage.com	16,920
25	FedEx	www.fedex.com	3,039
26	U.S. Census Bureau	www.census.gov	1,917
27	Historical Speeches Archive	www.webcorp.com/	6,196
28	Dell	www.dell.com	11,365
29	Intel	www.intel.com	15,809
30	3Com	www.3Com.com	14,787
31	Yahoo	www.yahoo.com	9,054
32	Computone	www.computone.com/	3,170
33	Adobe	www.adobe.com	7,498
34	Webmaster' Guild	www.webmaster.org/	4,792
35	The Moan and Groan Page	www2.tsixroads.com/Moan/	3,420
36	DigiCrime	www.digicrime.com/	16,395
37	RealAudio	www.realaudio.com	10,992
38	The Distance Education Program New Hampshire College	www.dist-ed.nhc.edu/	3,034
39	National Archives and Records Administration	www.nara.gov	7,454
40	The library of Congress	lcweb.loc.gov/homepage/lchp.html	22,828
41	Welcome to Kaplan Educational Centers	www.kaplan.com	11,448
42	Orange Source	source.syr.edu/	6,325
43	E Online	www.eonline.com	22,715
44	Sleaze	metaverse.com/vibe/sleaze/	21,581
45	Games Domain	www.gamesdomain.co.uk/	897
46	Dermatology in the Cinema	itsa.ucsf.edu/~vcr/Dermcin.html	734
47	Hot Hot Hot	www.hot.presence/hot/	7,416
48	Electro Magnetic Poetry	www.prominence.com/java/po	1,947

		etry/	
49	Zone Interactive	internet-plaza.net/zone/	2,166
50	Epicurious	www.epicurious.com/	3,317

After the FTIT search engine analyzed the sample web pages shown in Table 1, we find 4007 valid tokens. Table 2 shows the consequence of the test. According to the result of the test, if we were to index one million web pages, we would need 2.10 GB of hard-disk space to save the information of the dictionary level and 11.75 GB of hard-disk space to save the information of the occurrence file level. Although we spend 13.85 GB of hard-disk space to store the information of the FTIT search engine, we index 80.14 million valid tokens. Figure 9 to Figure 11 shows the growth of the dictionary level, occurrence file level, and valid tokens.

TABLE 2. The Consequence of the Test

WebSites	Dictionary Level (bytes)	Occurrence File level (bytes)	Valid Tokens
5	7,028	27,792	251
10	26,684	120,456	953
15	35,336	163,728	1262
20	46,900	225,648	1675
25	57,848	392,096	2066
30	66,724	334,872	2383
35	73,612	376,920	2629
40	93,548	517,752	3341
45	105,196	583,200	3757
50	112,196	630,720	4007

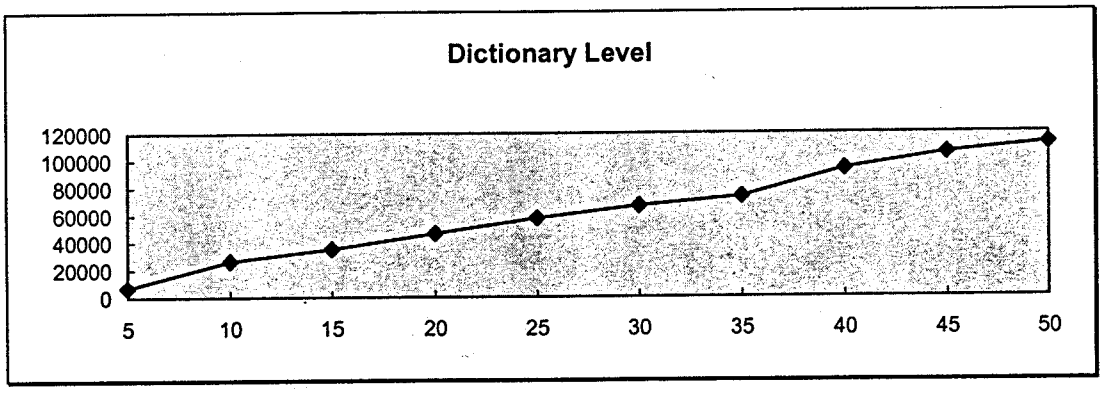


Figure 9. The Growth of the Dictionary Level

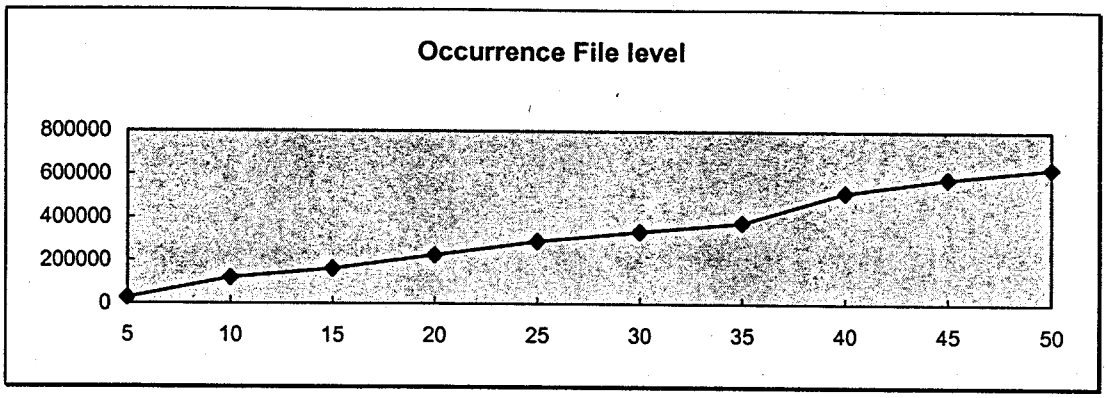


Figure 10. The Growth of the Occurrence File Level

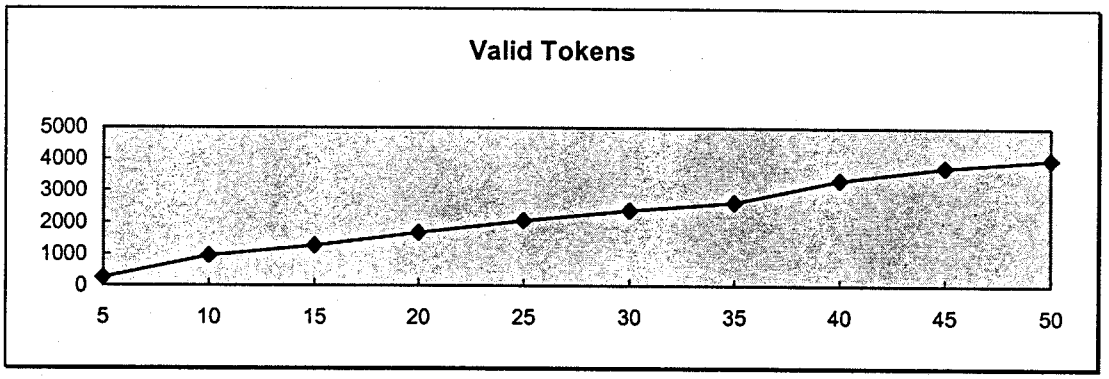


Figure 11. The Growth of Valid Tokens

4. Conclusion

There is so much information on the Internet that it may be difficult for people to find what would be most useful to them. People always use search engines to look for information on the Internet. According to the theory of this paper, we succeeded in developing a kind of full-text-indexed search engine, called an FTIT search engine. The search engine can analyze arbitrary files of HTML which are downloaded from the Internet and it also can find out the relation of valid tokens in these HTML files.

The theory of this paper has proved that it can be the kernel principle of a full-text-index search engine. The most important advantage of Full Text Indexing Technology is that it can find all of the valid tokens in the content of a web page and automatically save the associated information of those valid tokens to databases. Some search engine services claim to have indexed 25 million web pages or more, but these claims are often made on the basis of having indexed links on pages without having ever gone to the pages themselves and indexing their contents. This is like cataloging only the titles of books without knowing their contents.

The FTIT search engine puts all of the valid tokens to its database which actually consists of four databases: Matrix, Dictionary, Occurrence File, and Documents. The search tool of the FTIT search engine looks for information in its databases and retrieves the information for users.

Although the whole design has been completely considered, there are still some areas that can be modified further to improve the project in the future as follows:

1. On the matrix level, in order to maximize the size of the matrix level and to speed up the searching speed, a three- or four-dimensional array can be used to replace the original two-dimension array.
2. On the dictionary and occurrence file levels, the database of the FTIT search engine contains much information about the valid tokens, so we can add a function of evaluating search effectiveness to this search engine.
3. The indexed part of the FTIT is too complex for database maintenance. One can use either bit vectors or a general graph structure [3, p. 198] [6] [7] to simplify lists.

References

[1] Bott, E. (1997, March). *Search-Engine Secrets*. PC Computing. 298.

[2] Date, C. J. (1995). *An Introduction to Database Systems*. New York, NY: Addison-Wesley.

[3] Smith, P. D. & Barnes, G. M. (1988). *Files & Databases: An Introduction*. New York, NY: Addison-Wesley.

[4] Salton, G. & McGill, M.J. (1983). *Introduction to Modern Information Retrieval*. New York, NY: McGraw-Hill.

[5] Arick, M.R. (1996). *The Essential Guide to TCP/IP Commands*. New York, NY: John Wiley & Sons.

[6] Meadow, C.T. (1992). *Text Information Retrieval Systems*. San Diego, CA: Academic Press.

[7] Comer, C.E. (1995). *Internet Networking With TCP/IP Vol 1: Principles, Protocols, and Architecture, Third Edition*. Indianapolis, IN: QUE Corporation.

全文索引技術在進修專校資訊系統上的應用

董俊良 姚威宏 洪榮崇 林秋萍 呂佩怡

國立勤益技術學院 資訊管理科・機械系

摘要

本文主要是改良 IBM Storage And Information retrieval System (STAIRS) 理論，並且使用修改後之技術(Full Text Indexing Technology)，設計一網際網路全文索引搜索引擎(FTIT Search Engine)。全文索引技術的改良與發展，可以幫助使用者更有效率地在浩翰的網際網路上，找尋有用的資訊，FTIT 搜索引擎由三大部分所組成: 1.資源搜尋器(Spider)、2.資料庫(Database)、3.搜尋工具(Search Tool)，資源搜尋器的主要功能是找尋及取回網頁資訊並將有效字的相關資訊存入資料庫中，而 FTIT 搜索引擎的資料庫是一種以知識性資訊做爲基礎所建構的資料庫，它儲存大量的有效字相關資訊，例如有效字的名稱、有效字的屬性、有效字所屬的網頁網址...等等。FTIT 搜尋工具是一種有效字搜尋公用程式，它在資料庫中搜尋特定資訊，並藉由 CGI 將搜尋結果傳給使用者。爲了詳盡說明全文索引技術，本文將藉由 Visual C++、Hypertext Markup Language、CGI、Windows NT network 來做展示。

關鍵詞: Storage And Information Retrieval System(STAIRS), Full Text Indexing Technology (FTIT), 資源搜尋器, 資料庫, 搜尋工具, 有效字。