HRS: A Hypermedia Retrieval System

楊東麟　　陳俊吉
Don Lin Yang and Jiunn Jyi Chen
Department of Information Engineering
Feng Chia University
Taichung, Taiwan 407, ROC

Abstract

The service of the World-Wide Web (WWW), a typical hypermedia system, has been the fastest growing hypermedia application on the Internet. It does not only bring us a new way of representing knowledge, but also a better way of accessing related informaiton.

When users enter the hypermedia world like the World-Wide Web, they will face information overload. It requires an efficient way for users to search related topics. A practical solution is provided by our Hypermedia Retrieval System. In this research we have studied and designed a prototype of the Hypermedia Retrieval System, which was implemented under the World-Wide Web environment. The Hypermedia Retrieval System not only integrates the network resources available on the Internet, but also provides an efficient way for users to find the information they are interested in.

There are four components in our system: the World-Wide Web server, the HTML parser, the query processor and the descriptor database. We use a hierarchical view approach to represent the real world scenario in the World-Wide Web. There are three levels of view approach: global view, local view, and personal view. Each view reflects a different situation that users encounter in the World-Wide Web environment. A prototype system has been implemented in Sun SPARC workstations.

Keywords: Hypermedia, World-Wide Web, HyperText Markup Language, Database, Network.

1. Introduction

As the Internet grows, more and more people rely upon it to do their research. There are a lot of useful information resources on the Internet, such as News groups, anonymous ftp, gopher and Archie servers, and so on. In addition, these abundant information resources extend the scope of our knowledge.

Hypermedia, containing multimedia information under the network architecture, brings us a new way of representing knowledge. The World-Wide Web, a typical hypermedia system, has been the fastest growing hypermedia application on the Internet.

When users enter the hypermedia world like the World-Wide Web, they will face information overload. How can users find the information they need? And is there an efficient way for users to search related topics? What kind of system is required to provide a practical solution?

In this research, we studied the above issues and designed a prototype of the Hypermedia Retrieval System (HRS), which was implemented under the World-Wide Web environment. The hypermedia retrieval system not only integrates the network resources available on the Internet, but also provides an efficient way for users to find the information they are interested in.

We adopt a hierarchical view approach to reflect the real world situation in the World-Wide Web. There are three levels of view approach: global view, local view, and personal view. Each view reflects a different situation that users encounter in the World-Wide Web environment.

2. Background and Related Research

2.1 The World-Wide Web

Early in 1945, the concept of "Universal Information Database" has been formed: data not only accessible to people around the world, but related information that would link to other information easily. The World-Wide Web is an information system based on hypertext, which offers a means of moving from one document to another document (usually called to navigate) within a network. It started at CERN (Collective of European high-energy physics Research Nuclear), now with many participants. It has become the most popular part of the Internet.

2.2 Relevant Terms

The World-Wide Web has come to stand for a number of things, which should be distinguished [1]. These include: a network protocol (HTTP) [2], a markup language (HTML) [3], and an address system (URL) [4].

2.3 Other Protocols

Besides HTTP, the World-Wide Web model contains many other protocols. The World-Wide Web software can pick up information from many information sources, using existing protocols.

These protocols are: file transfer protocol, network news transfer protocol and z39.50 [5].

2.4 Internet Search Tools

The ALIWEB proposes that people should describe the services they provide, in such a way that automatic programs can simply pick up their descriptions, and combine them into a searchable database. The database can be updated regularly (currently once a day), the data is kept up-to-date. Since the ALIWEB does all the work of retrieving and combining these files, people only need to worry about the descriptions of their own services. Therefore, the information is likely to be correct and informative. Since only these small description files need to be gathered, there is little overhead [6].

The Lycos is a catalog of the Internet. It searches the World Wide Web every day (including Gopher and FTP servers) and builds a database of all the web pages it finds. The index is updated weekly [7]. The Lycos philosophy is to keep a finite model of the web that enables subsequent searches to proceed more rapidly. The idea is to prune the "tree" of documents and to represent the clipped ends with a summary of the documents found under that node [8].

The WebCrawler is a tool that solves the resource discovery problem in the specific context of the World-Wide Web. It provides a fast way of finding resources by maintaining an index of the Web that can be queried for documents about a specific topic. Because the Web is constantly changing and indexing is done periodically, the WebCrawler includes a second searching component that automatically navigates the Web on demand. The WebCrawler is best described as a "Web robot" [9].

The UCSTRI is a WWW service which provides a searchable index over thousands of existing technical reports, theses, preprints, and other documents broadly related to computer science. The UCSTRI requires two major modules. An index builder polls numerous FTP sites for item information to construct a master index file. The list of sites and their characteristics are the only components of the system's operation that must be maintained by hand. A search engine then processes queries to return citations and hypertext links to appropriate items. The overall structure is similar to other indexing systems such as the ALIWEB [10].

2.5 Search Techniques

The existing search techniques on the WWW fall into two main categories: hypertext browsing and keyword searching.

Navigating or browsing in hypermedia systems traditionally involves manual traversal of hyperlinks. It is well known that this technique by itself cannot be used to search hypermedia system containing more than several hundred nodes. Users quickly become lost because of the size of the search space [11].

In keyword searching, users input keyword items as query parameters to do query process. The keyword searching combines the keyword items and the combination of the boolean operators such as "and", "or", and "not".

The search tools on the Internet have the following drawbacks:

1. Centralized process:
   All these tools on the Internet use a centralized data base. Maintaining such a large amount of data is difficult.
2. A waste of network bandwidth and other resources:
   It is possible that a lot of people search for a specific document simultaneously. This would take up the whole network bandwidth to transmit such a large amount of information. In addition, it will increase the system overhead.

To solve the above problems, we offer the following approaches:

1. Distributed process:
   Change the use of databases from a centralized control to a distributed approach. A local site can maintain its own database. When a request is made, the local sever can connect other site to provide necessary information. There are two ways of doing it:
   a. Automatic linking:
      By connecting other server one can get requested information automatically. To the user, such process is transparent.
   b. Hint:
      By giving a proper hint to the user it can help users find information faster.

   When a server finds the information, it does the "cache" process to hold the information for the subsequent query action.
2. Tracking Process:
   The major reason of wasting network bandwidth is that too many people log on to search (browse) on-line information at the same time. System can change the query model depending on the network condition: the number of users, the data size, or the response time. It can also use different ways of sending search result, like e-mail. So the user can go off-line and receive the information result at its own site. The format of the result they receive can be a

simple index, or an HTML format document filled with hyperlinks.

3. Integrated environment:
   The major effort of our system is to provide an integrated environment for users to locate (or retrieve) the heterogeneous information they need.

We believe this proposal can reduce the overhead of the retrieval system and utilize the network bandwidth (resource) more efficiently.

The search tools and techniques mentioned above provide an efficient way for users to locate and retrieve the information they need. The major problem of these systems is to focus on the "text search" only. Keyword searching provides users a direct way to describe the information they need. Users don't need to learn the syntax of the query method. In the hypermedia world like the World-Wide Web, there are a lot of diverse information, such as audio, video, and image. To develop a global way to find (or locate) these heterogeneous information is not easy. Many object-oriented approaches are used in the multimedia query processing. In the future of the World-Wide Web development, there will be more efficient ways to process these different types of media.

3. System Architecture

3.1 Architecture Overview

Our system flow can be divided into two parts: the back end process and the front end process. The back end process reflects the functionality of our system. The front end process reflects the user's search behavior.

The back end process can be divided into 4 steps :

1. Analyze all the HTML documents at the server end.
2. Parse the HTML tags for each HTML document.
3. Store related information such as keyword items and link description to the descriptor database.
4. Receive the input parameter, search the descriptor database according to the user's input, and send the result in the proper format to the user.

The front end process can be divided into 3 steps :

1. Users log on to enter our service page.
2. Keywords are typed in the input field. The keyword describes the needed information.
3. Our system will return an appropriate result according to the user request.

We construct our system under the World-Wide Web environment. It consists of four components: the World-Wide Web server, the query processor, the HTML parser, and the descriptor database. The HTML parser is responsible for analyzing all the HTML documents at the server site, parsing the HTML tags for each HTML file, and writing the link information and description to the descriptor database. The query processor is responsible for receiving the user query parameters, searching the descriptor database, and returning the search result in an appropriate format. We will describe their functions in detail in Section 3.3. The architecture of our system is shown in Figure 3.1.

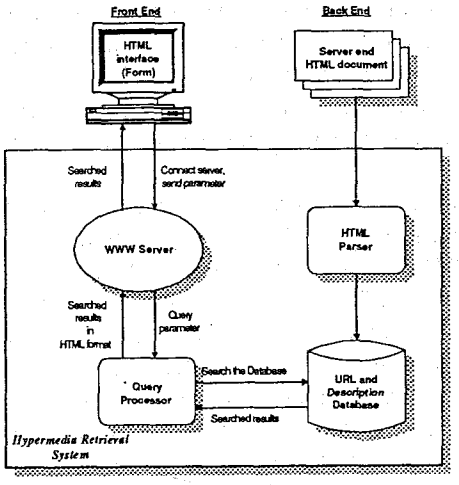

Fig. 3.1 The architecture of Hypermedia Retrieval System (HRS)

3.2 Hierarchical View Approach

We use a hierarchical view approach for our system service. Through the concept of view approach, we try to provide a better way for users to find diverse information on the World-Wide Web sites.

The view approach is the main idea of our system. It can be divided into three levels of "view": global view, local view, and personal view. Each view has its own feature to reflect the different situation and environment. Figure 3.2 shows a hierarchical view diagram.

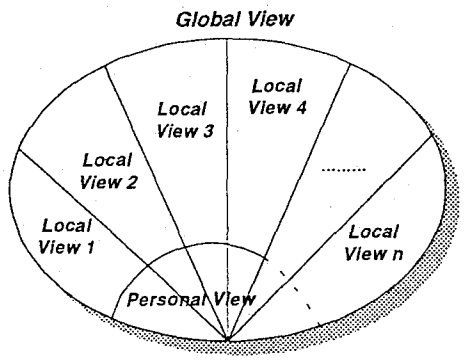

Fig. 3.2 The Hierarchical View diagram

Global view:

A World-Wide Web server uses hyperlinks to connect all related information for different types

of data, such as text, audio, image, video, and so on. Users can navigate to retrieve all the information they are interested in through the help of these hyperlinks. We can image a hypermedia world (or even a real world) is like pieces of information everywhere with hyperlinks to represent their relationship. We can get all the interesting information on the Web by using a World-Wide Global Index to retrieve relevant information. It is not practical to create a global index including all the information on the Web. Instead, we focus on the hyperlinks specified by the information providers. These hyperlinks are what the users would be interested in. Such an information model extends the scope of our knowledge in a global scale.

Local view:
Though a global view provides us a "world-wide" index to use, it has the following problems:

1. Space capacity: To store all useful information on the Web, it would require a very large amount of storage space.
2. Complexity: To maintain such a big global index, one needs a lot of efforts.
3. Time restriction: To retrieve a vast amount of information through network, it would be considered unacceptably high, especially when the network traffic is busy.
4. Network bandwidth: To retrieve distributed information, one needs a higher network bandwidth when the data size is huge.

If we want to avoid the problems existed in a global view, we can put our attention on a smaller scale, such as a single site. This is our "local view". Local views help us learn very well about the local site information where we are interested in the most. At the local site, information accesses are much more efficient too.

Personal view:
Personal views are focused on tracking the path of the user's navigation on the Web. By using the index from the server, users can specify their needs for the server to provide with the appropriate result. Along the path being visited, an HRS collects their link information to generate a personal view. As the navigation tour ends, one would have created a hypermedia document that contains all the information presented in the tour. It provides a simple way for users to revisit the information that they are interested in. User can retrieve information according to their specific need.

3.3 System Components

3.3.1 The World-Wide Web Server

It is a typical server like any Web site. We construct our system under the server homepage which provides the user a fast way to preview the information available at this site. We can provide different styles of service appearance at the Web site. Users can use service from our system to search the information they need.

The responsibilities of the World-Wide Web server are:

- To receive the input parameters from the user, and send them to the query processor
- To receive the search result from the query processor, and send the result to the user

3.3.2 The Query Processor

The responsibilities of the query processor are:

- To receive the query parameters sent by the World-Wide Web server
- Search the descriptor database according to the parameters
- Return the search result in the HTML format file filled with related links and description

The query processor receives the query parameter from the environment variable which is set by the server. It parses the parameter to get the actual query value, and uses the value to search the descriptor database. When the result is found, it transforms the search result to an HTML format file filled with a list of information including links and their description.

Receiving the user input is the HTML form's first task. Then it sends the user input as parameters to the query processor. The query processor then proceeds with the search action. Figure 3.3 shows the information flow of the query processor.

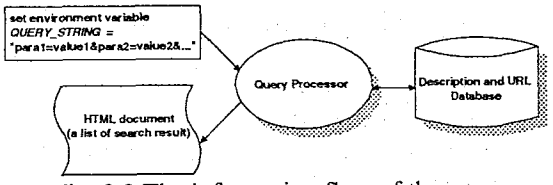

Fig. 3.3 The information flow of the query processor

3.3.3 The HTML Parser

The HTML parser is typically a state machine. The responsibilities of the HTML parser are:

- Retrieve all the HTML files in the World-Wide Web server site
- Parse the tags for every HTML file
- Record the link information and their description
- Write results to the descriptor database

When the HTML parser receives the href attribute in the HTML file, it records the link information. If it finds an anchor like ">", that means the tag is ended, then we are ready to receive the description. The input of the HTML parser is all of the HTML files in the server end, and the output is link

information (URL) and their description. The HTML parser parses every HTML document at the server end. Like a compiler parser, it analyzes the syntactic structure of the document. Figure 3.4 shows the HTML parser.

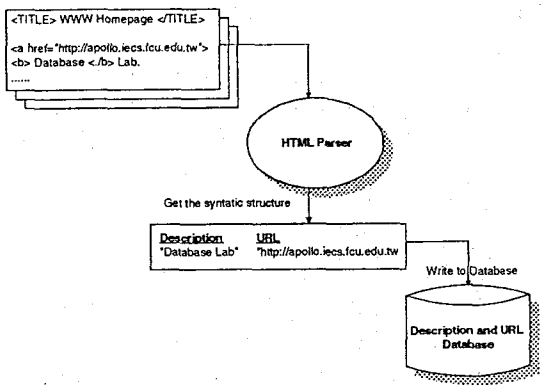

Fig. 3.4 The HTML parser

3.3.4 The Descriptor Database

The descriptor database is a simple data storage. It is designed to store the parsing result of the HTML parser, including keyword items, link information like URLs and their description.

The descriptor database is used for processing the searching request to respond to the user's search request. Figure 3.5 shows the descriptor database.

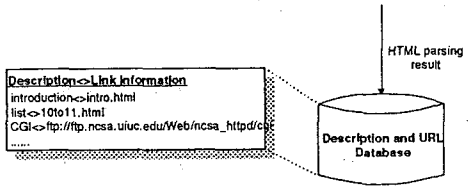

Fig. 3.5 The descriptor database

4. System Implementation

4.1 System Environment

We construct our system prototype under the World-Wide Web environment. The server (NCSA httpd) is installed on a Sun SPARC station 20 with SunOS 4.1.4. The other equipments (PCs and workstations) are connected via Ethernet. Figure 4.1 shows the environment of our network laboratory.

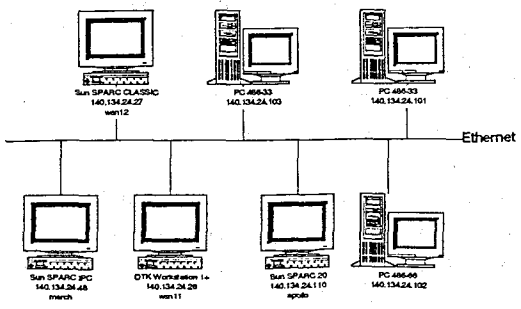

Fig. 4.1 The system environment

4.2 The Implementation of the HRS

4.2.1 The HTML Parser

The HTML parser creates all the index information for the hyperlinks of the HTML files at the Web site. It belongs to the back end process. The responsibilities of the HTML parser are:

- Process all the HTML files at the World-Wide Web server end
- Parse the tags for every HTML file
- Receive the href attribute and record the link and description
- Write results to the descriptor database.

The input of the HTML parser is all the HTML files in the server end, and the output is link information (URL) and its description.

We use C language to implement the HTML parser. The main procedures of the HTML parser are as follows:

1. **CreateAllHTML**(): Get the HTML file directory path as the parameter. The procedure creates all HTML files from the directory path recursively.
2. **HTML_character**(): Parse the "A"nchor tag and the "href" attribute. Note the description and link information (URL).
3. Write the results to the *resource.db*

Figure 4.2 shows the flowchart of the HTML parser.

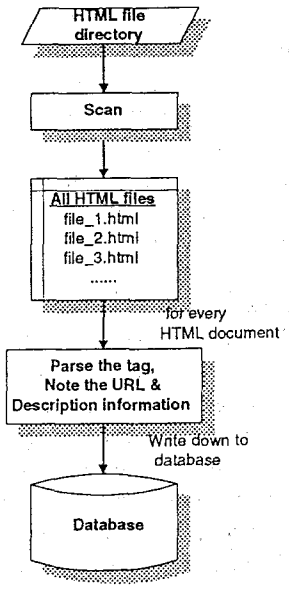

Fig. 4.2 The flowchart of the HTML parser

In HTML, there are a lot of tags used for improving the printing, but these are meaningless tags to us. All we are interested in are: the anchor tag <A> or </A>, the attribute href, the link description (URL), and the description used for keyword matching later. So we ignore the useless tags, and keep the useful information. The index

information that the HTML parser creates is for the hyperlinks of the HTML files in the Web site. We did not index the whole document except the hyperlinks because of the space limitation and other problems of a global view in Section 3.2. We use the URL as our index information because it provides a direct link to the related information.

When the parsing process and output process are finished , the final results will be written to the descriptor database. The *resource.db* is the actual database to store the related information.

4.2.2 The World-Wide Web Server

Our system is implemented under the World-Wide Web environment. The hypermedia retrieval system is put under the World-Wide Web server homepage. The user interfaces of our system are developed by using the fill-out forms. A fill-out form is a tag in HTML. It provides a "two-way" communication under the World-Wide Web. Fill-out forms allow users to return information to the World-Wide Web server. We use it as the interface for database accesses. It provides the user an input field, and transmits appropriate results to the user. Figure 4.3 shows the typical example of a fill-out form in HTML format. Figure 4.4 shows the result of the fill-out form.

```
<TITLE> FCU HRS </TITLE>
<CENTER>
<H1> World Wide Web Services</H1>
<H2> HyperMedia Retrieval and Database Retrieval Applications </H2>
</CENTER>
<HR>
<FORM ACTION =
"http://apollo.iecs.fcu.edu.tw/cgi-Service/select">
<BLOCKQUOTE>
<BLOCKQUOTE>
Select the required service:
<SELECT NAME="SELECT">
<OPTION selected VALUE="0"> Hypermedia Retrieval Application
<OPTION VALUE="1"> Database Retrieval Application
</SELECT>
<INPUT TYPE="SUBMIT" VALUE="Submit">
</BLOCKQUOTE>
</BLOCKQUOTE>
</FORM>
<HR>
```

Fig. 4.3 The HRS fill-out form in HTML format

The service of our system is based on the three levels of hierarchical view approach: global view, local view, and personal view. So the implementation is done accordingly.

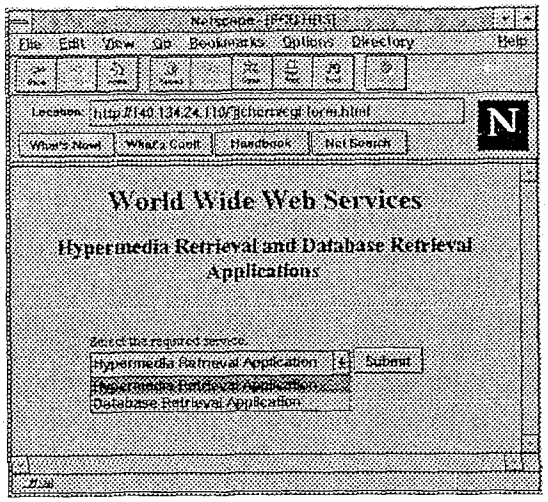

Fig. 4.4 The HRS fill-out form result

Global view:

A global view provides a "world wide" view of the information on all the Web sites. So the implementation of the global view needs to collect the information index of each site. By getting the index information such as the directory path of their HTML files, we can build a global index. A complete global view is only an ideal situation due to the problems described in Section 3.2. In reality we would construct the global index to include as many sites as an HRS implementation can afford. In our prototype implementation we create a global view of our two laboratory sites.

Local view:

The local view deals with a single site. The implementation is done by collecting all the information index at a local site. Given the HTML file directory path of a local site, the HTML parser creates a local index. This index contains all the index data at this site. Each Web site can establish its own local view by using the HTML parser.

The combination of the global view and the local view provides users flexibility to choose the range of information they are interested in. The global index merges the contents of each local site index. These contents can be updated periodically. The diagram is shown in Figure 4.5.

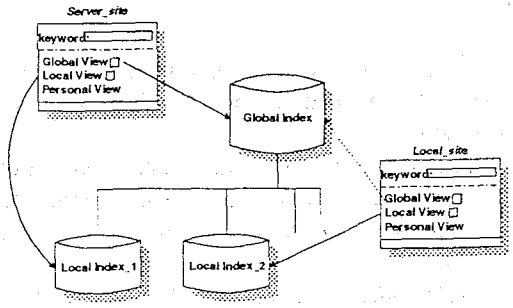

Fig. 4.5 The combination of global view and local view

Personal view:

The implementation of our personal view provides two basic functions: Creating a personal view and Revisiting a personal view. Because we are not able to modify or rewrite the Web browser Netscape, the personal view is implemented in a semi-automatic mode. It works as follows. First, a user inputs keywords to search the descriptor database. If any match is found, it returns an HTML format file filled with hyperlinks. The user uses these hyperlinks to navigate through the Web to view the information he is interested in. When the user finds a segment of information worth keeping, he can use the browser to make a copy and save it. For example, using the Netscape users can use mouse to move a cursor to mark the scope and make the copy by clicking the right button of the mouse. The copied information can be pasted in a document. It can be saved to the user site whether as a plain file or an HTML file. Then users can use an edtior like the Microsoft Word to edit the file to build a personal document. If the user wants to edit an HTML file, he must add some HTML tags. Then the personal document is created. If the user wants to revisit the information, he can choose the option of revisiting a personal view to get a help menu. This second function provides the user a hint to open the saved personal document.

4.2.3 The Query Processor

The responsibilities of the query processor are:

- Receive the user query typed in the HTML form
- Search the descriptor database and find the appropriate query result
- Return the query result in HTML file filled with related links and description corresponding to the user query.

When users receive the query result lists, they can choose the link to go directly to the document.

To get the user input is the responsiblity of a HTML form. So the input of the query processor is the query parameter and the output is an HTML file filled with the link information corresponding to the user query. The HTML file is created by the query processor. It contains a lot of lists for users to choose.

We use C language to implement the query processor. The procedures of the query processor are as follows:

1. **query_parse**(): Get the query parameter, store the related value, and search the database *resource.db* according to the parameter.
2. **GetField**(): Extract the link information (URL) and the description in the string format.
3. **TransForm**(): Transform the query result from the string type to the HTML format.

Figure 4.6 shows the flowchart of the query processor.

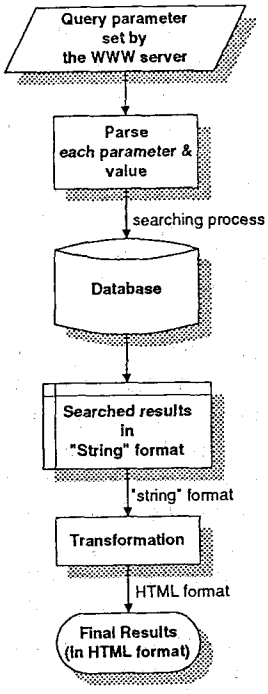

Fig. 4.6 The flowchart of the query processor

The result of the search in the descriptor database is the "string" type. So we have a "tranformation" procedure to change these string results into the HTML format, then these results can be sent to the server. The users use their World-Wide Web browser to view these results.

After the above steps are completed, the user will get the result information they requested: a list of searched result which cantains hyperlinks. So users can retrieve the information they need through the help of these hyperlinks taking them to a specific location.

4.2.4 The Descriptor Database

We use the file *resource.db* as the database to store the link infomation and description from the HTML parser. The content of the databse is a list of string type. Each string contains the description and link information. Figure 4.7 shows the contents of the descriptor database.

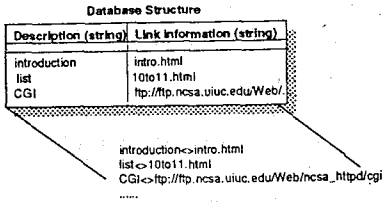

Fig. 4.7 The contents of the descriptor database

We can also use the commercial databases such as Infomix, Oracle, or Sybase to store information. All we need to do is to write an external program.

The Common Gateway Interface (CGI) will transform the excution result to a proper format. Then the user can view this result through the World-Wide Web browser.

5. Conclusions and Future Research

In this thesis, we have discussed the design of a hypermedia retrieval system. The system consists of four parts: the HTML parser, the descriptor database, the query processor and the World-Wide Web server. By creating and storing index automatically, the database provides keyword items and their related addresses (URL's) to locate information. Our system is based on the client-server model under the World-Wide Web environment. The sever site automatically creates and maintains index in a fixed period of time.

The major task of our hypermedia retrieval system is to provide an efficient way for users to find information from various resources. Presenting a useful index needs the following procedures:

- Retrieve all the HTML files at the World-Wide Web site
- Create the index automatically
- Transmit the index to the user interface
- Users retrieve the information they need

The execution efficiency of these procedures affects the overall system performance. Currently, our prototype system is implemented under the World-Wide Web server homepage. This can help users quickly find every piece of useful information at that site.

Besides providing an efficient way for users to search information, our system also integrates the network resource available on the Internet. All these efforts are based on the three levels of view approach we propose: global view, local view, and personal view. Each of the three views is used in the different environment and situation.

Keyword matching is the method we use for searching information. We use the full-text match to search information on the Web. One of our future work is to adopt the Natural Language Processing (NLP) technology in our system. It should have a dictionary function and would not identify keywords just as a combination of characters. We hope the NLP technology can improve the keyword recognition efficiently, so the users don't have to learn a complex query language.

Another future work of our system is to add management functions. Services provided in our system could increase the network traffic. We would like to add some management functions to monitor the network, such as the number of users logged on and the amount of data being transmitted.

In our system implementation, we use the file *resource.db* as our descriptor database. In the future we can use commercial databases to store these information. Then we can access data from databases in a general way such as using SQL. The database can also store various types of multimedia information for retrieval. A complete service model is shown in Figure 5.1.

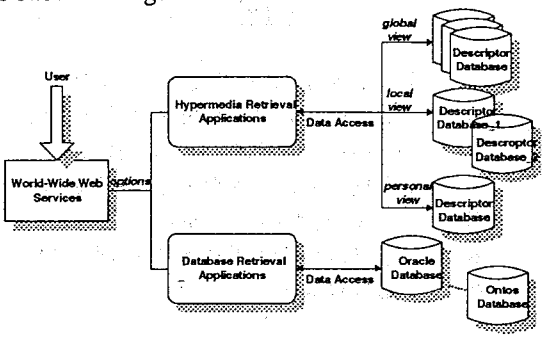

Fig. 5.1 A complete Web service model

6. References

[1] Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen, and Arthur Secret, "The World-Wide Web", Communications of the ACM, August 1994, Vol. 37, No. 8, pp. 76-82.

[2] "Overview of HTTP", http://www.w3.org/hypertext/WWW/Protocols/Overview.html

[3] "HyperText Markup Language: Working & Background Materials",http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html

[4] "UR* and The Names and Address of WWW objects", http://www.w3.org/hypertext/WWW/Addressing/Addressing.html

[5] "Protocols used in the World-Wide Web", http://www.w3.org/hypertext/WWW/Protocols/RelevantProtocols.html

[6] "Introduction to ALIWEB",http://web.nexor.co.uk/public/aliweb/doc/introduction.html

[7] "Lycos: Frequently Asked Questions", http://lycos.cs.cmu.edu/lycos-faq.html

[8] "Lycos Project Description", http://lycos.cs.cmu.edu/lycos-post-01.html

[9] Finding What People Want: Experiences with the WebCrawler", http://webcrawler.cs.washington.edu/WebCrawler/WWW94.html

[10] Marc VanHeyningen, "The Unified Computer Science Technical Report Index: Lessons in indexing diverse resources", http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/Vanheyningen/paper.html

[11] Howard Beck, Amir Mobini, and Viswanath Kadambari, "A word is worth 1000 pictures : Natural languages access to digital libraries", http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Searching/beck/beckmain.html